# Combinatorial Tasks as Model Systems of Deep Learning

HARVARD
Kenneth C. Griffin

GRADUATE SCHOOL
OF ARTS AND SCIENCES

# DISSERTATION ACCEPTANCE CERTIFICATE

The undersigned, appointed by the

Harvard John A. Paulson School of Engineering and Applied Sciences
have examined a dissertation entitled:

"Combinatorial Tasks as Model Systems of Deep Learning"

presented by: Benjamin L. Edelman

Signature Sham Kakade
*Typed name*: Professor Sham Kakade

Signature LG Valiant
*Typed name:* Professor Leslie Valiant

Signature Boaz Barak
*Typed name:* Professor Boaz Barak

May 2, 2024

# Combinatorial Tasks as Model Systems of Deep Learning


A dissertation presented
by
Benjamin L. Edelman
to
The John A. Paulson School of Engineering and Applied Sciences


in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy
in the subject of
Computer Science


Harvard University
Cambridge, Massachusetts

May 2024

Dissertation advisors: Professors Sham Kakade and Leslie Valiant Benjamin L. Edelman

# Combinatorial Tasks as Model Systems of Deep Learning


## Abstract

This dissertation is about a particular style of research. The philosophy of this style is that in order to scientifically understand deep learning, it is fruitful to investigate what happens when neural networks are trained on simple, mathematically well-defined tasks. Even though the training data is simple, the training algorithm can end up producing rich, unexpected results; and understanding these results can shed light on fundamental mysteries of high relevance to contemporary deep learning.

First, we situate this methodological approach in a broader scientific context, discussing and systematizing the role of *model systems* in science and in the science of deep learning in particular. We then present five intensive case studies, each of which uses a particular combinatorial task as a lens through which to demystify puzzles of deep learning.

The combinatorial tasks employed are sparse Boolean functions, sparse parities, learning finite group operations, performing modular addition, and learning Markov chains in-context. Topics of explanatory interest include the inductive biases of the transformer architecture, the phenomenon of emergent capabilities during training, the nuances of deep learning in the presence of statistical-computational gaps, the tradeoffs between different resources of training, the effect of network width on optimization, the relationship between symmetries in training data and harmonic structure in trained networks, the origins of the mechanisms of in-context learning in transformers, and the influence of spurious solutions on optimization.

# Contents

To Jenny, with high probability.

Surbhi Goel, Sham Kakade, Eran Malach, Cyril Zhang, Depen Morwani, Ezra Edelman, Nikolaos Tsilivis, Boaz Barak, Costin-Andrei Oncescu, and Rosie Zhao wrote with me the papers that form the basis of this work. Toiling, brainstorming, disagreeing, playing, and being surprised together with these and my other collaborators is what made my Ph.D., a traditionally oft-demoralizing affair, into such a joy. So thanks are also due to my other co-authors, including: Preetum Nakkiran, Gal Kaplun, Dimitris Kalimeris, Tristan Yang, Fred Zhang, Yonadav Shavit, Brian Axelrod, Enric Boix-Adserà, Siddhartha Jayanti, Hanlin Zhang, Danilo Francati, Daniele Venturi, Giuseppe Ateniese, Gustaf Ahdritz, Tian Qin, Nikhil Vyas, Usman Anwar, and Ekdeep Singh.

I came to Harvard in part because I was inspired by Leslie Valiant's investigations of learning, evolution, and cognition through the lens of computation, and of the limits of computation itself. As my advisor, he gave me the freedom to explore widely and deeply. It has been a privilege to have Les as a mentor.

This wide-ranging spirit permeated the entire Harvard Theory of Computation group, which was my academic home for much of my time at Harvard, especially before the tragic interruption of the COVID-19 pandemic. The theory group has been a wonderful, warm community—among many others, I would like to thank my fellow students Yonadav Shavit, Preetum Nakkiran, Thibaut Horel, Santhoshini Velusamy, Jayshree Sarathy, Chi-Ning Chou, Boriana Gjura, Kelly Zhang, Fred Zhang, Jarosław Błasiok, and Prayaag Venkat for being such good company during those pre-pandemic golden years. I also learned much from the theory faculty, especially Boaz Barak, Madhu Sudan, Salil Vadhan, and Cynthia Dwork.

Boaz in particular has been an advisor to me in all but name for the past few years. I am grateful for not just his signature nonstop creative energy, but also his thoughtful compassion.

Two events occurred during the middle period of my Ph.D. that jolted me onto the research agenda represented by this thesis. First: the release of GPT-3 in 2020. That such ...intelligence? ...could emerge from straightforward neural network training felt like science fiction, and I felt I needed to understand it.

Second: at the precise midpoint of my Ph.D., I was hired as a summer intern at Microsoft Research New York to work with Cyril Zhang, Surbhi Goel, and Sham Kakade. Like me, they were keen on demystifying the new transformer revolution, and together we wrote Edelman et al. (2022). Which led to another project (Barak et al., 2022), which led to another (Edelman et al., 2024a), and has continued to snowball ever since. Surbhi and Cyril became my intellectual co-conspirators and extraordinarily generous mentors. Together we cultivated (or stumbled upon) a style of research, a way of looking at deep learning, which this thesis is all about. Eran Malach also became essential to this circle of scientists, and I am so grateful he joined Harvard last year so we could launch a possibly inadvisable number of exciting new projects together. Portions of the Introduction (Chapter 1) have their roots in our collective discussions and draft writings.

I don't think this is how the tech intern pipeline usually works, but after that summer, Sham joined Harvard as a professor, and he agreed to become my co-advisor. Since then, he has kept me grounded and kept me on my toes. His savvy advice and his mentorship have been so helpful, and his well-honed research instincts have been essential to the snowball's momentum.

Speaking of snowballs—in my first year of graduate school, some fellow students, spearheaded by

Preetum Nakkiran (and soon joined by Boaz) started an informal reading group about deep learning theory. At the time I felt like an utter novice—I didn't even know how linear regression worked. We—Yamini Bansal, Gal Kaplun, Dimitris Kalimeris, Fred Zhang, Nikhil Vyas, Kelly Zhang, and others—learned about deep learning and caught up to the cutting edge together. Over time, our reading group became a popular seminar series, and the reading group participants started writing papers. I want to thank Preetum in particular not just for kickstarting this snowball (which became the ML Theory group, now known as the ML Foundations group), but also for pushing me and everyone else to broaden our conception of what theory means.

Post-pandemic, the ML Foundations group has been my academic home, and I have poured a lot of my time and energy into building it up into a vibrant, joyous, curious community. I want to thank everyone in this community (to be precise, the CS component of the ML Foundations group) for contributing to this spirit. In particular, those of us who were there in the pivotal 2022-3 year—Gal, Nikhil, and the first-years of that year: Anat, Costin, Depen, Gustaf, Rosie, and Sunny.

My thanks go to all my friends, near and far, for enlivening my time inside and outside Cambridge during these years. My mathematical career is, in all likelihood, causally downstream of the innumerable mathematical conversations I've had over the decades with my best friend, Aaron Berger. Aaron started a Ph.D. in mathematics at MIT at the same time I started mine in computer science at Harvard. He has not only been an invaluable sounding board for some of the research ideas in this thesis (and many others), but also the best companion in music, cooking, and conversation I could have asked for.

I am incredibly fortunate to have a loving, sprawling extended family, including grandparents who have always been eager to hear of and take pride in my exploits, academic and otherwise, regardless of how much I can make comprehensible. My cousin William Meyer happened to begin law school at Harvard at the same time I started my program; it was a pleasure to live with him during my second year.

To my parents I owe everything. They are my best advisors.

Everything is also owed to my siblings. I am the eldest of four, and my life and identity are inextricably tangled up with those younger three. As pertains to this thesis: I discovered the wonders of computation alongside Yafah when we were kids, and she continues to challenge and surprise me, in computer science and other realms. Ezra, meanwhile, is literally responsible for much of Chapter 8, has joined our snowballed collaboration cadre, and in general is on the same wavelength as me more than anyone else. And it has been a joy to have Eli (proudly not a computer scientist) in the Boston area during these past few years.

This dissertation is dedicated to Jenny, the most important person in my life for the first 98% of my time in graduate school. I am so thankful that I have *her* as my partner. Really, the term "other half" is better—I understand it now. The centerpiece of my time in grad school was our wedding, and I am delighted to have acquired such awesome additional family in the bargain.

Finally, last and smallest, I must thank Esther for being no help whatsoever. My thanks go to Jenny, Jenny's mom Carol, and Eli, for taking care of Esther during my days scraping together this screed. Esther—I am so, so excited to see the emergent phenomenon you become.

# 1

# Introduction

## 1.1 A Parable

Once, seven sages, voyaging together in unfamiliar lands, came across a strange object.

The first sage said, "This appears to be a sort of brain. See how it processes information using billions of neurons connected by synapses."

The second sage shook her head. "This is an engineered machine if I ever saw one. "

The third sage said, "Not just any engineered machine—a general-purpose computer. These neurons are gates that combine to form circuits, perhaps even Turing machines. To understand the computer, understand the algorithms it implements."

The fourth sage would have none of it. "All this talk of what *is*—the important thing is how it *came to be*. What we have before us is a snapshot of a dynamical system, like an ecosystem or a hurricane."

The fifth sage sat in awestruck contemplation. "What we see here is a mere shadow of Infinity. It is but a small part, a meager approximation, of a limitless, elegant Form. It is that infinite Form which we ought to study."

The sixth sage looked around at the others. "Do you not see? The mechanism is a mirror for the data that it consumes. You focus on the mechanism itself, when what matters most is the data."

The seventh sage, who was considered by some to be the wisest, said, "This thing is capable of working wonders. It is meant to be used and perhaps tinkered with. But to attempt to understand it is a fool's errand." He turned aside and continued trudging along the path.

## 1.2 What Kind of Science?

Deep learning, as an engineering discipline, has a fairly unified paradigm right now. Its central concept is the standardized *benchmark*, and its default methodology is twofold:

1. Scale up the resources of training as much as possible, with the optimal balance of these resources guided by empirical *scaling laws* (Kaplan et al., 2020; Hoffmann et al., 2022)

2. Incrementally improve the Pareto frontier of performance in terms of resources by a process of community-wide tinkering. Innumerable variations on training and inference pipelines are empirically tested by researchers, and those which lead to benchmark gains are widely adopted.

The paradigm has shifted over the past decade—the focus on scaling laws has only become dominant with the rise of large language models (LLMs) over the past several years. And it will change in the future—for instance, major benchmarks for LLMs are becoming increasingly expansive and complicated (Srivastava et al., 2022; Liang et al., 2023), and are increasingly being supplanted by direct human or LLM evaluation (Zheng et al., 2024).

Meanwhile, the *science* of deep learning is still pre-paradigmatic and fragmented. By the science of deep learning we mean the study of artificial neural networks (henceforth simply *neural networks* or *models*), often centered around the goal of explaining unexplained phenomena that appear during the practice of deep learning. This overlaps with, but is distinct from, employing aspects of the scientific method in order to engineer better models. The immediate goal is not *performance*, but *understanding*.*

As the parable illustrates, there are many ways of looking at a neural network. The perspective, methodological toolkit, and modes of inquiry of a computer scientist can be dramatically different from those of a statistician, or a neuroscientist, or a physicist, or a linguist, or a psychologist. Because the science of deep learning is defined by its subject (artificial neural networks trained with iterative optimization algorithms) rather than its methodology, the viewpoints of each of these fields and various others can be simultaneously relevant. This extends to subfields as well: optimization, natural language processing, computational learning theory, high-dimensional statistics, control theory, statistical mechanics, etc. all have something to bring to the table.

Situated at the nexus of these disparate disciplines, the science of deep learning is emerging as a discipline in its own right. It liberally incorporates approaches from its predecessors. Of particular importance are motifs of scientific inquiry which can be found across many of the established areas.

The focus of this dissertation is one such motif: the *model system*. Examples include economic

*Whether understanding is pursued as a means towards more performant models, or models that are better for humanity, or is thought of as a terminal goal, can influence which research questions are perceived as most compelling.

models, cell cultures, model organisms such as *C. elegans* and lab mice, particle accelerators, the Ising model (Baxter, 2016), agent-based models (Gilbert, 2019), and Turing machines. A model system is a means to an end. Each of these settings is the object of intense scientific study ...for the purpose of understanding *other* (more "realistic") settings, not the model system itself.

Why would scientists spend so much time studying one system when they really care about another? In order for this to be worth it, a model system need to be exceptionally fertile ground for producing scientific insights. At the same time, it needs to be sufficiently reflective of settings of interest in important aspects, so that the insights learned from the model system transfer. In short, a model system needs to satisfy two desiderata: scientific *productivity* and *transferability*.*

### Productivity

- Can experiments on the model system be run efficiently and cheaply?
- How amenable is it to mathematical analysis?
- Is the system particularly transparent? The nematode *C. elegans* is literally an optically transparent organism, which enables scientists to easily make observations of the worm that would otherwise be far more difficult (Corsi et al., 2015). Other model systems may be more metaphorically transparent to observation and explanation.
- Is the system controllable and extensible? Are there (figurative) knobs that can be tuned to change the nature of the setting?

### Transferability

- Does the model system share important features with real-life settings of interest, such that insights from the former transfer to the latter? This doesn't necessarily require that the high-

*See Gabaix & Laibson (2008) for a more fine-grained list of desiderata for the case of economic models.

level behavior of the settings is similar–it could alternatively be the case that lower-level aspects of the settings are shared. Sometimes, unrepresentative "extreme" cases (e.g. particle accelerators) are more conducive to uncovering shared underlying mechanisms.

- Can investigations on the model system be relevant for a wide spectrum of real-life settings? This can be achieved through extensibility, the fourth bullet point above.

Regarding the last bullet point—an alternative approach is to devise or discover new model systems for every phenomenon of interest. This approach can be useful, providing an otherwise impossible to achieve level of fit between model system and phenomenon. The physiologist August Krogh famously stated (in what has come to be known as Krogh's principle) that "for a large number of problems there will be some animal of choice, or a few such animals, on which it can be most conveniently studied" (Krebs, 1975). But there are advantages to employing standardized model organisms. Standardization enables the use of off-the-shelf protocols and analysis tools; it facilitates more efficient scientific communication if the setting is already understood by all parties; and it makes it easier to draw connections between seemingly disparate phenomena. Scientists must balance the pros and cons of creativity and conservativism when selecting a model system for study.

## 1.3 Model Systems of Deep Learning

In the lecture notes of a course on neural network training dynamics, Roger Grosse advises: "the first question to ask about any neural net phenomenon is: does it also happen for linear regression?" (Grosse, 2022). Indeed, linear settings (both regression and classification) are highly productive model systems of deep learning. Experiments are typically easy to run and computationally cheap; the optimal solution can often be expressed in a closed form; the dynamics of optimization algorithms can often be analytically derived; the trained model can be easily interpreted by reading off

the coefficients; and the data distribution can be modified in many ways to model different phenomena.

Lessons from linear settings can transfer to realistic deep learning settings for a few reasons. Firstly, a linear model is, equivalently, a neural network consisting of only a single neuron and no activation function. It is thus a "base case" for analyses of general networks. See, for instance, the influential paper of (Soudry et al., 2018) which proved that a linear classifier* optimized with gradient descent on linearly separable data has an implicit bias towards the maximum margin weights—and has been followed by a series of works generalizing the findings to increasingly broad classes of neural networks (Ji & Telgarsky, 2018; Lyu & Li, 2019; Ji & Telgarsky, 2020; Kunin et al., 2022). At the other extreme, extremely wide neural networks of any depth, with certain (often not best practice) scaling of initialization and learning rates, have training dynamics approaching those of a linear model (Jacot et al., 2018; Du et al., 2018; Allen-Zhu et al., 2019; Zou et al., 2020; Chizat et al., 2019). This "neural tangent kernel" (NTK) insight has enabled explanations for phenomena such as scaling laws (Bordelon et al., 2020, 2024) and spectral bias (Cao et al., 2019; Tancik et al., 2020).†

But while many phenomena of deep learning are already present in linear settings (Belkin et al., 2018), this class of model systems is fundamentally limited. Powerful linear approaches such as kernel and random feature methods (Rahimi & Recht, 2007) are restricted by definition to performing a linear map atop a collection of features that were computed from the input according to a fixed (or random) transformation. They can fit complex functions, but only if their pre-computed features happened to include the appropriate ones. Deep learning approaches, meanwhile, have the capacity to adaptively discover useful features based on patterns in the data. Because of this *feature learning*

*Specifically, unregularized logistic regression—a.k.a. a single neuron without activation function, trained with cross-entropy loss.

†Yet another reason for the transferability of insights from the linear setting, one emphasized by Grosse (2022), is that the loss landscape of linear regression is quadratic, and any smooth loss landscape can be approximated locally by a quadratic function.

or *representation learning*[*] gap, there are various tasks that can be solved by small neural networks, but require an exponential number of features for linear methods (Yehudai & Shamir, 2019; Allen-Zhu & Li, 2019; Malach et al., 2021). This isn't just a difference in capabilities—feature learning comes with richer, fundamentally nonconvex optimization dynamics. It also leads trained networks to be much more interesting *on the inside*, filled with learned hierarchical circuitry (Zeiler & Fergus, 2014; Olah et al., 2020; Clark et al., 2019; Olsson et al., 2022). It is what makes a neural network a sort of computer, not just an expensive curve fitter.

In this dissertation, we go beyond the linear, studying model systems of *feature learning*.

At this point, it's worth clarifying an ambiguity. There are (at least) two sorts of model systems of deep learning, corresponding to different parts of the learning pipeline. On the one hand, there are model architectures; on the other, there are model tasks. (A less confusing, more alliterative name might be "test tube tasks").[†] Model architectures with advantageous scientific productivity properties include multi-layer perceptrons (see Section 6.2 for a definition), deep linear networks (Saxe et al., 2014), simplified transformers (i.e., with only one or two layers, or without MLP modules), variants of the transformer architecture designed with interpretability or control in mind (Hewitt et al., 2023; Friedman et al., 2024), and linear models themselves.

Model systems of *tasks* are particularly useful because everything about the standard deep learning pipeline (the architecture, the training algorithm, the hyperparameters) is precisely mathematically well-defined and controlled by the user *except the data*, which in recent years has tended to look more and more like "everything we can find on the Internet".[‡] The standard analytical methodol-

[*]What do these terms mean? There is no consensus definition; historically (which in the deep learning community means farther back than a few years ago) they were often used synonymously with the term *deep learning* itself (Bengio et al., 2013). The International Conference on Learning Representation (ICLR), for instance, is a premier conference for deep learning research broadly.

[†]Another category is model systems that abstract away the optimization process itself; for instance, see works that exhibit deep learning phenomena in the model system of nearest neighbor methods (Belkin et al., 2019; Nakkiran & Bansal, 2020).

[‡]There are also the nuances of floating-point rounding and hardware, but we will not open those cans of

ogy for dealing with data in theoretical computer science is to assume the data is worst-case*, but the richness of deep learning tends to emerge from the structure of the data, so there is much scientific value in considering what happens when models are trained on particular test tube tasks.

## 1.4 Combinatorial Tasks

Over the past few years, there has been a rising wave of new works using combinatorial tasks as test tube tasks in order to better understand deep learning, including the works which form the basis of this thesis and also (among others): Hupkes et al. (2020); Daniely & Malach (2020); Bhattamishra et al. (2020a); Yao et al. (2021); Zhang et al. (2021); Power et al. (2021); Zhang et al. (2022); Xie et al. (2022); Abbe et al. (2022a); Anil et al. (2022); Liu et al. (2022a); Nanda et al. (2023); Michaud et al. (2023); Chughtai et al. (2023); Bietti et al. (2023); Valvoda et al. (2022); Guo et al. (2023); Glasgow (2023); Zhou et al. (2023); Liu et al. (2024); Sanford et al. (2024); Akyürek et al. (2024). We do not have a formal definition in mind for "combinatorial task"; we are using it as an umbrella term for mathematically well-defined tasks of a discrete, algorithmic, and/or algebraic flavor. There are two major reasons for the increasing prevalence of this style of research. Firstly, on the theoretical side, there is the recognition, beginning around 2019, that many interesting phenomena stem from feature learning, which cannot be studied using linear model systems. Secondly, there is the recent meteoric rise of LLMs, neural networks which have remarkable capacities to fluently converse in human languages, to write computer code, to recall knowledge, and to reason (Devlin et al., 2018; Radford et al., 2019; Brown et al., 2020; Chowdhery et al., 2022; Petroni et al., 2019; Wei et al., 2022). Well-specified combinatorial tasks can serve as targeted models of various aspects of language, enabling the analysis of individual capabilities in isolation.

In this dissertation, we present several case studies in the scientific investigation of deep learning

worms.

*Though see the theory of average-case complexity (Bogdanov et al., 2006) for an alternative perspective.

using combinatorial tasks. The particular tasks studied are:

- In Chapter 4, we study learning a Boolean function which depends on an unknown sparse subset of the input variables. We refer to these as **sparse Boolean functions**, though they are also known as *juntas* in the learning theory literature (Blum & Langley, 1997; Mossel et al., 2003).

- In Chapters 5 and 6, we investigate **sparse parities**, the special case of sparse Boolean functions where the function is the sum modulo 2 of the $k$ relevant bits. A sparse parity function can be equivalently thought of as a monomial in the $\{\pm 1\}$ basis, or as a pure $k$-way interaction among the relevant variables with no lower-order interaction terms.

- In Chapter 7, we consider the task of **performing finite group operations**. Specifically, each input is a pair $(a, b)$, where $a, b$ are elements of a group $G$, and the target output is $ab$. Of particular interest are cyclic groups $\mathbb{Z}_p$, or in other words, **modular addition**. The sparse parity task is again considered in this chapter.

- In Chapter 8, we focus on the task of **learning a Markov chain in-context**. This is a sequence learning task in which each sequence is sampled from a fresh unknown Markov chain.

The investigations in these chapters elucidate a variety of mysterious aspects of deep learning. In the next chapter, we summarize the contributions therein.

# 2

# Contributions

## Chapter 4: Variable Creation

This chapter is based on "Inductive Biases and Variable Creation in Self-Attention Mechanisms" (Edelman et al., 2022), written in collaboration with Surbhi Goel, Sham Kakade, and Cyril Zhang. The starting point is an intuition about transformers (Vaswani et al., 2017): that the self-attention mechanism allows each layer of a transformer to select a sparse subset of its inputs, and the multi-

layer perceptron (MLP) enables performing an arbitrary function on this selection. We call this process "variable creation", whereby transformer layers learn sparse Boolean functions that can then be used as inputs for later layers. It is a form of feature learning, and is meant to be a basic illustration of how transformers with many layers might learn rich circuits. The technical contribution are twofold. First, we prove a first-of-its-kind statistical generalization bound for transformers in terms of the norms of their weights, using the mathematical machinery of covering numbers. Second, we show that a single transformer layer with small weights can express an arbitrary sparse Boolean function. Together, these results imply that empirical risk minimization (exhaustive search for the model that achieves the lowest training error) over bounded-norm transformer layers is sufficient to achieve essentially optimal sample efficiency for learning sparse Boolean functions.

In practice, however, we cannot perform empirical risk minimization and must train using iterative optimization procedures instead. Accordingly, we conclude by performing experimental evaluations of the sample complexity of learning different sparse Boolean functions with transformers trained with SGD. For functions such as sparse conjunctions, the observed sample efficiency is consistent with our prediction. Sparse parities, on the other hand, are expected to pose severe optimization difficulties. A curious empirical finding is that transformers can still learn sparse parities if trained for sufficiently long, exhibiting a long period of trivial performance followed by a sudden drop in error rates at the end of training.

### Chapter 5: Hidden Progress

This chapter is based on "Hidden Progress in Deep Learning: SGD Learns Parities Near the Computational Limit" (Barak et al., 2022), written in collaboration with Boaz Barak, Surbhi Goel, Sham Kakade, Eran Malach, and Cyril Zhang. We begin with the concluding observation from Chapter 4: a transformer trained on the sparse parity function exhibits what appears to be a striking phase tran-

sition (informally speaking) at some point in training, in which there is sudden progress in accuracy out of the blue. This parallels a similar observation in Power et al. (2021) in the setting of modular arithmetic. First, we show that this phase transition phenomenon is not unique to transformers—standard MLPs and a variety of other simple architectures also exhibit the same behavior.

The sparse parity learning problem (with $k$ relevant variables out of $n$ total variables) is notorious in the learning theory community. On the one hand, it is easy in a statistical sense—it can be learned by empirical risk minimization with only $k \log n$ samples. On the other hand, it is hard in a computational sense: it is the paradigmatic example of a problem that is hard for the statistical query learning model (Kearns, 1998; Blum et al., 1994), and its hardness when labels are noisy is a core assumption in modern cryptography (Pietrzak, 2012). Thus, sparse parity learning is an ideal model system for exploring the role of optimization in solving problems that are statistically feasible, but only through exhaustive search.

The main question of this chapter is: if SGD on neural networks is able to learn sparse parities near the computational limits just mentioned (and empirically it indeed can) then what is the optimization process doing? Is it performing some sort of exhaustive search that eventually stumbles upon the correct solution, precipitating the phase transition? Or is there gradual progress towards the solution all along that is hidden when we only look at the loss and accuracy curves? Through several empirical and theoretical lines of evidence, we demonstrate that the latter scenario is actually true: there is hidden progress throughout that seemingly fruitless plateau. We perform an extensive empirical investigation, which includes, among various other findings, the identification of hidden progress measures based on the internal weights of networks which show improvements throughout training. Theoretically, we define a notion of "Fourier gap" based on the machinery of Boolean analysis (O'Donnell, 2014) in order to show that even a single step of SGD on a single neuron, with a sufficiently large batch size, can lead to meaningful progress towards the solution.

## Chapter 6: Pareto Frontiers

This chapter is based on "Pareto Frontiers in Deep Feature Learning: Data, Compute, Width, and Luck" (Edelman et al., 2024a), written in collaboration with Surbhi Goel, Sham Kakade, Eran Malach, and Cyril Zhang. Chapter 5 focused primarily on *online* SGD training, where there is no risk of overfitting to the training set. This paper explores the more complicated setting of *offline* training of neural networks (in particular, MLPs) on the sparse parity task. Here, the tradeoffs are manifold: successful learning can achieved through more training samples, more training iterations, more neurons, or more fresh initializations. The goal of the chapter is to map out the Pareto frontier of successful learning in terms of these resources. Empirically, we present the results of hundreds of thousands of training runs, providing literal plots of the frontier, which reflect nuanced phenomena such as grokking (Power et al., 2021) and sample-wise double descent (Nakkiran et al., 2021). Theoretically, we map the frontier by presenting a multi-resource statistical query lower bound, and a set of upper bounds which approach this lower bound. One notable insight used for the upper bounds is that sparsely initializing the first-layer weights can improve performance. Another is that model width effectively serves as a form of parallelization over simultaneous optimization processes, providing theoretical justification for the lottery ticket hypothesis of Frankle & Carbin (2018). We also perform a preliminary experimental validation of our findings on real-world tabular tasks, demonstrating improved performance for MLPs over prior work, sometimes matching or exceeding the performance of random forest predictors.

## Chapter 7: Feature Emergence

This chapter is based on "Feature Emergence via Margin Maximization: Case Studies in Algebraic Tasks" (Morwani et al., 2023b), written in collaboration with Depen Morwani, Costin-Andrei Oncescu, Rosie Zhao, and Sham Kakade. It is striking that even though natural data sources often

contain symmetries (e.g. translational symmetry in images), general-purpose architectures such as transformers can, with sufficient training data, perform as well as specialized architectures that explicitly enforce these symmetries (Dosovitskiy et al., 2020). Relatedly, a few recent works on algebraic test tube tasks (Nanda et al., 2023; Chughtai et al., 2023) have identified instances where symmetries in the data seem to result in emergent harmonic structure in trained networks. In this chapter, we explain this remarkable empirical finding, in the setting of quadratic-activation MLPs with trained to perform group operations—with a special focus on modular addition. In particular, we prove that all *maximum margin* (with respect to a certain norm) parameter settings display the empirically observed harmonic structure. When the task is modular addition, the incoming weights to each neuron must form a sine wave. For more general groups, these weight vectors correspond to irreducible representations of the group. We introduce a novel duality argument in order to prove these results on the relationship between network structure and network function.

### Chapter 8: Induction Heads

This chapter is based on "The Evolution of Statistical Induction Heads: In-Context Learning Markov Chains" (Edelman et al., 2024b), written in collaboration with Ezra Edelman, Surbhi Goel, Eran Malach, and Nikolaos Tsilivis. Transformer language models are known to be able to learn in-context: when there are patterns in an input sequence, the transformer will tend to continue these patterns in its next-token predictions. Elhage et al. (2021); Olsson et al. (2022) discovered the *induction head*, a learned component of transformer language models that facilitates this in-context learning behavior. In this chapter, we introduce a test tube task that requires only an induction head in order to solve. The task is *in-context learning of Markov chains* (ICL-MC): each sequence in the data distribution is sampled from a fresh Markov chain, which in turn is sampled from a prior distribution over Markov chain transition matrices. In order to perform next-token prediction, a

model trained on this task must infer, in-context, the transition matrix of the Markov chain; in other words, it must learn the 2-gram statistics of the sequence. We also devise extensions of the tasks corresponding to $n$-grams for $n > 2$. Empirically, we find that (attention-only) transformers indeed learn induction heads in order to solve the task. During training, they move through a series of discrete phases: first relying on in-context 1-gram statistics, then 2-gram statistics, and so on. Each phase corresponds to a plateau in the overall loss. Theoretically, we analyze the interplay between the different layers of a simplified transformer-esque architecture as they follow a complex feature learning process during training. Moreover, we provide empirical evidence that the presence of simpler inadequate solutions (e.g. 1-gram statistics in the original task) slows down the learning of the eventual correct solution, an instance of a pitfall of simplicity bias (Shah et al., 2020).

# 3

# Deep Learning Preliminaries

This chapter is a primer on fundamental deep learning notions which will be employed in the dissertation.[†]

We will primarily focus on the classic machine learning setting of supervised classification, in which there is an input space $\mathcal{X}$, a finite output (or label) space $\mathcal{Y}$, and a target distribution $\mathcal{D}$ over

[†]For the sake of novelty and pedantry, some definitions are presented in an idiosyncratic way. Specifically, the definition of neural networks in terms of circuits in Section 3.1, and the notation introduced in Section 3.2 for pre- and post-processing real-valued neural network inputs and outputs for discrete classification.

$\mathcal{X} \times \mathcal{Y}$. A classification algorithm takes as input a training set $S$ of samples $(x, y)$ drawn from $\mathcal{D}$, and produces a classifier $h : \mathcal{X} \to \mathcal{Y}$.* The goal is to produce a classifier that accurately predicts the labels of new inputs from the population.

Population accuracy measures the proportion of samples from the distribution for which the hypothesis is correct:

$$\Pr_{(x,y)\sim\mathcal{D}}[h(x) = y].$$

and population error is the proportion of samples for which the hypothesis is incorrect. These quantities can be estimated by measuring empirical performance on a fresh sample from $\mathcal{D}$ referred to as a *test* or *validation* set, yielding test (or validation) error/accuracy.

## 3.1 Neural Networks

In deep learning, an *architecture* is a circuit that implements a multivariate function over the reals of the form

$$f : \mathbb{R}^{d_{\text{in}}} \times \mathbb{R}^{d_{\text{params}}} \to \mathbb{R}^{d_{\text{out}}}$$

Its input consists of two parts: an encoding of the input $x$, and a parameter vector $\theta$. The elements of $\theta$ are referred to as *parameters* or *weights*. A *neural network* (also referred to as simply a *network* or *model*) is a circuit obtained by fixing the weights to a particular constant setting $\theta$ (thus computing the function $f(\cdot; \theta)$).

We are using the term 'circuit' in the sense it is used in computational complexity theory (Arora & Barak, 2009). A circuit is a directed acyclic† graph in which each node (gate) corresponds to an

*In much of this thesis, we will focus on the case where all the randomness in $D$ is over $\mathcal{X}$—i.e., $y$ is a deterministic function of $x$.

†Recurrent neural networks (RNNs) are often described as having cyclic information flow. However, during training RNNs are typically unrolled into acyclic networks. In any case, we will focus on *feedforward* neural networks, which are by definition acyclic.

intermediate function. The input gates (those with in-degree zero) are labeled with input variables or constants. The remaining gates are labeled with elementary operations from some set of allowed operations; a node with in-degree $k$ must be labeled with a $k$-ary operation. The output of the gate is obtained by applying the operation to its inputs; in this way, some intermediate function is inductively computed at every gate. Finally, $d_{\text{out}}$ of the gates are labeled as output gates, and we say that the circuit computes the function corresponding to the tuple of the functions computed at the output gates. What elementary operation are to be allowed? One basis set of operations which suffices for virtually all neural networks commonly studied and used is: addition, multiplication, and arbitrary unary operations.[*] In other words, neural networks can typically be expressed as arithmetic circuits (Shpilka et al., 2010) augmented with unary gates.[†‡]

In practice, the success of deep learning has been enabled by massively parallel implementations of neural networks, permitting the serial runtime of the circuit to be drastically lower than the total number of operations. It is standard to organize arithmetic operations on individual scalars into linear algebraic operations on massive tensors; and to apply single unary operations in bulk to all the entries of a tensor. Thus, architectures are typically described using the language of linear algebra and function composition; the above formulation in terms of arithmetic circuits is an attempt towards generality.

[*] Allowing *arbitrary* unary gates is not restrictive enough to be reasonable. Given a countable class of functions $\mathcal{F}$, there exists a unary gate that can be incorporated into a "universal" network with only a single parameter that nevertheless is able to simulate arbitrary functions in $\mathcal{F}$ with different settings of the parameter (exercise for the reader). It is not obvious what the "right" restricted class of unary gates should be.

[†] The definition of neural network presented here is nonstandard, but there is no consensus definition. See the notion of "tensor programs" introduced in Yang (2019) for another attempt at a general characterization. Yang's definitions, which deal directly with vectors rather than scalars, have the desirable property that there is a well-defined notion of scaling the width of a fixed architecture.

[‡] Baur & Strassen (1983) proved that the gradient of an arithmetic circuit's output with respect to its input can be computed in a number of arithmetic operations proportional to the number of gates in the circuit. This fact is essentially equivalent to the fact that the gradient of a neural network's output with respect to its parameters can be efficiently computed for neural networks through backpropagation, which motivates the use of first-order optimization algorithms (see Section 3.3) in deep learning.

One classic architecture is the *multi-layer perceptron* (MLP), also known as the *fully-connected network*. The term "perceptron" comes from the pioneering work of Rosenblatt (1958). Roseblatt's perceptron was originally conceived as a model system of neural processing—a "hypothetical nervous system ...designed to illustrate some of the fundamental properties of intelligent systems in general, without becoming too deeply enmeshed in the special, and frequently unknown, conditions which hold for particular biological organisms." It is the forerunner of the multi-layer perceptron studied in this dissertation, and differs in only a few details.

As an illustrative example of MLPs and neural networks in general, we will define a two-layer MLP (a.k.a. one-hidden layer MLP) with scalar output. The parameters of this architecture are a matrix $W \in \mathbb{R}^{d_{\text{hidden}}} \times \mathbb{R}^{d_{\text{in}}}$ and a vector $u \in \mathbb{R}^{d_{\text{hidden}}}$, for some *hidden dimension* or *width* $d_{\text{hidden}}$. The full parameter vector $\theta$ is the concatenation of $u$ with a flattened version of $W$. Hence, we have $d_{\text{params}} = d_{\text{hidden}} + d_{\text{hidden}} d_{\text{in}}$. The architecture is:

$$f_{\text{MLP}}(x; (u, W)) := u^\top \sigma(Wx)$$

Here, $\sigma$ is an *activation function*. It is defined as a univariate function over the reals; when, in the architecture definition, we give it a vector input, we are abusing notation to indicate that it is applied coordinate-wise—i.e.,

$$\sigma(Wx) := \begin{bmatrix} \sigma({W_1}^\top x) & \dots & \sigma({W_{d_{\text{hidden}}}}^\top x) \end{bmatrix}.$$

Note that without the activation function, a MLP network would be a linear map. By choosing a nonlinear function for $\sigma$, a far richer class of functions can be represented by the architecture over different settings of the weights. One common activation function is the ReLU (rectified linear

unit):

$$\sigma(z) = \max(z, 0).$$

The functions $x \mapsto \sigma((Wx)_i)$ are called *neurons*, by analogy to biological neurons. In the analogy, elements of $x$ are input stimuli which are propagated to neuron $i$ by synapses with strengths $W_i$; the neuron then "fires" with a "firing rate" (here the analogy is strained) determined by its response function $\sigma$. In deep learning parlance, the value of $\sigma((Wx)_i)$ is referred to as the *activation* of neuron $i$; and our MLP has a "hidden layer" consisting of $d_{\text{hidden}}$ neurons. Note that our general definition of a neural network does not require that it is decomposable into neurons, and indeed some architectures, such as the transformer (Vaswani et al., 2017), include components that don't fit neatly into the "neurons and synapses" analogy.

## 3.2 Classifiers from Networks

We now describe how a network which operates over the reals induces a discrete classifier. Given a network with parameter vector $\theta$, define the corresponding classifier as

$$h(x) := \mathsf{discretize}(f(\mathsf{encode}(x); \theta))$$

where $\mathsf{encode} : \mathsf{X} \to \mathbb{R}^{d_{\text{in}}}$ is an optional helper function that embeds inputs into the proper format, and $\mathsf{discretize} : \mathbb{R}^{d_{\text{out}}} \to \mathcal{Y}$ converts the real-valued output of the model into a discrete label. In this dissertation, we focus on cases where $\mathcal{X} = S^d$ for some finite set $S$ and input size $d$. When $|S| = 2$, we encode the two elements of $S$ as scalars $\pm 1$. When $|S| > 2$, we encode each

element $i$ of $S$ as an indicator vector ("one-hot encoding"):

$$e_j := \begin{cases} 1 & \text{if } j = i \\ 0 & \text{otherwise} \end{cases}.$$

Finally, encode encodes each coordinate of $x$ accordingly and concatenates the results. Typically, for notational simplicity, we will elide encode and assume the inputs are already prepared in the proper real-valued format.

Meanwhile, the method of discretizing outputs also (by convention) depends on the cardinality of the space. If $|\mathcal{Y}| = 2$, we let $d_{\text{out}} = 1$, and use $\mathsf{discretize}(f(x;\theta)) := \text{sign}(f(x;\theta))$. If $|\mathcal{Y}| > 2$, we let $d_{\text{out}} = |\mathcal{Y}|$ and use $\mathsf{discretize}(f(x;\theta)) := \arg\max_{i \in \mathcal{Y}} f(x;\theta)_i$.

## 3.3 Training

Neural networks are typically trained with first-order iterative optimization algorithms. By *iterative* we mean that the learning algorithm begins with some (typically random) initial setting for the parameters $\theta$ (an *initialization*) and then incrementally improves $\theta$ in a sequence of updates. By *first-order* we mean that the update rule employs first-order derivatives of some *loss function* $\ell : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ with respect to the parameters.

The loss function is meant to be a proxy for population error. Recall that the goal is to minimize population error:

$$\mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}[\mathbb{1}[h(x) \neq y]].$$

The issue with using first-order optimization algorithms to minimize this directly, without a loss function, is that the discretization of the network's outputs renders population error insensitive to

infinitesimal changes in the parameters. So instead we try to minimize the *population loss*

$$\mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}[\ell(y,f(x;\theta))]$$

for a judiciouly chosen $\ell$. Examples of loss functions for binary classification (with $\mathcal{Y} = \{\pm 1\}$, but considered as a subset of $\mathbb{R}$) include:

- Cross-entropy (logistic) loss: $\ell(y,f(x;\theta)) = -\log \frac{\exp(yf(x;\theta))}{1+\exp(yf(x;\theta))}$
- Square loss: $\ell(y,f(x;\theta)) = (y - f(x;\theta))^2$
- Hinge loss: $\ell(y,f(x;\theta)) = \max(0, 1 - yf(x;\theta))$
- Correlation loss: $\ell(y,f(x;\theta)) = yf(x;\theta)$

The prototypical first-order iterative optimization algorithm is *gradient descent* (GD). To run GD, sample the initial network weights $\theta_0$ from some initialization distribution. Then, for $t = 0, \ldots$ until some stopping condition is reached, perform the following update:

$$\theta_{t+1} = \theta_t - \eta_t \cdot \nabla_\theta \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}[\ell(y,f(x;\theta))].$$

where $\{\eta_t\}_{t=1}^T$ is some learning rate (a.k.a. step size) schedule. The *hyperparameters* (tunable aspects of the algorithm specification) of vanilla gradient descent are the initialization distribution and the learning rate schedule. The motivation for the algorithm is that we are iteratively adjusting the weights in the direction of steepest descent of the loss.

In practice, it is more common to use *stochastic* gradient descent (SGD), which only requires access to the gradient of the loss on a small sample of data points, rather than the entire population:

$$\theta_{t+1} = \theta_t - \eta_t \cdot \nabla_\theta \left( \frac{1}{B} \sum_{i=1}^{B} \ell(y_{t,i}, f(x_{t,i};\theta_t)) \right).$$

where $B$ is some *batch size* (another hyperparameter), and each $\{(x_{t,i}, y_{t,i})\}_{i=1}^{B}$ is a *batch* (a.k.a. minibatch) of training examples.

In *online* SGD, the elements of each batch are sampled i.i.d. from the full population $\mathcal{D}$. Over the randomness of sampling, the expectation of each minibatch gradient is the population gradient. Equivalently, we can conceptualize online SGD as iterating over consecutive chunks of $b$ examples from a fixed training set $S$ (drawn i.i.d. from $\mathcal{D}$) in a single pass (or *epoch*). In *offline* SGD, we perform *multiple* passes over $S$, typically shuffling $S$ after each epoch. In *offline gradient descent*, every batch is the full training set.

There are many other first-order iterative optimizers which address various shortcomings of the simple (yet still powerful) SGD algorithm. In this dissertation, when we are dealing with the transformer architecture, we perform training with the Adam optimizer (Kingma & Ba, 2014), which is currently popular for transformer training.

With any of these algorithms, there is the option of adding a function of the weights, called a *regularization term*, to the empirical average of the loss function in the update rule. A common regularization term is $\|\theta\|$, for some norm $\|\cdot\|$. The goal of regularization is typically to improve *generalization* (performance on the full population, in the offline training setting) by reducing the capacity of the learning algorithm to overfit to spurious patterns in the training sample. Note that the solutions found by offline first-order optimization algorithms are often *implicitly* biased (Vardi, 2023) towards particular parts of weight space, leading to favorable generalization properties even without explicit regularization.

*This chapter is based on "Inductive Biases and Variable Creation in Self-Attention Mechanisms" (Edelman et al., 2022), written in collaboration with Surbhi Goel, Sham Kakade, and Cyril Zhang.*

# 4

# Variable Creation

Self-attention, an architectural motif designed to model long-range interactions in sequential data, has driven numerous recent breakthroughs in natural language processing and beyond. This work provides a theoretical analysis of the inductive biases of self-attention modules. Our focus is to rigorously establish which functions and long-range dependencies self-attention blocks prefer to represent. Our main result shows that bounded-norm Transformer networks "create sparse variables": a single self-attention head can represent a sparse function of the input sequence, with sample com-

plexity scaling only logarithmically with the context length. To support our analysis, we present synthetic experiments to probe the sample complexity of learning sparse Boolean functions with Transformers.

## 4.1 Introduction

Self-attention mechanisms have comprised an era-defining cornerstone of deep learning in recent years, appearing ubiquitously in empirical breakthroughs in generative sequence modeling and unsupervised representation learning. Starting with natural language (Vaswani et al., 2017), self-attention has enjoyed surprising empirical successes in numerous and diverse modalities of data. In many of these settings, self-attention has supplanted traditional recurrent and convolutional architectures, which are understood to incorporate inductive biases about temporal and translational invariances in the data. Self-attention models discard these functional forms, in favor of directly and globally modeling long-range interactions within the input context.

The proliferation of self-attention raises a fundamental question about its inductive biases: *which functions do self-attention networks prefer to represent?* Various intuitions and empirics inform the design of these architectures, but formal statistical abstractions and analyses are missing in this space. To this end, this work initiates an analysis of the statistical foundations of self-attention.

We identify an inductive bias for self-attention, for which we coin the term *sparse variable creation*: a bounded-norm self-attention head learns a sparse function (which only depends on a small subset of input coordinates, such as a constant-fan-in gate in a Boolean circuit) of a length-$T$ context, with sample complexity scaling as $\log(T)$. The main technical novelty in this work is a covering number-based capacity bound for attention mechanisms (including Transformer heads, as well as related and future architectures), implying norm-based generalization bounds. This is accompanied by a matching representational result, showing that bounded-norm self-attention heads are indeed

capable of representing $s$-sparse functions with weight norms $2^{O(s)}$ (or $\text{poly}(s)$, for symmetric sparse functions). This provides a theoretical account for why attention models can learn long-range dependencies without overfitting.

Finally, we conduct synthetic experiments to probe the sample efficiency of learning sparse interactions with self-attention. We train Transformer models to identify sparse Boolean functions with randomly chosen indices, and corroborate the sample complexity scaling law predicted by the theory. A variant of this experiment (with i.i.d. samples) reveals a *computational* mystery, beyond the scope of our current statistical analysis: we find that Transformers can successfully learn the "hardest" (in the sense of SQ-dimension) $s$-sparse functions: the XOR (parity) functions.

### 4.1.1 Related work

The direct precursors to modern self-attention architectures were recurrent and convolutional networks augmented with attention mechanisms (Bahdanau et al., 2014; Luong et al., 2015; Xu et al., 2015). Landmark work by Vaswani et al. (2017) demonstrated significantly improvements in machine translation via a pure self-attention architecture; autoregressive language models (Liu et al., 2018; Radford et al., 2018, 2019; Brown et al., 2020), and self-supervised representation learning via masked language modeling (Devlin et al., 2018) followed shortly.

Norm-based capacity bounds for neural nets. There is a vast body of literature dedicated to establishing statistical guarantees for neural networks, including VC-dimension and shattering bounds (dating back to Anthony & Bartlett (1999)). In recent years, classical norm-based generalization bounds have been established for various architectures (Bartlett et al., 2017; Neyshabur et al., 2015, 2017; Golowich et al., 2018; Long & Sedghi, 2019; Chen et al., 2019) using covering-based arguments. Jiang et al. (2019) provide an extensive empirical study of how well these bounds predict generalization in practice. Our work complements these results by establishing the first

norm-based capacity analysis for attention models. Our main results rely on a novel reduction to the $\ell_\infty$ covering number bound for linear function classes given by Zhang (2002).

Other theoretical lenses on attention. Our work complements various existing theoretical perspectives on attention-based models. Vuckovic et al. (2020) formulate a dynamical system abstraction of attention layers, arriving at similar Lipschitz constant calculations to ours (which are coarser-grained, since they focus on contractivity and stability rather than finite-sample statistical guarantees). Zhang et al. (2019); Snell et al. (2021) study idealizations of the optimization problem of learning self-attention heads. Wei et al. (2021) propose a definition of *statistically meaningful approximation* of function classes that ties statistical learnability with expressivity, and show that Boolean circuits can be *SM-approximated* by Transformers with a sample complexity bound that depends mildly on circuit depth (rather than context size), using a margin amplification procedure. Kim et al. (2021) show that standard dot-product attention is not Lipschitz for an *unbounded* input domain, whereas our paper shows that norm-based generalization bounds are attainable with a $\|\cdot\|_{2,\infty}$-bounded input domain.

See Appendix A.4 for a broader survey of the literature on attention and self-attention networks.

## 4.2 Background and notation

Throughout this paper, the input $X := [x_1 x_2 \dots x_T]^\top \in \mathbb{R}^{T\times d}$ to an attention module (a.k.a. the context) will be a length-$T$ sequence of embeddings $x_t \in \mathbb{R}^d$; $m$ refers to the sample size (i.e. number of length-$T$ sequences in a dataset). $\|\cdot\|_2$ denotes the spectral norm for matrices, and $\|\cdot\|_{p,q}$ denotes the $(p,q)$ matrix norm where the $p$-norm is over columns and $q$-norm over rows. For vectors, $\|\cdot\|_p$ denotes the $\ell_p$ norm; we drop the subscript for the $\ell_2$ norm. $B$ is generally used to quantify bounds on norms of matrices and $L$ for Lipschitz constants. $\Delta^{n-1}$ denotes the simplex in dimension $n$, that is, $\Delta^{n-1} := \{x \in \mathbb{R}^n : x \geq 0, \|x\|_1 = 1\}$.

Covering numbers. Our main technical contribution is a generalization bound arising from carefully counting the number of functions representable by a Transformer. The main technical ingredient is the notion of a covering number. We will use the following definition of $\infty$-norm covering number adapted from Zhang (2002):

**Definition 4.1** (Covering number)**.** For a given class of vector-valued functions $\mathcal{F}$, the covering number $\mathcal{N}_\infty(\mathcal{F}; \varepsilon; \{z^{(i)}\}_{i=1}^m; \|\cdot\|)$ is the smallest size of a collection (a cover) $\mathcal{C} \subset \mathcal{F}$ such that $\forall f \in \mathcal{F}, \exists \widehat{f} \in \mathcal{C}$ satisfying

$$\max_i \|f(z^{(i)}) - \widehat{f}(z^{(i)})\| \leq \varepsilon.$$

Further, define

$$\mathcal{N}_\infty(\mathcal{F}, \varepsilon, m, \|\cdot\|) = \sup_{z^{(1)} \ldots z^{(m)}} \mathcal{N}_\infty(\mathcal{F}; \varepsilon; z^{(1)}, \ldots, z^{(m)}, \|\cdot\|).$$

If $\mathcal{F}$ is real-valued (instead of vector-valued), we drop the norm from the notation. Furthermore for functions parameterized by a set of parameters $\Theta$, we exploit the notation to replace $\mathcal{F}$ by $\Theta$.

Recall that for the class of linear functions,

$$\mathcal{F}_{\text{lin}} = \{x \mapsto w \cdot x \,:\, w \in \mathbb{R}^d, \|w\|_2 \leq B_W\},$$

we have the covering number bound (Zhang, 2002) of

$$\mathcal{N}_\infty(\mathcal{F}; \varepsilon; \{x^{(i)}\}_{i=1}^m) \leq O\left(\frac{B_X^2 B_W^2}{\varepsilon^2} \cdot \log\left(\frac{B_X B_W m}{\varepsilon}\right)\right),$$

where $\|x^{(i)}\| \leq B_X$ for $i \in [m]$. Importantly, note that the covering number has a mild dependence on $m$, only logarithmic; this logarithmic dependence on $m$ will be helpful when we turn our analysis to the capacity of attention mechanisms.

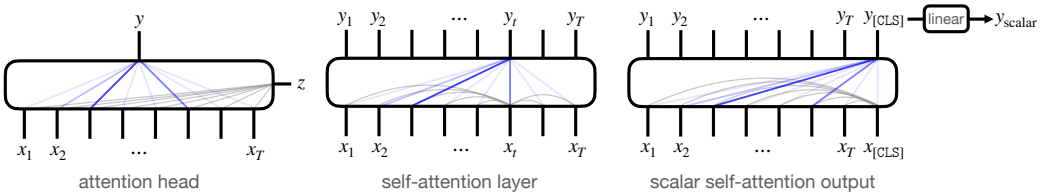



**Figure 4.1:** Diagrams of attention modules $f_{\text{tf-head}}, f_{\text{tf-layer}}, f_{\text{tf-scalar}}$ described in Section 4.3: alignment scores (grey edges) determine normalized attention weights (blue), which are used to mix the inputs $x_{1:T}$. *Left:* Attention with a general context $z$. *Center:* Self-attention layer, where both the input and the context come from $x_{1:T}$. *Right:* Auxiliary `[CLS]` token to extract a single scalar from a self-attention layer, providing a real-valued function class for classification or regression tasks.

GENERALIZATION BOUNDS. This work focuses on providing log-covering number bounds, which imply uniform generalization bounds via standard arguments. The following lemma relates these quantities; we refer the reader to Appendix A.1.1 for a formal review.

**Lemma 4.2** (Generalization bound via covering number; informal)**.** *Suppose $\mathcal{F}$ is a class of bounded functions, and $\log \mathcal{N}_\infty(\mathcal{F}; \varepsilon; x^{(1)}, \ldots, x^{(m)}) \le C_{\mathcal{F}}/\varepsilon^2$ for all $x^{(1)}, \ldots, x^{(m)} \in \mathcal{X}^m$. Then for any $\delta > 0$, with probability at least $1 - \delta$, simultaneously for all $f \in \mathcal{F}$, the generalization error $\varepsilon_{\text{gen}}$ satisfies*

$$\varepsilon_{\text{gen}}(f) \le \tilde{O}\left(\sqrt{\frac{C_{\mathcal{F}}}{m}} + \sqrt{\frac{\log(1/\delta)}{m}}\right).$$

## 4.3 ABSTRACTIONS OF (SELF-)ATTENTION

The precise definition of attention is less straightforward to define than for architectural components such as convolutions and residual connections. In this section, guided by the manifestations of attention discussed in (Luong et al., 2015), we present some notation and definitions which generalize attention mechanisms commonly seen in practice, including the Transformer. Intuitively, these definitions encompass neural network layers which induce context-dependent representation bottlenecks. Subsequently, we show how to represent the Transformer (the predominant attention-

based architecture) as a special case of this formulation.

### 4.3.1 Attention

Intuitively, we would like to capture the notion that an output variable selects ("attends to") a part of the input sequence on which it will depend, based on a learned function of global interactions (see Figure 4.1, left). To this end, we define an *attention head* as a function which maps a length-$T$ input sequence (e.g. the tokens in a sentence, pixels in an image, or intermediate activations in a deep Transformer network) and an additional context $z \in \mathcal{Z}$ to an output $y \in \mathcal{Y}$. In this work, we will exclusively consider $\mathcal{X}, \mathcal{Y}, \mathcal{Z}$ to be $\mathbb{R}^d$. An attention head uses $z$ to select the input coordinates in $X$ to which the output $y$ will "attend", formalized below:

**Definition 4.3** (Attention head)**.** An attention head is a function $f : \mathcal{X} \to \mathcal{Y}$, specified by an alignment score function $\mathsf{Score} : \mathcal{X} \times \mathcal{Z} \to \mathbb{R}$ parameterized by $\theta_s \in \Theta_s$, normalization function $\mathsf{Norm} : \mathbb{R}^T \to \Delta^{T-1}$, and position-wise maps $\varphi_{\text{in}} : \mathcal{X} \to \mathcal{V}, \varphi_{\text{out}} : \mathcal{V} \to \mathcal{Y}$ parameterized by $\theta_{\text{in}} \in \Theta_{\text{in}}$ and $\theta_{\text{out}} \in \Theta_{\text{out}}$. The output of an attention head on input $X \in \mathcal{X}^T, z \in \mathcal{Z}$ is

$$
\begin{aligned}
y &= \varphi_{\text{out}}\Big( \sum_{t=1}^{T} \Big[ \mathsf{Norm}\Big( \mathsf{Score}(x_1, z; \theta_s), \ldots, \mathsf{Score}(x_T, z; \theta_s) \Big) \Big]_t \varphi_{\text{in}}(x_t; \theta_{\text{in}});\ \theta_{\text{out}} \Big) \\
&= \varphi_{\text{out}}\Big( \varphi_{\text{in}}(X; \theta_{\text{in}})^\top \mathsf{Norm}\Big( \mathsf{Score}(x_1, z; \theta_s), \ldots, \mathsf{Score}(x_T, z; \theta_s) \Big);\ \theta_{\text{out}} \Big)
\end{aligned}
$$

where $\varphi_{\text{in}}(X; \theta) = [\varphi_{\text{in}}(x_1; \theta) \ldots \varphi_{\text{in}}(x_T; \theta)]^\top$ denotes the row-wise application of $\varphi_{\text{in}}$.

The above definition corresponds to the leftmost diagram in Figure 4.1. Here, $\mathcal{V}$ is a vector space of input representations "mixed" by the normalized alignment scores; in this work, we will set $\mathcal{V} = \mathbb{R}^k$. A function class of attention heads is induced by specifying parameter classes for $\{\Theta_s, \Theta_{\text{in}}, \Theta_{\text{out}}\}$.

### 4.3.2 Self-attention and Transformers

A *self-attention head* is a special case of an attention head, in which the context $z$ comes from one of the inputs $x_t$ themselves: pairwise interactions among the elements in $X$ are used to select the elements of $X$ on which $f$ depends. In this case, we use "input" and "context" interchangeably to refer to $X$. For example, a self-attention head which uses $z := x_t$ is defined by

$$y = \varphi_{\text{out}}\left(\varphi_{\text{in}}(X;\theta_{\text{in}})^\top \mathsf{Norm}(\mathsf{Score}(X, x_t;\theta_s));\theta_{\text{out}}\right).$$

We now define the Transformer self-attention architecture as a special case of the above. Since a Transformer layer has shared parameters between multiple output heads, we will define all $T$ outputs of the layer at once.

**Definition 4.4** (Transformer layer)**.** A Transformer layer is a collection of $T$ attention heads (whose outputs are $y_1, \dots, y_T$) with the following shared parameters:

- The context for head $\tau$ is $x_\tau$, and the alignment score function is quadratic:

$$\mathsf{Score}(x, x_\tau; \{W_Q, W_K\}) := x_\tau^\top W_Q W_K^\top x, \quad W_Q, W_K \in \mathbb{R}^{d\times k}.$$

- $\varphi_{\text{in}}$ is linear:

$$\varphi_{\text{in}}(x; W_V) := W_V^\top x, \quad W_V \in \mathbb{R}^{d\times k}.$$

- $\varphi_{\text{out}}$ is a linear function, composed with an $L_\sigma$-Lipschitz activation function $\sigma : \mathbb{R} \to \mathbb{R}$ such that $\sigma(0) = 0$:

$$\varphi_{\text{out}}(x; W_C) := W_C^\top \sigma(x), \quad W_C \in \mathbb{R}^{k\times d}.$$

- The normalization function is the $T$-dimensional softmax:

$$\mathsf{Norm}(x) := \mathsf{softmax}\,(x) = \frac{\exp\,(x)}{1^\top \exp\,(x)}.$$

Defining $Y := [y_1 y_2 \ldots y_T]^\top \in \mathbb{R}^{T\times d}$ and $[\mathsf{RowSoftmax}(M)]_{t,:} := \mathsf{softmax}(M_{t,:})$, we have

$$Y = \sigma\left(\mathsf{RowSoftmax}\left(XW_Q(XW_K)^\top\right)XW_V\right)W_C.$$

Functions from the above class of Transformer layers map $\mathbb{R}^{T\times d}$ to itself, and can thus be iteratively composed. We discuss remaining discrepancies between Definition 4.4 and real Transformers (positional embeddings, position-wise feedforward networks, layer normalization, parallel heads, residual connections) in Section 4.4.3 and the appendix.

Extracting scalar outputs from a Transformer. Finally, we establish notation for a canonical way to extract a scalar prediction from the final layer of a Transformer. For a context of size $T$, a Transformer layer with $T+1$ inputs is constructed, with a special index `[CLS]`.* The input at this position is a vector $x_{\texttt{[CLS]}} \in \mathbb{R}^d$ (which can be fixed or trainable); the output is a linear function $w^\top y_{\texttt{[CLS]}}$, for a trainable parameter $w \in \mathbb{R}^d$. This defines a class of functions mapping $\mathbb{R}^{T\times d} \to \mathbb{R}$, parameterized by a Transformer layer's parameters and $w$, which we call the class of *scalar-output Transformers*. This is the setup used by the classification modules in BERT (Devlin et al., 2018) and all of its derivatives.

*`[CLS]` stands for "class", as in "treat the output at this position as the classifier's prediction".

## 4.4 Capacity bounds for attention modules

In this section, we present covering number-based capacity bounds for generic attention heads and Transformers, along with overviews of their proofs. Section 4.4.1 bounds the capacity of a general attention head. Section 4.4.2 instantiates this bound for the case of a single Transformer self-attention head. Section 4.4.3 generalizes this bound for full depth-$L$ Transformer networks. The sample complexity guarantees for Transformers scale only *logarithmically* in the context length $T$, providing rigorous grounding for the intuition that the architecture's inductive bias selects sparse functions of the context.

Note: Throughout this section, assume that $\|x_t\|_2 \leq B_X$ for all $t \in [T]$ (i.e. $\|X\|_{2,\infty} \leq B_X$). Note that this allows for the Frobenius norm $\|X\|_F$ to scale with $\sqrt{T}$. The key challenge throughout our analysis is to avoid incurring factors of norms which take a sum over the $t$ dimension, by constructing covers appropriately.

### 4.4.1 Capacity of a general attention head

Recall that the attention head architecture can be represented as a function $f_{\mathsf{head}} : \mathbb{R}^{T\times d} \times \mathbb{R}^d \to \mathbb{R}^d$ parameterized by $\theta_s, \theta_{\text{in}}, \theta_{\text{out}}$ as

$$f_{\mathsf{head}}(X, z; \theta_s, \theta_{\text{in}}, \theta_{\text{out}}) = \varphi_{\text{out}}\left(\varphi_{\text{in}}(X; \theta_{\text{in}})^\top \mathsf{Norm}(\mathsf{Score}(X, z; \theta_s)); \theta_{\text{out}}\right).$$

Denote the corresponding function class by

$$\mathcal{F}_{\mathsf{head}} := \{(X, z) \mapsto f_{\mathsf{head}}(X, z; \theta_s, \theta_{\text{in}}, \theta_{\text{out}}) : \theta_s \in \Theta_s, \theta_{\text{in}} \in \Theta_{\text{in}}, \theta_{\text{out}} \in \Theta_{\text{out}}\}$$

To convert the vector-valued function class to a scalar output function class, we define $\mathcal{F}_{\mathsf{scalar}} :=$

$\{(X, z) \mapsto w^\top f(X, z) : f \in \mathcal{F}_{\mathsf{head}}, w \in \mathbb{R}^d, \|w\| \le B_w\}$.

For simplicity of presentation, we will focus only on the attention head, and assume that $\varphi_{\text{out}}$ and $w$ are fixed. We handle the general case of trainable downstream layers in the analysis of multi-layer Transformers in Appendix A.1.7.

**Assumption 4.5.** We make the following assumptions:

1. $\varphi_{\text{out}}$ is $L_{\text{out}}$-Lipschitz in the $\ell_2$-norm, that is,

$$\forall a, b \in \mathbb{R}^k : \|\varphi_{\text{out}}(a) - \varphi_{\text{out}}(b)\| \le L_{\text{out}}\|a - b\|.$$

2. $\varphi_{\text{in}}$ is $B_{\text{in}}$-bounded in $\ell_2$-norm, that is,

$$\forall a \in \mathbb{R}^d, \theta_{\text{in}} \in \Theta_{\text{in}} : \|\varphi_{\text{in}}(a; \theta_{\text{in}})\| \le B_{\text{in}}\|a\|.$$

3. $\mathsf{Norm}$ is continuously differentiable and its Jacobian satisfies

$$\forall \theta \in \mathbb{R}^T, \|J\,\mathsf{Norm}(\theta)\|_{1,1} \le C_{\mathsf{Norm}}.$$

Note that $\mathsf{softmax}$ (the most commonly used $\mathsf{Norm}$ function) satisfies the Jacobian assumption with $C_{\mathsf{softmax}} = 2$ (see Corollary A.7).

We prove the following bound on the covering number of $\mathcal{F}_{\mathsf{head}}$ for $m$ samples,

**Theorem 4.6** (Attention head capacity). *Under Assumptions 4.5, $\forall \alpha \in [0, 1]$ the covering number of $\mathcal{F}_{\mathsf{head}}$ satisfies*

$$\log \mathcal{N}_\infty \left(\mathcal{F}_{\mathsf{head}}; \varepsilon; \left\{(X^{(i)}, z^{(i)})\right\}_{i=1}^m; \|\cdot\|_2\right) \le$$
$$\log \mathcal{N}_\infty \left(\mathcal{F}_{\mathsf{Score}}; \frac{\alpha\varepsilon}{C_{\mathsf{Norm}} L_{\text{out}} B_{\text{in}} B_X}; \{(x_t^{(i)}, z^{(i)})\}_{t\in[T]}^{i\in[m]}\right) + \log \mathcal{N}_\infty \left(\mathcal{F}_{\text{in}}; \frac{(1-\alpha)\varepsilon}{L_{\text{out}}}; \{x_t^{(i)}\}_{t\in[T]}^{i\in[m]}; \|\cdot\|_2\right),$$

*where* $\mathcal{F}_{\mathsf{Score}} = \{(x,z) \mapsto \mathsf{Score}(x,z;\theta_s) : \theta_s \in \Theta_s\}$, *and* $\mathcal{F}_{\text{in}} = \{x \mapsto \varphi_{\text{in}}(x;\theta_{\text{in}}) : \theta_{\text{in}} \in \Theta_{\text{in}}\}$.

Note that the bound is in terms of the $\mathcal{N}_\infty$ covering number of functions that dependent on dimensions $d$ or $k$ and not $T$. The effect of $T$ only shows up in the number of samples to cover. Crucially, for some function classes (e.g. linear functions (Zhang, 2002)), $\mathcal{N}_\infty$ scales only logarithmically with the number of samples. This is exactly what allows us to obtain our log $T$ capacity bounds.

Since $w$ is fixed, an $\varepsilon$-covering of $\mathcal{F}_{\mathsf{head}}$ directly gives us an $\varepsilon B_w$-covering for $\mathcal{F}_{\mathsf{scalar}}$, implying

$$\log \mathcal{N}_\infty\left(\mathcal{F}_{\mathsf{scalar}}; \varepsilon; \left\{(X^{(i)}, z^{(i)})\right\}_{i=1}^m\right) \le \log \mathcal{N}_\infty\left(\mathcal{F}_{\mathsf{head}}; \varepsilon/B_w; \left\{(X^{(i)}, z^{(i)})\right\}_{i=1}^m, \|\cdot\|_2\right).$$

Proof overview. In order to prove the bound, we first show a Lipschitzness property of $f_{\mathsf{head}}$. This property allows us to construct a cover by using covers for $\mathcal{F}_{\mathsf{Score}}$ and $\mathcal{F}_{\text{in}}$.

**Lemma 4.7** ($\ell_\infty$-Lipschitzness of $f_{\mathsf{head}}$)**.** *For any* $\theta_s, \widehat{\theta}_s \in \Theta_s, \theta_{\text{in}}, \widehat{\theta}_{\text{in}} \in \Theta_{\text{in}}$*; for all* $X \in \mathbb{R}^{T\times d}$*, such that* $\left\|X^\top\right\|_{2,\infty} \le B_X$*,*

$$\left\|f_{\mathsf{head}}(X,z;\theta_s,\theta_{\text{in}},w) - f_{\mathsf{head}}(X,z;\widehat{\theta}_s,\widehat{\theta}_{\text{in}},w)\right\| \le$$
$$C_{\mathsf{Norm}} L_{\text{out}} B_{\text{in}} B_X \left\|\mathsf{Score}(X,z;\theta_s) - \mathsf{Score}(X,z;\widehat{\theta}_s)\right\|_\infty + L_{\text{out}} \left\|\varphi_{\text{in}}(X;\theta_{\text{in}}) - \varphi_{\text{in}}(X;\widehat{\theta}_{\text{in}})\right\|_{2,\infty}.$$

The most crucial aspect of this proof is to avoid a spurious $T$ dependence when accounting for the attention mechanism. The key observation here is that the attention part of the network is computed using $\mathsf{Norm}$, whose Jacobian norm is bounded. This allows us to use the mean-value theorem to move to the maximum ($\ell_\infty$) error over $T$ tokens instead of sum ($\ell_1$), which could potentially incur a $T$ factor. Furthermore, this allows us to combine all samples and tokens and construct an $\ell_\infty$-cover directly for $mT$ samples.

### 4.4.2 Capacity of a Transformer head

Let us now look at the case of a Transformer self-attention head and instantiate the covering bound. For ease of presentation and to focus on the self-attention part, we collapse $W_Q W_K^\top$ to a single matrix, set $k = d$ and remove the linear layer $W_C$[*]. Then the Transformer self-attention head (for any fixed $\tau \in [T]$) can be described as

$$f_{\text{tf-head}}(X; W_V, W_{QK}) := \sigma\left(W_V^\top X^\top \mathsf{softmax}\left(X W_{QK}^\top x_\tau\right)\right)$$

which is obtained from the general formulation by setting the context to be $x_\tau$, $\mathsf{Score}(X, x_\tau; W_{QK}) = X W_{QK}^\top x_\tau$, $\mathsf{Norm} = \mathsf{softmax}$ and $\varphi_{\text{out}} = \sigma$.

Because the number of parameters in a Transformer self-attention head is $O(d^2)$, with no dependence on $T$, one might presume by simple parameter counting that the capacity of the class of these heads does not grow as the context length $T$ grows. But capacity is not solely governed by the number of parameters—for example, the class $\{x \in \mathbb{R} \mapsto \text{sign}(\sin(\alpha x))\}_{\alpha \in \mathbb{R}}$ has a single parameter but infinite VC-dimension. One might still hope to prove, for the special case of Transformer heads, a $T$-independent upper bound on the VC-dimension (or rather, its analog for real-valued functions, the *pseudo-dimension*). We observe that, in fact, the pseudo-dimension of this class does grow with $T$.

**Proposition 4.8.** *When the embedding dimension is $d = 3$, the class*

$$\mathcal{F}_{\text{tf-head-unbounded}} := \{X \mapsto f_{\text{tf-head}}(X; W_V, W_{QK}) : W_V, W_{QK} \in \mathbb{R}^{d \times d}\}$$

*of Transformer self-attention heads with unbounded norm has pseudo-dimension $\geq \lfloor \log T \rfloor$.*

The proofs for this subsection can be found in Appendix A.1.

[*] See Appendix A.1.7 for an analysis of general deep Transformer models.

Let us now define the function class of self-attention heads with bounded weight norms:

$$\mathcal{F}_{\text{tf-head}} := \{X \mapsto f_{\text{tf-head}}(X; W_V, W_{QK}) : \quad \|W_V\|_{2,1} \le B_V^{2,1}, \|W_V\| \le B_V, \|W_{QK}^\top\|_{2,1} \le B_{QK}^{2,1}\}.$$

Since $W_V$, $W_{QK}$ have dimensions dependent on $d$ and $k$, bounding their norms does not hide a $T$ dependence. As before, to convert this vector-valued function class to a scalar output function class, we define

$$\mathcal{F}_{\text{tf-scalar}} := \quad \{X \mapsto w^\top f(X) : f \in \mathcal{F}_{\text{tf-head}}, w \in \mathbb{R}^d, \|w\| \le B_w\}.$$

We obtain the following bound on the covering number of $\mathcal{F}_{\text{tf-head}}$ as a corollary of Theorem 4.6:

**Corollary 4.9.** *For any $\varepsilon > 0$ and $X^{(1)}, \ldots, X^{(m)} \in \mathbb{R}^{T\times d}$ such that $\left\|X^{(i)\top}\right\|_{2,\infty} \le B_X$ for all $i \in [m]$, the covering number of $\mathcal{F}_{\text{tf-head}}$ satisfies*

$$\log \mathcal{N}_\infty(\mathcal{F}_{\text{tf-head}}; \varepsilon; X^{(1)}, \ldots, X^{(m)}, \|\cdot\|_2) \lesssim (L_\sigma B_X)^2 \cdot \frac{\left((B_V^{2,1})^{\frac{2}{3}} + (B_{QK}^{2,1} B_V)^{\frac{2}{3}}\right)^3}{\varepsilon^2} \cdot \log(mT)$$

*Here $\lesssim$ hides logarithmic dependencies on quantities besides $m$ and $T$.*

Proof overview. The above result follows from bounding the covering numbers of

$$\mathcal{F}_{QK} := \{z \mapsto x_\tau^\top W_{QK} z : \|W_{QK}^\top\|_{2,1} \le B_{QK}^{2,1}\}, \text{ and}$$

$$\mathcal{F}_V := \{z \to W_V^\top z : \|W_V\|_{2,1} \le B_V^{2,1}, \|W_V\| \le B_V\}.$$

Note that $|x_\tau^\top W_{QK} z - x_\tau^\top W_{QK} \widehat{z}| \le \|W_{QK} z - W_{QK}\widehat{z}\|$ since $\|x_\tau\| \le 1$, so the covering number of $\mathcal{F}_{QK}$ is at most the covering number of the class of functions of the form $z \mapsto W_{QK} z$. Therefore,

a covering number bound for the vector-valued linear function class suffices to handle both covering numbers:

**Lemma 4.10.** *Let* $\mathcal{W} : \{W \in \mathbb{R}^{d_1 \times d_2} : \|W^\top\|_{2,1} \le B_W\}$*, and consider the function class* $\mathcal{F} : \{x \mapsto Wx : W \in \mathcal{W}\}$*. For any* $\varepsilon > 0$ *and* $x^{(1)}, \dots, x^{(N)} \in \mathbb{R}^{d_1}$ *satisfying* $\forall i \in [N], \left\|x^{(i)}\right\| \le B_X$*,*

$$\log \mathcal{N}_\infty(\mathcal{F}; \varepsilon; x^{(1)}, \dots, x^{(N)}; \|\cdot\|_2) \lesssim \frac{(B_X B_W)^2}{\varepsilon^2} \log(d_1 N).$$

Note that this bound only depends logarithmically on the context length, as desired. The proof can be found in Appendix A.1.

Finally, our analysis is compatible with the following additional components:

Positional embeddings. In practice, the permutation-invariant symmetry of a Transformer network is broken by adding a *positional embedding* matrix $P \in \mathbb{R}^{T \times d}$ to the input $X$ at the first layer. In practice, the embedding matrix is often fixed (not trainable). Our results extend to this setting in a straightforward way; see Appendix A.1.5. If these matrices are to be trained from a sufficiently large class (say, $\|P\|_{2,\infty} \le 1$), the dependence of the log-covering number on $T$ could become linear.

Residual connections. Including residual connections (e.g. redefining $f_{\text{tf-head}}(X)$ as $x_t + f_{\text{tf-head}}(X)$ for some index $t \in [T]$) simply increases the Lipschitz constant of each layer (w.r.t. the input) by at most 1. As long as $B_V = \Omega(1)$, this only changes our covering number bounds by a constant factor.

Multi-head self-attention. In almost all applications of Transformers, multiple parallel self-attention heads are used, and their outputs aggregated, to allow for a richer representation. Our analysis directly extends to this setting; see Appendix A.1.6 for details. When a single attention

head is replaced with the sum of $H$ parallel heads, the log-covering number scales up by a factor of $\text{poly}(H)$.

Layer normalization. State-of-the-art Transformer networks are trained with layer normalization modules (Ba et al., 2016), which is generally understood to aid optimization. We keep a variant of layer normalization in the covering number analysis– it proves to be useful in the analysis of full attention blocks (see Appendix A.1.7), as it keeps the norm of the embedding of each token bounded. Removing these layers would lead to a worse dependence on the spectral norm of the matrices.

### 4.4.3 Capacity bounds for multi-layer Transformers

In this section, we will extend our results for $L$-layer Transformer blocks. Denote the weights of layer $i$ by $W^{(i)} := \left\{ W_Q^{(i)}, W_K^{(i)}, W_V^{(i)}, W_C^{(i)} \right\}$. Further denote the set of weights up to layer $i$ by $W^{1:i} = (W^{(1)}, \ldots, W^{i-1})$. Denote the input representation of layer $i$ by $g^{(i)}_{\text{tf-block}}(X; W^{1:i})$. We inductively define $g^{(i)}_{\text{tf-block}} : \mathbb{R}^{T\times d} \to \mathbb{R}^{T\times d}$ starting with $g^{(1)}_{\text{tf-block}}(X; W^{1:1}) = X$ (the input):

$$g^{(i+1)}_{\text{tf-block}}\left(X; W^{1:i+1}\right) := \Pi_{\text{norm}}\left(\sigma\left(\Pi_{\text{norm}}\left(f\left(g^{(i)}_{\text{tf-block}}\left(X; W^{1:i}\right); W^{(i)}\right)\right)\right) W_C^{(i)}\right),$$
$$\text{with } f\left(Z; \{W_Q, W_K, W_V, \cdot\}\right) := \text{RowSoftmax}\left(ZW_Q\left(ZW_K\right)^\top\right) ZW_V,$$

where $\Pi_{\text{norm}}$ denotes layer normalization* applied to each row. We use a slightly modified version of LayerNorm where instead of normalizing to norm 1, we project it to the unit ball. Let the class of

*Layer normalization allows for the norms of the outputs of each token in each layer to remain bounded by 1. Note that the norm of the entire input can still have a dependence on $T$. Our results would go through with a worse dependence on the spectral norms if we were to remove layer norm.

depth-$L$ transformer blocks be

$$\mathcal{F}^{(L)}_{\text{tf-block}} := \Big\{ X \to g^{(L+1)}_{\text{tf-block}}(X; W^{1:L+1}) : \forall\, i \in [L], \\ \left\| W_V^{(i)} \right\|_2, \left\| W_K^{(i)} W_Q^{(i)\top} \right\|_2, \left\| W_C^{(i)} \right\|_2 \le C_2, \\ \left\| W_V^{(i)} \right\|_{2,1}, \left\| W_K^{(i)\top} W_Q^{(i)} \right\|_{2,1}, \left\| W_C^{(i)} \right\|_{2,1} \le C_{2,1} \Big\}.$$

To obtain a final scalar output, we use a linear function of the `[CLS]` output:

$$g_{\text{tf-scalar}}(X; W^{1:L+1}, w) = w^\top \left[ g\left(X; W^{1:L+1}\right) \right]_{\texttt{[CLS]},:}.$$

Let the scalar output function class be $\mathcal{F}^{(L)}_{\text{tf-scalar}} := \{X \to w^\top f(X)_{\texttt{[CLS]}} : f \in \mathcal{F}^{(L)}_{\text{tf-block}}, w \in \mathbb{R}^d, \|w\| \le B_w\}$.

**Theorem 4.11** (Theorem A.17 (simplified)). *Suppose* $\forall i \in [m], \left\|X^{(i)}\right\|_{2,\infty} \le B_X$, *then we have*

$$\log \mathcal{N}_\infty(\mathcal{F}^{(L)}_{\text{tf-block}}; \varepsilon; X^{(1)}, \ldots, X^{(m)}) \lesssim (C_2 L_\sigma)^{O(L)} \cdot \frac{B_X^2 B_w^2 C_{2,1}^2}{\varepsilon^2} \cdot \log(dmT).$$

Note that the dependence on $d$ and $T$ is only logarithmic even for deeper networks. The dependence on $(2,1)$-norms of the weight matrices is quadratic. As long as the spectral norms of the matrices are bounded by 1 and $\sigma$ is 1-Lipschitz (which holds for sigmoids and ReLUs), the exponential dependence on $L$ can be avoided.

## 4.5 Attention approximates sparse functions

The results in Section 4.4 show that function classes bottlenecked by self-attention mechanisms have "small" statistical capacity in terms of the context size. In this section, we answer the converse

question: *which functions of interest are in these classes?* We show that Transformers are able to represent sparse interactions in the context with bounded weight norms, and can thus learn them sample-efficiently.

Consider the class of Boolean functions $f : \{0,1\}^T \to \mathbb{R}$ which are *s-sparse*: they only depend on $s \ll T$ of their inputs. We will construct mappings from such functions to parameters of a self-attention head $f_{\text{tf-head}}$ composed with a feedforward network $f_{\text{mlp}}$; note that $f_{\text{tf-head}} \circ f_{\text{mlp}}$ is the standard Transformer block. Intuitively, $f_{\text{tf-head}}$ is constructed to "keep" the correct $s$-dimensional subset of inputs and "forget" the rest, while $f_{\text{mlp}}$ "memorizes" the values of $f$ on these $s$ inputs, using $2^{O(s)}$ parameters.

Setup. We consider the classes of Boolean functions $f : \{0,1\}^T \to \mathbb{R}$ representable by bounded-norm scalar-output Transformer heads $f_{\text{tf-scalar}} : \mathbb{R}^{T\times d} \to \mathbb{R}$. To do this, we must first fix a mapping from $\{0,1\}^T$ to $\mathbb{R}^{T\times d}$; we discuss several natural choices in Appendix A.2.1. The simplest of these uses a sum of token and positional embeddings $X(b)_{t,:} := e_{b_t} + v_t$, for a set of approximately orthogonal unit vectors $\{e_0, e_1\} \cup \{v_1, \ldots, v_T\}$ of dimension $d = \Theta(\log T)$. After choosing a mapping $X(b)$, the setup of the representation problem is as follows: given $f(b)$, find Transformer weights $\theta_{\text{tf-head}}$ and feedforward network weights $\theta_{\text{mlp}}$ such that

$$f_{\text{tf+mlp}}(X(b); \theta_{\text{tf-head}}, \theta_{\text{mlp}}) := \quad f_{\text{mlp}}\left(f_{\text{tf-head}}(X(b); \theta_{\text{tf-head}}); \theta_{\text{mlp}}\right) \approx f(b), \forall b \in \{0,1\}^T.$$

Main representational results. For any size-$s$ subset of indices $\mathcal{I} \subseteq [T]$, we show that Transformer blocks can represent all $\mathcal{I}$-sparse Boolean functions, whose values only depend on the inputs at the coordinates in $\mathcal{I}$. We give informal statements of these approximation results below, and present the precise statements in Appendix A.2.2.

**Proposition 4.12** (Sparse variable creation via Transformers; informal)**.** *Under any of the input*

*mappings $X(b)$, we have the following guarantees:*

- $f_{\text{tf-scalar}}$ *can approximate a particular monotone symmetric s-sparse Boolean function, with norms* $\left\|W_Q\right\|_F \leq O\left(\log(Ts)\right)$ ; $\|W_K\|_F, \|W_V\|_F, \|W_C\|_F \leq O(s)$.
- $f_{\text{tf+mlp}}$ *can exactly represent symmetric s-sparse functions, with the same Transformer weight norms as above; the feedforward network weights satisfy* $\|W_1\|_F, \|W_2\|_F, \|w\|_F \leq O(\text{poly}(s))$.
- $f_{\text{tf+mlp}}$ *can exactly represent general s-sparse functions, with the same Transformer weight norms as above; the feedforward network weights satisfy* $\|W_1\|_F, \|W_2\|_F, \|w\|_F \leq O(2^s \cdot \text{poly}(s))$.

These results and the capacity bounds from Section 4.4 are simultaneously meaningful in the regime of $s \ll \log T$. An appealing interpretation for the $s = 2$ case is that a single Transformer head can learn a single logical gate (i.e. AND, OR, NAND, ...) in a Boolean circuit, with $d$ and weight norms scaling as $\log T$.

Proof ideas. Each construction uses the same basic idea: select $W_Q$, $W_K$ so that the attention mixture weights approximate the uniform distribution over the relevant positions, then use the ReLU network to memorize all distinct values of $f$. Full proofs are given in Appendix A.2.5.

Other realizable functions. Since there are $\binom{T}{s}$ $s$-sparse subsets of input indices, the sample complexity of learning a sparse Boolean function must scale at least as $\Omega(s \log T)$, matching the capacity bounds in terms of the $\log T$ dependence. However, sparse functions are not the *only* potentially useful functions realizable by bounded-norm Transformers. For instance, with $W_Q = W_K = 0$, so that all scores are zero, a Transformer head can take an average of $T$ embeddings $X \to \frac{1}{T}1^\top X W_V$. More generally, departing from the "orthogonal context vectors" embedding of Boolean inputs but using the same constructions as in this section, it is straightforward to conclude that bounded-norm Transformers can compute global averages of tokens whose $XW_K$ embeddings lie in an $s$-dimensional subspaces. This is why our results do not contradict the empirical finding of

Clark et al. (2019) that some attention heads in trained Transformer models attend broadly. It is also straightforward to extend some of these results beyond Boolean domains; see Section A.2.4 for a sketch.

Bypassing theoretical limitations. Hahn (2020) points out that with constant weight norms, a Transformer's ability to express global dependencies degrades with context length: as $T \to \infty$, the maximum change in output caused by altering a single input token approaches 0, and thus various interesting formal languages cannot be modeled by a Transformer in this particular limit. The constructions in this section show that this can be circumvented by allowing $d$ and the weight norms to scale as $\log(T)$.

## 4.6 Experiments

Sections 4.4 and 4.5 show theoretically that Transformers can learn sparse Boolean functions, with sparse regression-like sample complexity (in terms of the $\log T$ dependence). In this section, we present an empirical study which probes the end-to-end sample efficiency of Transformer architectures with standard training and architecture hyperparameters, and how it scales with the context length $T$.

Setup. We introduce a synthetic benchmark to support our analysis, in which we measure the statistical limit for learning sparse Boolean functions with Transformers. We choose a distribution $\mathcal{D}$ on $\{0,1\}^T$, and a family of distinct functions $\{f_i : \{0,1\}^T \to \{0,1\}\}_{i=\in[N]}$, where $N$ grows with $T$. Then, we choose an $i^* \in [N]$ uniformly at random, and train a Transformer binary classifier on $m$ samples from $\mathcal{D}$, with labels given by $f_{i^*}$, evaluating generalization error via holdout samples. Then, for any learner to reach 100% accuracy on this sample, $m \geq \Omega(\log N)$ samples are required

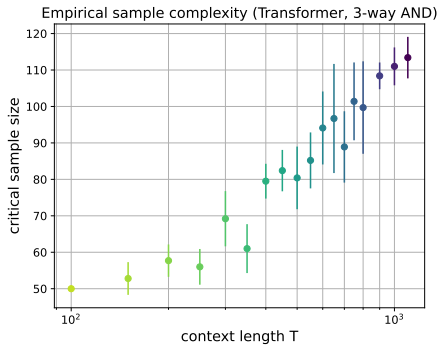



**Figure 4.2:** Main experimental finding: the sample complexity of learning a 3-sparse AND function of $T$ input bits with Transformers. For each $T$, we measure the smallest sample size $m$ necessary to reach $100\%$ validation accuracy on $\geq 80\%$ of random trials. We find that this threshold scales logarithmically with $T$.

(one sample reveals at most one bit of information about $i^*$). We can then measure the empirical scaling of the sufficient sample size $m$ to solve this problem, in terms of $N$ (and thus $T$).

Learning sparse conjunctions. Concretely, we can choose $f_i$ be the set of all $\binom{T}{s}$ conjunctions of $s$ inputs (e.g. $y = x_2 \wedge x_3 \wedge x_{10}$), fixing the input distribution $\mathcal{D}$ to be i.i.d. Bernoulli (we choose the bias to balance the labels). The model must learn which subset of $s$ features are relevant, out of $\binom{T}{s}$ possibilities; this requires at least $m \geq \Omega(s \log T)$ samples. The theoretical analysis predicts that the sample complexity of learning any function realizable by a bounded-norm Transformer should asymptotically have the same $\log T$ scaling. We choose a fixed sparsity parameter $s = 3$, and measure how the empirical sample complexity (i.e. the smallest sample size $m(T)$ at which model training succeeds with non-negligible probability) scales with $T$.

Results. With architecture and training hyperparameters typical of real Transformer setups (except the number of layers, which we set to 1), we indeed observe that the empirical sample complexity appears to scale as $\log T$; see Figure 4.2. Despite the exponentially large support of input bit strings $x$ and large total parameter count ($\sim 10^5$), the attention weights vanish on the $T - s$ irrele-

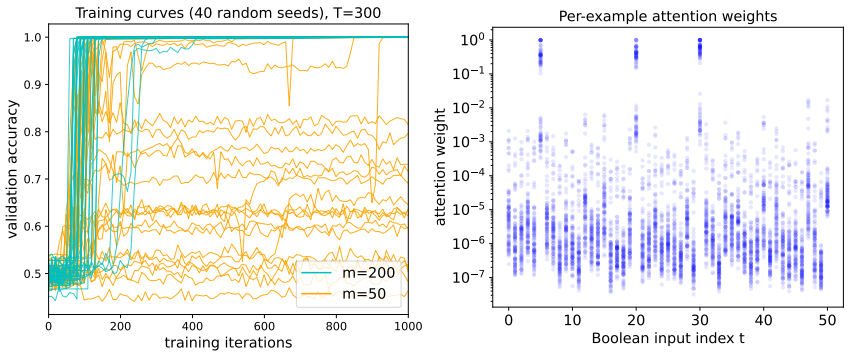



**Figure 4.3:** Additional visualizations for the sparse function learning experiments. *Left:* Examples of validation accuracy curves on the same problem instance ($T = 300$), with sample sizes above ($m = 200$) and below ($m = 50$) the threshold ($\approx 70$ from Figure 4.2). Training accuracy goes to $100\%$ in both cases, but the Transformer overfits (orange curves) when $m$ is too small. *Right:* Per-example attention weights for a successfully trained model ($T = 50$, $m = 300$, $\mathcal{I} = \{5, 20, 30\}$). The input-dependent attention weights approximately zero out the irrelevant bits.

vant coordinates, and the model converges to sparse solutions; this is visualized in Figure 4.3 *(right)*. Details are provided in Appendix A.3.1; in particular, model training near the statistical threshold is extremely unstable, and extensive variance reduction (best of 5 random restarts; 40 replicates; a total of $\sim 10^4$ training runs across each $T, m$) was necessary to produce these scaling plots.

Beyond our analysis: sparse parities. When choosing the family of sparse functions $\{f_i\}$, we can replace the AND operation with XOR: the label is the parity of a randomly chosen subset of i.i.d. uniform input bits. In this setting, unlike the AND case, there is a computational-statistical gap: $\Theta(s \log T)$ samples suffice to identify, but the fastest known algorithms for learning parities with noise require $T^{\Omega(s)}$ time. In the statistical query model, $\Omega(T^s)$ iterations of noisy batch gradient descent are necessary (Kearns, 1998). Figure 4.4 (with details in Appendix A.3.2) shows that when trained with i.i.d. samples, Transformer models can learn sparse parities. This raises an intriguing question, which is the *computational* analogue of the current work's *statistical* line of inquiry: how does local search (i.e. gradient-based training) succeed at finding solutions that correspond to sparse discrete functions? The present work merely shows that these solutions exist; we intend to

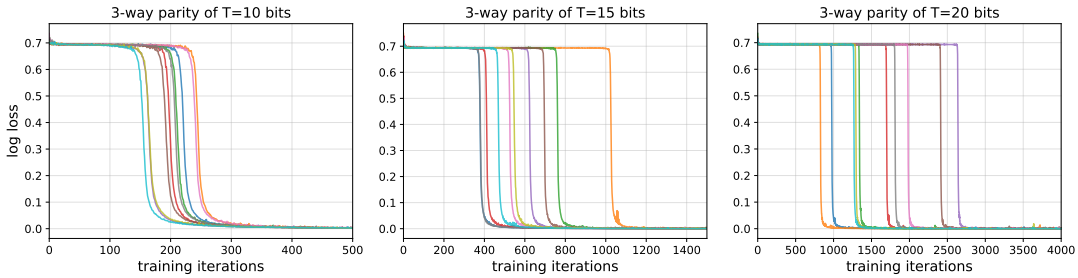



**Figure 4.4:** A curious empirical finding: Transformers can learn sparse parities. Loss curves (across 10 random seeds for initialization and SGD samples) are shown for this setup with $s = 3, T \in \{10, 15\}$, exhibiting phase transitions from random guessing to $100\%$ accuracy. See Appendix A.3.2 for details.

address the computational mystery in future work.

## 4.7 Conclusion and future work

This work establishes a statistical analysis of attention and self-attention modules in neural networks. In particular, we identify an inductive bias we call *sparse variable creation*, consisting of (1) covering number-based capacity bounds which scale as $\log T$, and (2) constructions which show that self-attention models with small weight norms can represent sparse functions. This analysis is supported by an empirical study on learning sparse Boolean functions with Transformers. We hope that these rigorous connections between attention and sparsity, as well as the proposed experimental protocols, will inform the practice of training and regularizing these models, and the design of future attention-based architectures.

We believe that it is possible to refine the covering number bounds (where we have only sought to obtain optimal dependences on $T$) as well as the representation results (where we have not used the structure of the MLP, beyond its capacity for exhaustive memorization). Significant challenges (which are not specific to attention) remain in closing the theory-practice gap: precisely understanding the role of depth, as well as the trajectory of the optimization algorithm.

An exciting line of empirical work has made progress on understanding and interpreting state-

of-the-art Transformer language models by examining the activations of their attention mechanisms (Clark et al., 2019; Tenney et al., 2019; Rogers et al., 2020). In some cases, these works have found instances in which Transformers seem to have learned features that are reminiscent of (sparse) hand-crafted features used in natural language processing. Reconciling our theoretical foundations work with this area of *BERTology* is an avenue for future synthesis.

*This chapter is based on "Hidden Progress in Deep Learning: SGD Learns Parities Near the Computational Limit" (Barak et al., 2022), written in collaboration with Boaz Barak, Surbhi Goel, Sham Kakade, Eran Malach, and Cyril Zhang.*

# 5

# Hidden Progress

There is mounting evidence of *emergent phenomena* in the capabilities of deep learning methods as we scale up datasets, model sizes, and training times. While there are some accounts of how these resources modulate statistical capacity, far less is known about their effect on the *computational* problem of model training. This work conducts such an exploration through the lens of learning a $k$-sparse parity of $n$ bits, a canonical discrete search problem which is statistically easy but computationally hard. Empirically, we find that a variety of neural networks successfully learn sparse parities,

with discontinuous phase transitions in the training curves. On small instances, learning abruptly occurs at approximately $n^{O(k)}$ iterations; this nearly matches SQ lower bounds, despite the apparent lack of a sparse prior. Our theoretical analysis shows that these observations are *not* explained by a Langevin-like mechanism, whereby SGD "stumbles in the dark" until it finds the hidden set of features (a natural algorithm which also runs in $n^{O(k)}$ time). Instead, we show that SGD gradually amplifies the sparse solution via a *Fourier gap* in the population gradient, making continual progress that is invisible to loss and error metrics.

## 5.1 Introduction

In deep learning, performance improvements are frequently observed upon simply scaling up resources (such as data, model size, and training time). While these improvements are often continuous in terms of these resources, some of the most surprising recent advances in the field have been *emergent capabilities*: at a certain threshold, behavior changes qualitatively and *discontinuously*. Through a statistical lens, it is well-understood that larger models, trained with more data, can fit more complex and expressive functions. However, far less is known about the analogous *computational* question: *how does the scaling of these resources influence the success of gradient-based optimization?*

These *phase transitions* cannot be explained via statistical capacity alone: they can appear even when the amount of data remains fixed, with only model size or training time increasing. A timely example is the emergence of reasoning and few-shot learning capabilities when scaling up language models (Radford et al., 2019; Brown et al., 2020; Chowdhery et al., 2022; Hoffmann et al., 2022); Srivastava et al. (2022) identify various tasks which language models are only able to solve if they are larger than a critical scale. Power et al. (2021) give examples of discontinuous improvements in population accuracy ("grokking") when running time increases, while dataset and model sizes

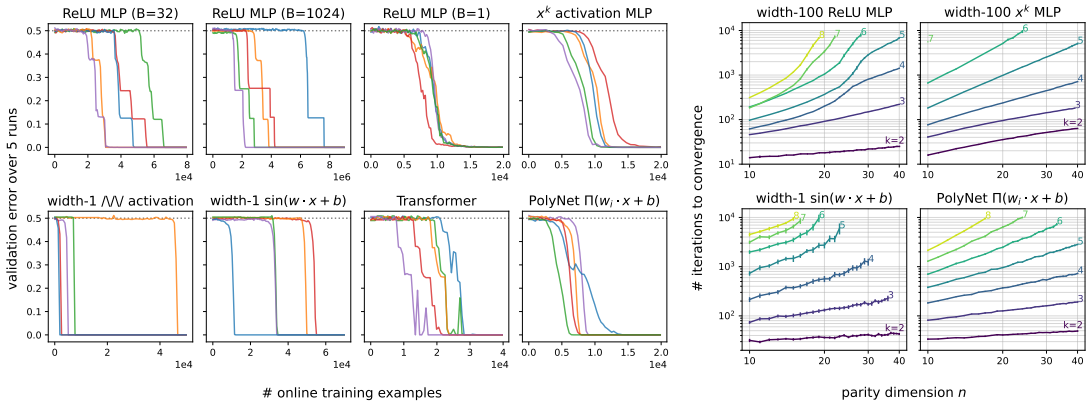



**Figure 5.1:** Main empirical findings at a glance. A variety of neural networks, with standard training and initialization, can solve the $(n, k)$-parity learning problem, with a number of iterations scaling as $n^{O(k)}$. *Left:* Training curves under various algorithmic choices (architecture, batch size, learning rate) on the $(n = 50, k = 3)$-parity problem. *Right:* Median convergence times for small $(n, k)$.

remain fixed.

In this work, we analyze the computational aspects of scaling in deep learning, in an elementary synthetic setting which already exhibits discontinuous improvements. Specifically, we consider the supervised learning problem of *learning a sparse parity*: the label is the parity (XOR) of $k \ll n$ bits in a random length-$n$ binary string. This problem is computationally difficult for a range of algorithms, including gradient-based (Kearns, 1998) and streaming (Kol et al., 2017) algorithms. We focus on analyzing the resource measure of *training time*, and demonstrate that the loss curves for sparse parities display a phase transition across a variety of architectures and hyperparameters (see Figure 5.1, left). Strikingly, we observe that SGD finds the sparse subset (and hence, reaches 0 error) with a variety of activation functions and initialization schemes, even with *no* over-parameterization.

A natural hypothesis to explain SGD's success in learning parities, with no visible progress in error and loss for most of training, would be that it simply "stumbles in the dark", performing random search for the unknown target (e.g. via stochastic gradient Langevin dynamics). If that were the case, we might expect to observe a convergence time of $2^{\Omega(n)}$, like a naive search over parameters or subsets of indices. However, Figure 5.1 *(right)*, already provides some evidence against this "ran-

dom search" hypothesis: the convergence time adapts to the sparsity parameter $k$, with a scaling of $n^{O(k)}$ on small instances. Notably, such a convergence rate implies that SGD is closer to achieving the *optimal* computation time among a natural class of algorithms (namely, statistical query algorithms).

Through an extensive empirical analysis of the scaling behavior of a variety of models, as well as theoretical analysis, we give strong evidence against the "stumbling in the dark" viewpoint. Instead, there is a *hidden progress measure* under which SGD is steadily improving. Furthermore, and perhaps surprisingly, we show that SGD achieves a computational runtime much closer to the optimal SQ lower bound than simply doing (non-sparse) parameter search. More generally, our investigations reveal a number of notable phenomena regarding the dependence of SGD's performance on resources: we identify phase transitions when varying data, model size, and training time.

### 5.1.1 Our contributions

SGD LEARNS SPARSE PARITIES. It is known from SQ lower bounds that with a constant noise level, gradient descent on *any* architecture requires at least $n^{\Omega(k)}$ computational steps to learn $k$-sparse $n$-dimensional parities (for background, see Appendix B.1). We first show a wide variety of positive empirical results, in which neural networks successfully solve the parity problem in a number of iterations which scales near this computational limit:

*Empirical Finding* 5.1. For all small instances ($n \leq 30, k \leq 4$) of the sparse parity problem, architectures $\mathcal{A} \in \{$2-layer MLPs, Transformers*, sinusoidal/oscillating neurons, PolyNets†$\}$, initializations in $\{$uniform, Gaussian, Bernoulli$\}$, and batch sizes $1 \leq B \leq 1024$, SGD on $\mathcal{A}$ solves the $(n, k)$-sparse parity problem (w.p. $\geq 0.2$) within at most $c \cdot n^{\alpha k}$ steps, for small constants $c, \alpha$.

*With a smaller range of hyperparameters.

†A non-standard architecture introduced in this work; see Section 5.3 for the definition.

Theoretical analyses of sparse feature emergence. Our empirical results suggest that, in a number of computational steps matching the SQ limit, SGD is able to solve the parity problem and identify the influential coordinates, without an explicit sparse prior. We give a theoretical analysis which validates this claim.

*Informal Theorem* 5.2. On 2-layer MLPs of width $2^{\Theta(k)}$, and with batch size $n^{O(k)}$, SGD converges with high probability to a solution with at most $\varepsilon$ error on the $(n, k)$-parity problem in at most $2^{O(k)} \cdot \mathrm{poly}(1/\varepsilon)$ iterations.

We also present a stronger analysis for an idealized architecture (which we call the *disjoint-PolyNet*), which allows for any batch size, and captures the phase transitions observed in the error curves.

*Informal Theorem* 5.3. On disjoint-PolyNets, SGD (with any batch size $B \geq 1$) converges with high probability to a solution with at most $\varepsilon$ error on the $(n, k)$-parity problem in at most $n^{O(k)} \cdot \log(1/\varepsilon)$ iterations. Continuous-time gradient flow exhibits a phase transition: it spends a $1 - o(1)$ fraction of its time before convergence with error $\geq 49\%$.

Our theoretical and empirical results hold in non-overparameterized regimes (including with a width-1 sinusoidal neuron), in which no fixed kernel, including the neural tangent kernel (NTK) (Jacot et al., 2018), is sufficiently expressive to fit all sparse parities with a large margin. Thus, our findings comprise an elementary example of *combinatorial feature learning*: SGD can only successfully converge by learning a low-width sparse representation.

Further empirical explorations. Building upon our core positive results, we provide a wide variety of preliminary experiments, showing sparse parity learning to be a versatile testbed for understanding the challenges and surprises in solving combinatorial problems with neural networks. These include quantities which reveal the continual *hidden progress* behind uninformative training curves (as predicted by the theory), experiments at small sample sizes which exhibit *grokking* (Power

et al., 2021), as well as an example where greedy layer-wise learning is impossible but end-to-end SGD can learn the layers jointly.

### 5.1.2 Related work

We present the most directly related work on feature learning, and learning parities with neural nets. A broader discussion can be found in Appendix B.1.3.

SGD AND FEATURE LEARNING. Theoretical analysis of gradient descent on neural networks is notoriously hard, due to the non-convex nature of the optimization problem. That said, it has been established that in some settings, the dynamics of GD keep the weights close to their initialization, thus behaving like convex optimization over the Neural Tangent Kernel (see, for example, (Jacot et al., 2018; Allen-Zhu et al., 2019; Du et al., 2018)). In contrast, it has been shown that in various tasks, moving away from the fixed features of the NTK is essential for the success of neural networks trained with GD (for example (Yehudai & Shamir, 2019; Allen-Zhu & Li, 2019; Wei et al., 2019a) and the review in (Malach et al., 2021)). These results demonstrate that feature learning is an important part of the GD optimization process. Our work also focuses on a setting where feature learning is essential for the target task. In our theoretical analysis, we show that the initial population gradient encodes the relevant features for the problem. The importance of the first gradient step for feature learning has been recently studied in (Ba et al., 2022).

LEARNING PARITIES WITH NEURAL NETWORKS. The problem of learning parities using neural networks has been investigated in prior works from various perspectives. It has been demonstrated that parities are hard for gradient-based algorithms, using similar arguments as in the SQ analysis (Shalev-Shwartz et al., 2017; Abbe & Sandon, 2020). One possible approach for overcoming the computational hardness is to make favorable assumptions on the input distribution. Indeed, re-

cent works show that under various assumptions on the input distribution, neural networks can be efficiently trained to learn parities (XORs) (Daniely & Malach, 2020; Shi et al., 2021; Frei et al., 2022a; Malach et al., 2021). In contrast to these results, this work takes the approach of intentionally focusing on a hard benchmark task, without assuming that the distribution has some favorable (namely, non-uniform) structure. This setting allows us to probe the performance of deep learning at a known computational limit. Notably, the work of Andoni et al. (2014) provides analysis for learning polynomials (and in particular, parities) under the uniform distribution. However, their main results require a network of size $n^{O(k)}$ (i.e., extremely overparameterized network), and provides only partial theoretical and empirical evidence for the success of smaller networks. Studying a related subject, some works have shown that neural networks display a spectral bias, learning to fit low-frequency coefficients before high-frequency ones (Rahaman et al., 2019; Cao et al., 2019).

## 5.2 Preliminaries

We provide an expanded discussion of background and related work in Appendix B.1.

Sparse parities. For integer $n \geq 1$ and non-empty set $S \subseteq [n]$, the $(n, S)$-*parity function* $\chi_S : \{\pm 1\}^n \to \{\pm 1\}$ is defined as $\chi_S(x) = \prod_{i \in S} x_i$. We define the $(n, S)$-*parity distribution* $\mathcal{D}_S$ as the joint distribution over $(x, y)$* where $x$ is drawn from $\mathrm{Unif}(\{\pm 1\}^n)$, the uniform distribution over random length-$n$ sign vectors, and $y := \chi_S(x)$ is the product of the inputs at the indices given by the "relevant features" $S$ (thus, $\pm 1$, depending on whether the number of relevant $-1$ inputs is even or odd). We define the $(n, k)$-*parity learning problem* as the task of recovering the $S$ using samples from $\mathcal{D}_S$, where $S$ is chosen at random from $\binom{[n]}{k}$.

A key fact about parities is that they are orthogonal under the correlation inner product: for

*Our theoretical analyses and experiments can tolerate noisy parities, that is, random flipping of the label; see Appendix B.3.6. For ease of presentation, we state the noiseless setting in the main paper.

$S' \subseteq [n]$,

$$\mathop{\mathbb{E}}_{x\sim\mathrm{Unif}(\{\pm1\}^n)}\left[\chi_S(x)\chi_{S'}(x)\right] = \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}_S}\left[\chi_{S'}(x)\,y\right] = \begin{cases} 1 & S' = S \\ 0 & \text{otherwise} \end{cases}.$$

That is, a learner who guesses indices $S'$ cannot use correlations (equivalently, the accuracy of the hypothesis $\chi_{S'}$) as feedback to reveal which indices in $S'$ are correct, unless $S'$ is *exactly* the correct subset. This notion of indistinguishability leads to a *computational* lower bound in the statistical query (SQ) model (Kearns, 1998): $\Omega(n^k)$ constant-noise queries are necessary, which is far greater than the statistical limit of $\Theta(\log\binom{n}{k}) \approx k\log n$ samples. The hardness of parity has been used to derive computational hardness results for other settings, like agnostically learning halfspaces (Klivans & Kothari, 2014) and MLPs (Goel et al., 2019). Beyond the restricted computational model of statistical queries, noiseless parities can be learned in $\mathrm{poly}(n)$ time via Gaussian elimination. However, learning sparse *noisy* parities, even at a very small noise level (i.e., $o(1)$ or $n^{-\delta}$), is believed to inherently require $n^{\Omega(k)}$ computational steps.* In all, learning sparse parities is a well-studied combinatorial problem which exemplifies the computational difficulty of learning a joint dependence on multiple relevant features.

Notation for neural networks and training. Our main results are presented in the *online learning* setting, with a stream of i.i.d. batches of examples. At each iteration $t = 1, \ldots, T$, a learning algorithm $\mathcal{A}$ receives a batch of $B$ examples $\{(x_{t,i}, y_{t,i})\}_{i=1}^{B}$ drawn i.i.d. from $\mathcal{D}_S$, then outputs a classifier $\widehat{y}_t : \{\pm1\}^n \to \{\pm1\}$. We say that $\mathcal{A}$ solves the parity task in $t$ steps (with error $\varepsilon$) if

$$\Pr_{(x,y)\sim\mathcal{D}_S}[\widehat{y}_t(x) = y] \geq 1 - \varepsilon.$$

*This was first explicitly conjectured by Alekhnovich (2003), and has been the basis for several cryptographic schemes (e.g., (Ishai et al., 2008; Applebaum et al., 2009, 2010; Bogdanov et al., 2019)).

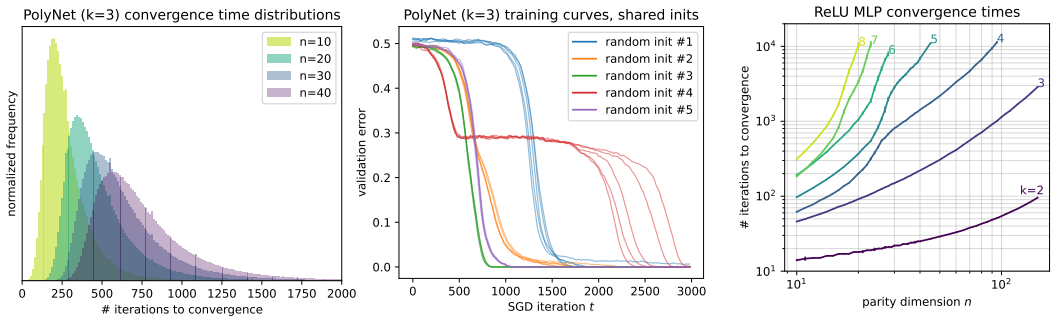



**Figure 5.2:** Black-box observations on the training dynamics. *Left:* Histograms of convergence times over $10^6$ random trials, with heavy upper tails but no observed successes near $t = 0$ (unlike random search). *Center:* Loss curves (and thus, convergence time) depend heavily on initialization, not the randomness of SGD; $B = 128, \eta = 0.01$ are shown here. *Right:* The power-law exponent ($\alpha$ such that $t_c \propto n^\alpha$) eventually worsens on larger problem instances.

We will focus on the case that $\widehat{y}_t = \text{sign}(f(x;\theta_t))$ for some parameters $\theta_t$ in a continuous domain $\Theta$ and for a continuous function $f : \{\pm 1\}^n \times \Theta \to \mathbb{R}$,* updated with the ubiquitous online learning algorithm of gradient descent (GD), whose update rule is given by

$$\theta_{t+1} \leftarrow (1-\lambda_t)\theta_t - \eta_t \cdot \nabla_\theta \left( \frac{1}{B} \sum_{i=1}^{B} \ell(y_{t,i}, f(x_{t,i};\theta_t)) \right), \tag{5.1}$$

for a loss function $\ell : \{\pm 1\} \times \mathbb{R} \to \mathbb{R}$, learning rate schedule $\{\eta_t\}_{t=1}^T$, and weight decay schedule $\{\lambda_t\}_{t=1}^T$†. The initialization $\theta_0$ is drawn randomly from a chosen distribution.

## 5.3 Empirical findings

### 5.3.1 SGD on neural networks learns sparse parities

The central phenomenon of study in this work is the empirical observation that neural networks, with standard initialization and training, can solve the $(n,k)$-parity problem in a number of iterations scaling as $n^{O(k)}$ on small instances. We observed robust positive results for randomly-initialized

*When $f(x;\theta) = 0$ in practice (e.g. with sign initialization), we break the tie arbitrarily. We ensure in the theoretical analysis that this does not happen.

†We allow different layers to have different learning rate and weight decay schedules.

SGD on the following architectures, indexed by Roman numerals:

- **2-layer MLPs:** ReLU ($\sigma(z) = (z)_+$) or polynomial ($\sigma(z) = z^k$) activation, in a wide variety of width regimes $r \geq k$. Settings (i), (ii), (iii) (resp. (iv), (v), (vi)) use $r = \{10, 100, 1000\}$ ReLU (resp. polynomial) activations. We also consider $r = k$ (exceptional settings (*i), (*ii) ), the minimum width for representing a $k$-wise parity for both activations.

- **1-neuron networks:** Next, we consider non-standard activation functions $\sigma$ which allow a one-neuron architecture $f(x; w) = \sigma(w^\top x)$ to realize $k$-wise parities. The constructions stem from letting $w^* = \sum_{i \in S} e_i$, and constructing $\sigma(\cdot)$ to interpolate (the appropriate scaling of) $\frac{k - w^{*\top} x}{2}$ mod 2 with a piecewise linear *k-zigzag* activation (vii), or a degree-$k$ polynomial (viii). Going a step further, a single $\infty$-*zigzag* (ix) or *sinusoidal* (x) neuron can represent *all* $k$-wise parities. In settings (xi), (xii), (xiii), (xiv), we remove the second trainable layer (setting $u = 1$). We find that wider architectures with these activations also train successfully.

- **Transformers:** There is growing interest in using parity as a benchmark for combinatorial function learning, long-range dependency learning, and length generalization in Transformers (Lu et al., 2021; Edelman et al., 2022; Hahn, 2020; Anil et al., 2022; Liu et al., 2022a). Motivated by these recent theoretical and empirical works, we consider a simplified specialization of the Transformer architecture to this sequence classification problem. This is the less-robust setting (*iii); the architecture and optimizer are described in Appendix B.4.1.

- **PolyNets:** Our final setting (xv) is the PolyNet, a slightly modified version of the parity machine architecture. Parity machines have been studied extensively in the statistical mechanics of ML literature (see the related work section) as well as in a line of work on 'neural cryptography' (Rosen-Zvi et al., 2002). A parity machine outputs the sign of the product of $k$ linear functions of the input. A PolyNet simply outputs the product itself. Both architectures can clearly realize $k$-sparse parities. The PolyNet architecture was originally motivated by the search for

an idealized setting where an end-to-end optimization trajectory analysis is tractable (see Section 5.4.1); we found in these experiments that this architecture trains very stably and sample-efficiently.

Robust space of positive results. All of the networks listed above were observed to successfully learn sparse parities in a variety of settings. We summarize our findings as follows: for all combinations of $n \in \{10, 20, 30\}$, $k \in \{2, 3, 4\}$, batch sizes $B \in \{1, 2, 4, \ldots, 1024\}$, initializations {uniform, Gaussian, Bernoulli}, loss functions {hinge, square, cross entropy}, and architecture configurations {(i), (ii), ..., (xv)}, SGD solved the parity problem (with 100% accuracy, validated on a batch of $2^{13}$ samples) in at least 20% of 25 random trials, for at least one choice of learning rate $\eta \in \{0.001, 0.01, 0.1, 1\}$. The models converged in $t_c \leq c \cdot n^{\alpha k} \leq 10^5$ steps, for small architecture-dependent constants $c, \alpha$ (see Appendix B.3). Figure 5.1 *(left)* shows some representative training curves.

Less robust configurations. Settings (*i) and (*ii), where the MLP just barely represents a $k$-sparse parity, and the Transformer setting (*iii), are less robust to small batch sizes. In these settings, the same positive results as above only held for sufficiently large batch sizes: $B \geq 16$. Also, setting (*iii) used the Adam optimizer (which is standard for Transformers); see Appendix B.4.1 for details.

Phase transitions in training curves. For almost all of the architectures, we find that that the training curves exhibit phase transitions in terms of running time (and thus, in the online learning setting, dataset size as well): long durations of seemingly no progress, followed by periods of rapid decrease in the validation error. Strikingly, for architectures (v) and (vi), this plateau is absent: the error in the initial phase appears to decrease with a linear slope. See Appendix B.3.8 for more plots.

### 5.3.2 Random search or hidden progress?

The remainder of this paper seeks to answer the question: *"By what mechanism does deep learning solve these emblematic computationally-hard optimization problems?"*

A natural hypothesis would be that SGD somehow implicitly performs Monte Carlo random search, "bouncing around" the loss landscape in the absence of a useful gradient signal. Upon closer inspection, several empirical observations clash with this hypothesis:

- **Scaling of convergence times:** Without an explicit sparsity prior in the architecture or initialization, it is unclear how to account for the runtimes observed in experiments, which adapt to the sparsity $k$. The initializations, which certainly do not prefer sparse functions*, are close to the correct solutions with probability $2^{-\Omega(n)} \ll n^{-k}$.

- **No early convergence:** Over a large number of random trials, no copies of this randomized algorithm get "lucky" (i.e. solve the problem in significantly fewer than the median number of iterations); see Figure 5.2 *(left)*. The success times of random exhaustive search would be distributed as $\mathrm{Geom}(1/\binom{n}{k})$, whose probability mass is highest at $t = 0$ and decreases monotonically with $t$.

- **Sensitivity to initialization, not SGD samples:** Running these training setups over multiple stochastic batches from a common initialization, we find that loss curves and convergence times are highly correlated with the architecture's random initialization, and are quite concentrated conditioned on initialization; see Figure 5.2 *(center)*.

- **Elbows in the scaling curves:** For larger $n$, the power-law scaling ceases to hold: the exponent worsens (see Figure 5.2 *(right)*, as well as the discussion in Appendix B.3.2). This would not be

*Indeed, under all standard architectures and initialization, the probability that a random network is $\Omega(1)$-correlated with a sparse parity would be $2^{-\Omega(n)}$, since with that probability $1 - o(1)$ of the total influence would be accounted by the $n - k$ irrelevant features.

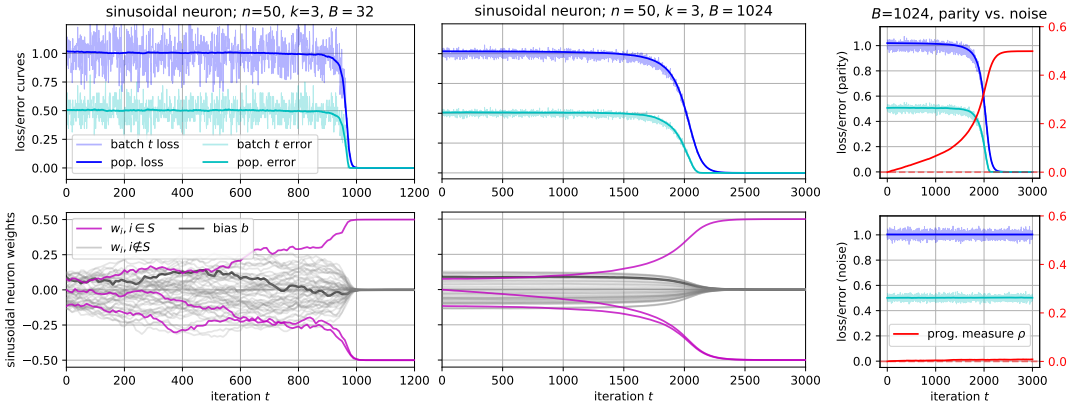



**Figure 5.3:** Hidden progress when learning parities with neural networks. *Left*, *center:* Black-box losses and accuracies exhibit a long plateau and sharp phase transition (top), hiding gradual progress in the SGD iterates (bottom). *Right:* A hidden progress measure which distinguishes gradual feature amplification (top) from training on noise (bottom).

true for random exhaustive search.

Even these observations, which do not probe the internal state of the algorithm, suggest that exhaustive search is an insufficient picture of the training dynamics, and a different mechanism is at play.

## 5.4 Theoretical analyses

### 5.4.1 Provable emergence of the parity indices in high-precision gradients

We now provide a theoretical account for the success of SGD in solving the $(n, k)$-parity problem. Our main theoretical observation is that, in many cases, the *population* gradient of the weights at initialization contains enough "information" for solving the parity problem. That is, given an accurate enough estimate of the initial gradient (by e.g. computing the gradient over a large enough batch size), the relevant subset $S$ can be found.

As a warm-up example, consider training a single ReLU neuron $\widehat{y}(x; w) = (w^\top x)_+$ with the correlation loss $\ell(y, \widehat{y}) = -y\widehat{y}$ over $\mathcal{D}_S$, from an all-ones initialization $w = [1 \ \dots \ 1] \in \mathbb{R}^n$. While a

single neuron cannot *express* the parity, we observe that the correct subset can be extracted from the population gradient at initialization:

$$\mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}_S}[\nabla_{w_i}\ell(y,\widehat{y}(x;w))] = \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}_S}\left[-y\,\nabla_{w_i}(w^\top x)_+\right] = \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}_S}\left[-\chi_S x_i 1\left[\sum_i x_i \geq 0\right]\right].$$

The key insight is that each coordinate in the above expression is a correlation between a parity and the function $x \mapsto -1[\sum_i x_i \geq 0]$, and thus a Fourier coefficient of this Boolean function. At each relevant coordinate ($i \in S$), the population gradient is the order-$(k-1)$ Fourier coefficient $S \setminus \{i\}$; for the irrelevant features ($i \notin S$), it is instead the order-$(k+1)$ coefficient $S \cup \{i\}$. All we require is a detectable *gap* between these quantities. Formally, letting $f(x;w) = \sigma(w^\top x)$, letting $\widehat{f}(S) := \mathbb{E}\left[f(x)\chi_S(x)\right]$ denote the Fourier coefficient of $f$ at $S$, we isolate the desired property:

**Definition 5.4** (Fourier gap)**.** For a function $f\colon \{\pm 1\}^n \to \mathbb{R}$ and $S \subseteq [n]$ of size $k$, we say that $f$ has a *$\gamma$-Fourier gap* at $S$ if, for every $(k-1)$-element subset $S^- \subset S$ and $(k+1)$-element superset $S^+ \supset S$, it holds that $|\widehat{f}(S^-)| \geq |\widehat{f}(S^+)| + \gamma$.

For the all-ones initialization, observe $1[\sum_i x_i \geq 0] = \frac{1+\mathrm{sign}\left(\sum_i x_i\right)}{2}$ is just an affine transformation of the majority function of $x$, for which a Fourier gap can be established, with $\gamma = \Theta(n^{-(k-1)/2})$. This arises from closed-form formulas for the Fourier spectrum of majority (see Lemma B.2 in Appendix B.2.1), a landmark result from the harmonic analysis of Boolean functions (Titsworth, 1962; O'Donnell, 2014). Thus, the coordinates in $S$ can be recovered from $\widetilde{O}(1/\gamma^2) = \widetilde{O}(n^{k-1})$ samples; see Proposition B.7 in Appendix B.2.2 for a formal argument.

Carefully extending this insight, we obtain an end-to-end convergence result for ReLU-activation MLP networks with a particular symmetric choice of $\pm 1$ initialization, trained with the hinge loss:

**Theorem 5.5** (SGD on MLPs learns sparse parities)**.** *Let $\varepsilon \in (0,1)$. Let $k \geq 2$ an even integer, and let $n = \Omega(k^4 \log(nk/\varepsilon))$ be an odd integer. Then, there exist a random initialization scheme, $\eta_t$, and*

*$\lambda_t$ such that for every $S \subseteq [n]$ of size $k$, SGD on a ReLU MLP of width $r = \Omega(2^k k \log(k/\varepsilon))$, with batch size $B = \Omega(n^k \log(n/\varepsilon))$ on $\mathcal{D}_S$ with the hinge loss, outputs a network $f(x;\theta_t)$ with expected[*] loss $\mathbb{E}\left[\ell(f(x;\theta_t), y)\right] \leq \varepsilon$ in at most $O(k^3 r^2 n/\varepsilon^2)$ iterations.*

This does not capture the full range of settings in which we empirically observe successful convergence. First, it requires a sign vector initialization, while we observe convergence with other random initialization schemes (namely, uniform and Gaussian). Second, it requires the batch size to scale with $n^{\Omega(k)}$[†], while we also obtain positive results when $B$ is small (even $B = 1$). Analogous statements for these cases (as well as other activations and losses) would require Fourier gaps for population gradient functions other than majority; lower bounds on the degree-$(k-1)$ coefficients (*"Fourier anti-concentration"*) are particularly elusive in the literature, and we leave it as an open challenge to establish them in more general settings. We provide preliminary empirics in Appendix B.3.1, suggesting that the Fourier gaps in our empirical settings are sufficiently large.[‡]

Low width necessitates feature learning. We note that in the low-width (non-overparameterized) regimes considered in this work, no fixed kernel (including the neural tangent kernel (Jacot et al., 2018), whose dimensionality is the network's parameter count) can solve the sparse parity problem. The following is a consequence of results in (Kamath et al., 2020; Malach & Shalev-Shwartz, 2022):

**Theorem 5.6** (Low-width NTK cannot fit all parities). *Let $\Upsilon : \{\pm 1\}^n \to \mathbb{R}^D$ be any $D$-dimensional embedding with $\sup_x \|\Upsilon(x)\|_2 \leq 1$. Let $R, \varepsilon > 0$, and let $\ell$ denote the 0-1 loss or hinge loss. If $DR^2 < \varepsilon^2 \cdot \binom{n}{k}$, then there exists some $S \subseteq [n]$ of size $k$ such that*

$$\inf_{\|w\| \leq R} \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}_S} \left[\ell(\Upsilon(x)^\top w, y)\right] > 1 - \varepsilon.$$

[*] The expectation is over the randomness of initialization, training and sampling $(x, y) \sim \mathcal{D}_S$.

[†] In fact, at this batch size, the correct parity indices emerge in a single SGD step.

[‡] Interestingly, we observe that the Fourier gap tends to *increase* over the course of training. This is not captured by our current theoretical analysis.

Thus, our low-width results lie outside the NTK regime, which requires far larger models (size $n^{\Omega(k)}$) to express parities. However, we note that better sample complexity bounds are possible in the NTK regime, with an algorithm more similar to standard SGD (see (Telgarsky, 2022) and Appendix B.1.3).

### 5.4.2 Disjoint-PolyNet: exact trajectory analysis for an idealized architecture

In this section, we present an architecture (a version of PolyNets (xv)) which empirically exhibits similar behavior to MLPs and bypasses the difficulty of analyzing Fourier gaps. The *disjoint-PolyNet* takes a product over $k$ linear functions of an equal-sized* partition $P_1, \ldots, P_k$ of the input coordinates: $f(x; w_{1:k}) := \prod_{i=1}^{k} \langle w_i, x_{P_i} \rangle$. As noted in the Section 5.1.2, this is equivalent to a tree parity machine, with real-valued (instead of $\pm 1$) outputs.

This architecture also requires us to assume that the set $S$ of size $k$ in the $(n, k)$-parity problem is selected such that exactly one index belongs to each disjoint partition, that is, for all $i \in [k]$, $S \cap P_i = 1$. We refer to this problem as the $(n, k)$-disjoint parity problem. Note that there are still $(n')^k = (n/k)^k$ different possibilities for set $S$ under this restriction. For fixed $k$, these represent a constant fraction of the $\binom{n}{k} \approx (ne/k)^k$ (by Stirling's approximation) possibilities for $S$ in the general non-disjoint case.

Consider training a disjoint-PolyNet w.r.t. the correlation loss. Without loss of generality, assume that each relevant coordinate in $S$ is the first element $P_i$. Then, the population gradient is non-zero only at indices $i \in S$:

$$g_i(w_{1:k}) = \mathbb{E}\left[\nabla_{w_i} \ell(f(x; w_{1:k}), y)\right] = -\mathbb{E}\left[y \left(\prod_{j \neq i} \langle w_j, x_{P_j} \rangle\right) x_{P_i}\right] = -\left(\prod_{j \neq i} w_{j,1}\right) e_1.$$

This allows us to analyze the gradient flow dynamics of the disjoint-PolyNet, without needing to

*We assume for simplicity that $n$ is divisible by $k$.

establish Fourier gaps. For each $i \in [k]$, in this section we treat $w_i$ as a function from $\mathbb{R}_{\geq 0} \to \mathbb{R}^{n'}$ which satisfies the following differential equation: $\dot{w}_i = -g_i(w_{1:k}(t))$. For clarity of exposition, assume all-ones initialization.* Then, all of the relevant weights $\{w_{i,1} : i \in [k]\}$ follow the same trajectory. By analyzing the resulting differential equations, we can formally exhibit "phase transition"-like behavior in the fully deterministic gradient flow setting.

**Theorem 5.7** (Loss plateau for gradient flow on disjoint-PolyNets)**.** *Suppose $k \geq 3$. Let $T(\varepsilon)$ denote the smallest time at which the error is at most $\varepsilon$. Then,*

$$\frac{T(0.49)}{T(0)} \geq 1 - O\left((n')^{1-k/2}\right).$$

Informally, the network takes much longer to reach slightly-better-than-trivial accuracy than it takes to go from slightly better than trivial to perfect accuracy. Returning to discrete time, we also analyze the trajectory of disjoint-PolyNets trained with online SGD at any batch size, confirming that a neural network can learn $k$-sparse disjoint parities within $n^{O(k)}$ iterations.

**Theorem 5.8** (SGD on disjoint-PolyNets learns disjoint parities)**.** *Suppose we train a disjoint-PolyNet, initialized as above, with online SGD. Then there exists an adaptive learning rate schedule such that for any $\varepsilon > 0$, with probability 0.99, the error falls below $\varepsilon$ within $\tilde{O}\left((n')^{(2k-1)} \log(1/\varepsilon)\right)$ steps.*

Extended versions of these theorems, along with proofs, can be found in Appendix B.2.3.

## 5.5 Hidden progress: discussion and additional experiments

So far, we have shown that sparse parity learning provides an idealized setting in which neural networks successfully learn sparse combinatorial features, with a mechanism of continual progress

*Results for Bernoulli and Gaussian initializations are similar, and can be found in the appendix.

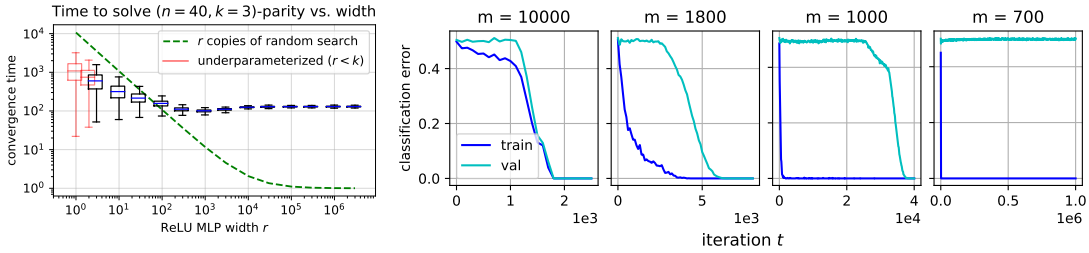



**Figure 5.4:** Parity as a sandbox for understanding the effects of model size and dataset size. *Left:* Success times vs. network width $r$ on a fixed $(40, 3)$-parity task: in accordance with the theory, parallelization experiences diminishing returns (unlike expected success times for random search, shown in green). Underparameterized models ($r = 1, 2$) were considered successful upon reaching $55\%$ accuracy. *Right:* Training curves where only the sample size $m$ is varied. The two center panels display *"grokking"*: a large gap between the time to zero train error vs. zero test error.

hiding behind discontinuous training curves. In this section, we outline preliminary explorations on a broader range of interesting phenomena which arise in this setting. Details are provided in Appendix B.3, while more systematic investigations are deferred to future work.

Hidden progress measures for learning parities. The theoretical and (black-box) empirical results suggest that SGD does *not* learn parities via the memoryless process of random exhaustive search. This suggests the existence of *progress measures*: scalar quantities which are functions of the training algorithm's state (i.e. the model weights $w_t$) and are predictive of the time to successful convergence. We provide some *white-box* investigations which further support the hypothesis of hidden progress, by examining the gradual improvement in quantities other than the training loss. In Appendix B.3.1, we directly plot the Fourier gaps of the population gradient, as a function of $t$, finding that they are large (within a small constant factor of those of majority) in practice. In Figure 5.3 and Appendix B.3.3, we examine the weight movement norm $\rho(w_{0:t}) := \|w_t - w_0\|_\infty$ to reveal hidden progress, motivated by the fact that $w_t - w_0$ is a linearized estimate for the initial population gradient.

Roles of overparameterization vs. *oversampling*. An interesting consequence of our analysis is that it illuminates scaling behaviors with respect to a third fundamental resource parameter: *model size*, which we study in terms of network width $r$. If SGD operated by a "random search" mechanism, one would expect width to provide a parallel speedup. Instead, we find that SGD sequentially amplifies progress. The sharp lower tails in Figure 5.2 *(left)* imply that running $r$ identical copies of SGD does *not* give $(1/r)\times$ speedups; more directly, in Appendix B.3.4 (previewed in Figure 5.4 *(left)*), we find that convergence times for sparse parities empirically plateau at large model sizes.

Emergence of grokking in the finite-sample (multi-pass) setting. Our main results are presented in the *online learning* setting (fresh minibatches from $\mathcal{D}_S$ at each iteration). While this mitigates the confounding factor of overfitting, it couples the resources of training time and independent samples in a suboptimal way, due to the computational-statistical gap for parity learning. In Appendix B.3.5, we find empirically that minibatch SGD (with weight decay) can learn sparse parities, even with smaller sample sizes $m \ll n^k$. We reliably observe the *grokking* phenomenon (Power et al., 2021): an initial overfitting phase, then a *delayed* phase transition in the generalization error; see the two center panels of Figure 5.4 *(right)*. These results complement and corroborate the findings of Nanda & Lieberum (2022), who analyze the hidden progress of Transformers trained on arithmetic tasks (a setting which also exhibits grokking).

Deeper networks. It is a significant challenge (and generally outside the scope of this work) to understand the interactions between network *depth* and computational/statistical efficiency. In Appendix B.3.7, we show that learning parities with deeper polynomial-activation MLPs comprises a simple counterexample to the *"deep only works if shallow is good"* principle of Malach & Shalev-Shwartz (2019): a deep network can get near-perfect accuracy, even when greedy layer-wise training

(e.g. (Belilovsky et al., 2019)) cannot beat trivial performance. By providing positive theory and empirics which elude these simplified explanations of SGD, we hope to point the way to a more complete understanding of learning dynamics in the challenging cases where no apparent progress is made for extended periods of time.

## 5.6 Conclusion

This work puts forward sparse parity learning as an elementary test case to explore the puzzling features of the role of computational (as opposed to statistical) resources in deep learning. In particular, we have shown that a variety of neural architectures solve this combinatorial search problem, with a number of computational steps nearly matching the sparsity-dependent SQ lower bound. Furthermore, we have shown that despite abrupt phase transitions in the loss and accuracy curves, SGD works by gradually amplifying the sparse features "under the hood".

Even in this simple setting, there are several open experimental and theoretical questions. This work largely focuses on the online learning case, which couples training iterations with fresh i.i.d. samples. We believe it would be instructive to investigate parity learning when the three resources of samples, time, and model size are scaled separately. Some very preliminary findings along these lines are presented in Section 5.3. It is an open problem to extend our theoretical results to the small-batch setting, as well as to the full range of architectures and losses in our experiments. Resolving these questions would require a better understanding of the anti-concentration behavior of Boolean Fourier coefficients, which is much less studied than the analogous concentration phenomena.

Another important follow-up direction is understanding the extent to which these insights extend from parity learning to more complex (including real-world) combinatorial problem settings, as well as the extent to which non-synthetic tasks (in, e.g., natural language processing and program synthesis) embed within them parity-like subtasks of exhaustive combinatorial search. We hope that

our results will lead to further progress towards understanding and improving the optimization dynamics behind the recent slew of dramatic empirical successes of deep learning in these types of domains.

*This chapter is based on "Pareto Frontiers in Deep Feature Learning: Data, Compute, Width, and Luck" (Edelman et al., 2024a), written in collaboration with Surbhi Goel, Sham Kakade, Eran Malach, and Cyril Zhang.*

# 6

# Pareto Frontiers

In modern deep learning, algorithmic choices (such as width, depth, and learning rate) are known to modulate nuanced resource tradeoffs. This work investigates how these complexities necessarily arise for feature learning in the presence of computational-statistical gaps. We begin by considering offline sparse parity learning, a supervised classification problem which admits a statistical query lower bound for gradient-based training of a multilayer perceptron. This lower bound can be interpreted as a *multi-resource tradeoff frontier*: successful learning can only occur if one is sufficiently

rich (large model), knowledgeable (large dataset), patient (many training iterations), or lucky (many random guesses). We show, theoretically and experimentally, that sparse initialization and increasing network width yield significant improvements in sample efficiency in this setting. Here, width plays the role of parallel search: it amplifies the probability of finding "lottery ticket" neurons, which learn sparse features more sample-efficiently. Finally, we show that the synthetic sparse parity task can be useful as a proxy for real problems requiring axis-aligned feature learning. We demonstrate improved sample efficiency on tabular classification benchmarks by using wide, sparsely-initialized MLP models; these networks sometimes outperform tuned random forests.

## 6.1 Introduction

Algorithm design in deep learning can appear to be more like "hacking" than an engineering practice. Numerous architectural choices and training heuristics can affect various performance criteria and resource costs in unpredictable ways. Moreover, it is understood that these multifarious hyperparameters all interact with each other; as a result, the task of finding the "best" deep learning algorithm for a particular scenario is foremost empirically-driven. When this delicate balance of considerations is achieved (i.e. when deep learning works well), learning is enabled by phenomena which cannot be explained by statistics or optimization in isolation. It is natural to ask: *is this heterogeneity of methods and mechanisms necessary?*

This work studies a single synthetic binary classification task in which the above complications are recognizable, and, in fact, *provable*. This is the problem of offline (i.e. small-sample) sparse parity learning: identify a $k$-way multiplicative interaction between $n$ Boolean variables, given $m$ random examples. We begin by interpreting the standard statistical query lower bound for this problem as a *multi-resource tradeoff frontier* for deep learning, balancing between the heterogeneous resources of dataset size, network size, number of iterations, and success probability. We show that in different

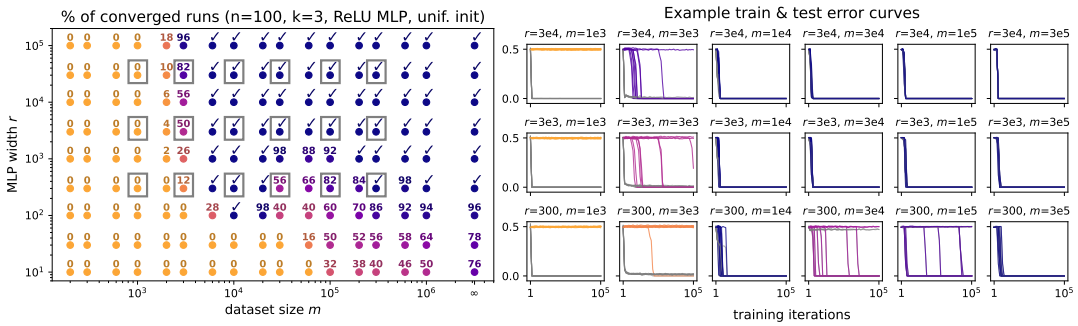



**Figure 6.1: "More data or larger network?"** Effects of jointly scaling the dataset and model sizes, for 2-layer MLPs trained to learn sparse parities. *Left:* A success frontier, where the computational challenge of feature learning can be surmounted by increasing sample size *or* model size. Each point shows the percentage of successful training runs (out of 50 trials) for each (dataset size $m$, width $r$); $(\checkmark)$ denotes 100% successful trials. *Right:* 10 example training curves (gray = training error; colored = test error) for each of the boxed $(m, r)$ pairs. See main text and Section 6.4 for further details.

regimes of simultaneous resource constraints (data, parallel computation, sequential computation, and random trials), the standard algorithmic choices in deep learning can succeed by diverse and entangled mechanisms. Specifically, our contributions are as follows:

MULTI-RESOURCE LOWER AND UPPER BOUNDS. We formulate a "data × width × time × luck" lower bound for offline sparse parity learning with feedforward neural nets (Theorem 6.3). This barrier arises from the classic statistical query (SQ) lower bound for this problem. We show that under different resource constraints, the tractability of learning can be "bought" with varied mixtures of these resources. In particular, in Theorems 6.4 and 6.5 we prove that by tuning the width and initialization scheme, we can populate this frontier with a spectrum of successful models ranging from narrow networks requiring many training samples to sample-efficient networks requiring many neurons, as summarized by the following informal theorem statement:

*Informal Theorem* 6.1. Consider the problem of learning $(n, k)$-parities from $m$ i.i.d. samples with a 2-layer width-$r$ ReLU MLP, whose first-layer neurons are initialized with sparsity $s$. After $O_n(1)$ steps of gradient descent, the $k$ relevant coordinates are identified with probability 0.99 when *(1)*

$s > \Omega(k), r = \Theta((n/s)^k)$ and $m = \Theta(n^2(s/k)^{k-1})$, and when *(2)* $s < k, r = \Theta((n/k)^s)$ and $m = \Theta((n/k)^{k-s-1})$.

Intuitively, this analysis reveals a feature learning mechanism by which overparameterization (i.e. large network width) plays a role of parallel search over randomized subnetworks. Each individual hidden-layer neuron has its own sample complexity for identifying the relevant coordinates, based on its *Fourier gap* (Barak et al., 2022) at initialization. Trained with parallel gradient updates, the full network implicitly acts as an ensemble model over these neurons, whose overall sample complexity is determined by the "winning lottery tickets" (Frankle & Carbin, 2018) (i.e. the lucky neurons initialized to have the lowest sample complexity). This departs significantly from the *neural tangent kernel* (Jacot et al., 2018) regime of function approximation with wide networks, in which overparameterization *removes* data-dependent feature selection (rather than parallelizing it across subnetworks).

Empirical study of neural nets' statistical thresholds for sparse parity learning. We corroborate the theoretical analysis with a systematic empirical study of offline sparse parity learning using SGD on MLPs, demonstrating some of the (perhaps) counterintuitive effects of width, data, and initialization. For example, Figure 6.1 highlights our empirical investigation into the interactions between data, width, and success probability. The left figure shows the fractions of successful training runs as a function of dataset size (x-axis) and width (y-axis). Roughly, we see a "success frontier", where having a larger width can be traded off with smaller sample sizes. The right figure depicts some training curves (for various widths and sample sizes). Grokking (Power et al., 2021; Liu et al., 2023) (discontinuous and delayed generalization behavior induced by optimization dynamics) is evident in some of these figures.

"Parity2real" transfer of algorithmic improvements. It is often observed that deep learning methods underperform tree-based methods (e.g. random forests) on tabular datasets, par-

ticularly those where the target function depends on a few of the input features in a potentially non-smooth manner; see Grinsztajn et al. (2022) for a recent discussion. Motivated by our findings in the synthetic parity setting (and the observation that the sparse parity task possesses a number of the same properties of these problematic real-world datasets), we then turn to experimentally determine the extent to which our findings also hold for real tabular data. We evaluate MLPs of various depths and initialization sparsities on 16 tabular classification tasks, which were standardized by Grinsztajn et al. (2022) to compare neural vs. tree-based methods. Figure 6.3 shows that wider networks and sparse initialization yield improved performance, as in the parity setting. In some cases, our MLPs outperform tuned random forests.

### 6.1.1 Related work

In the nascent empirical science of large-scale deep learning, *scaling laws* (Kaplan et al., 2020; Hoffmann et al., 2022) have been shown to extrapolate model performance with remarkable consistency, revealing flexible tradeoffs and Pareto frontiers between the heterogeneous resources of data and computation. The present work reveals that in the simple synthetic setting of parity learning, the same intricacies can be studied theoretically and experimentally. In particular, viewing *model size* $\times$ *training iterations* $\times$ *random restarts* as a single *"total FLOPs"* resource, our study explains why data $\times$ compute can be a necessary and sufficient resource, through the lens of SQ complexity.

Analyses of deep feature learning. Formally characterizing the representation learning mechanisms of neural networks is a core research program of deep learning theory. Many recent works have analyzed gradient-based feature learning (Wei et al., 2019a; Barak et al., 2022; Zhenmei et al., 2022; Abbe et al., 2022a; Damian et al., 2022; Telgarsky, 2022), escaping the "lazy" neural tangent kernel (NTK) regime (Jacot et al., 2018; Chizat et al., 2019), in which features are fixed at initialization.

Learning parities with neural networks. The XOR function has been studied as an elementary challenging example since the dawn of artificial neural networks (Minsky & Papert, 1969), and has been revisited at various times: e.g. neural cryptography (Rosen-Zvi et al., 2002); learning interactions via hints in the input distribution (Daniely & Malach, 2020); a hard case for self-attention architectures (Hahn, 2020). Closest to this work, Barak et al. (2022) find that in the case of *online* (infinite-data) parity learning, SGD provides a feature learning signal for a *single neuron*, and can thus converge at a near-optimal computational rate for non-overparameterized networks. They note computational-statistical tradeoffs and grokking in the offline setting, which we address systematically. Merrill et al. (2023) examine the same problem setting empirically, investigating a mechanism of competing sparse and dense sub-networks. Abbe et al. (2023) provide evidence that the time complexity required by an MLP to learn an arbitrary sparse Boolean function is governed by the largest "leap" in the *staircase* of its monomials, each of which is a sparse parity. Telgarsky (2022) gives a margin-based analysis of gradient flow on a two-layer neural network that achieves improved sample complexity ($\tilde{O}(n/\varepsilon)$) for the 2-sparse parity problem, at the cost of exponential width.

Neural nets and axis-aligned tabular data. Decision tree ensembles such as random forests (Breiman, 2001) and XGBoost (Chen & Guestrin, 2016) remain more popular among practitioners than neural networks on tabular data (Kaggle, 2021), despite many recent attempts to design specialized deep learning methods (Borisov et al., 2022). Some of these employ sparse networks (Yang et al., 2022b; Lutz et al., 2022) similar to those considered in our theory and experiments.

We refer the reader to Appendix C.1 for additional related work.

## 6.2 Background

### 6.2.1 Parities, and parity learning algorithms

A light bulb is controlled by $k$ out of $n$ binary switches; each of the $k$ influential switches can toggle the light bulb's state from any configuration. The task in parity learning is to identify the subset of $k$ important switches, given access to $m$ i.i.d. uniform samples of the state of all $n$ switches. Formally, for any $1 \leq k \leq n$ and $S \subseteq [n]$ such that $|S| = k$, the parity function $\chi_S : \{\pm 1\}^n \to \pm 1$ is defined as $\chi_S(x_{1:n}) := \prod_{i \in S} x_i$.* The $(n, k)$-parity learning problem is to identify $S$ from samples $(x \sim \text{Unif}(\{\pm 1\}^n, y = \chi_S(x)))$; i.e. output a classifier with 100% accuracy on this distribution, without prior knowledge of $S$.

This problem has a rich history in theoretical computer science, information theory, and cryptography. There are several pertinent ways to think about the fundamental role of parity learning:

(i) **Monomial basis elements:** A $k$-sparse parity is a degree-$k$ monomial, and thus the analogue of a Fourier coefficient with "frequency" $k$ in the harmonic analysis of Boolean functions (O'Donnell, 2014). Parities form an important basis for polynomial learning algorithms (e.g. Andoni et al. (2014)).

(ii) **Computational hardness:** There is a widely conjectured *computational-statistical gap* for this problem (Applebaum et al., 2009, 2010), which has been proven in restricted models such as SQ (Kearns, 1998) and streaming (Kol et al., 2017) algorithms. The statistical limit is $\Theta(\log(\text{number of possibilities for } S)) = \Theta(k \log n)$ samples, but the amount of computation needed (for an algorithm that can tolerate an $O(1)$ fraction of noise) is believed to scale as $\Omega(n^{ck})$ for some constant $c > 0$ – i.e., there is no significant improvement over trying all subsets.

*Equivalently, parity can be represented as a function of a bit string $b \in \{0,1\}^n$, which computes the XOR of the influential subset of indices $S$: $\chi_S(b_{1:n}) := \oplus_{i \in S} b_i$.

(iii) **Feature learning:** This setting captures the learning of a concept which depends jointly on multiple attributes of the inputs, where the lower-order interactions (i.e. correlations with degree-$k' < k$ parities) give no information about $S$. Intuitively, samples from the sparse parity distribution look like random noise until the learner has identified the sparse interaction. This notion of feature learning complexity is also captured by the more general *information exponent* (Arous et al., 2021).

### 6.2.2 Notation for neural networks

For single-output regression, a 2-layer multi-layer perceptron (MLP) is a function class, parameterized by a matrix $W \in \mathbb{R}^{r \times n}$, vectors $v, b \in \mathbb{R}^r$ and scalar $\beta \in \mathbb{R}$, defined by

$$\widehat{y} : x \mapsto v^\top \sigma(Wx + b) + \beta,$$

where $\sigma$ is an activation function, usually a scalar function applied identically to each input. For rows $w_i$ of $W$, the intermediate variables $\sigma(w_i^\top x + b_i)$ are thought of as *hidden layer* activations or *neurons*. The number of parallel neurons is often called the *width*.

## 6.3 Theory

In this section we theoretically study the interactions between data, width, time, and luck on the sparse parity problem. We begin by rehashing Statistical Query (SQ) lower bounds for parity learning in the context of gradient-based optimization, showing that without sufficient resources sparse parities cannot be learned. Then, we prove upper bounds showing that parity learning is possible with correctly scaling either width (keeping sample size small), sample size (keeping width small), or a mixture of the two.

Statistical query algorithms. A seminal work by Kearns (1998) introduced the statistical query (SQ) algorithm framework, which provides a means of analyzing noise-tolerant algorithms. Unlike traditional learning algorithms, SQ algorithms lack access to individual examples but can instead make queries to an *SQ oracle*, which responds with noisy estimates of the queries over the population. Notably, many common learning algorithms, including noisy variants of gradient descent, can be implemented within this framework. While SQ learning offers robust guarantees for learning in the presence of noise, there exist certain problems that are learnable from examples but not efficiently learnable from statistical queries (Blum et al., 1994, 2003). One notable example of such a problem is the parity learning problem, which possesses an SQ lower bound. This lower bound can be leveraged to demonstrate the computational hardness of learning parities with gradient-based algorithms (e.g., Shalev-Shwartz et al. (2017)).

### 6.3.1 Lower bound: a multi-resource hardness frontier

We show a version of the SQ lower bound in this section. Assume we optimize some model $h_\theta$, where $\theta \in \mathbb{R}^r$ (i.e., $r$ parameters). Let $\ell$ be some loss function satisfying: $\ell'(\hat{y}, y) = -y + \ell_0(\hat{y})$; for example, one can choose $\ell_2(\hat{y}, y) = \frac{1}{2}(y - \hat{y})^2$. Fix some hypothesis $h$, target $f$, a sample $\mathcal{S} \subseteq \{\pm 1\}^n$ and distribution $\mathcal{D}$ over $\{\pm 1\}^n$. We denote the empirical loss by $L_{\mathcal{S}}(h, f) = \frac{1}{|\mathcal{S}|} \sum_{\mathbf{x} \in \mathcal{S}} \ell(h(\mathbf{x}), f(\mathbf{x}))$ and the population loss by $L_{\mathcal{D}}(h, f) := \mathbb{E}_{\mathbf{x} \sim \mathcal{D}} [\ell(h(\mathbf{x}), f(\mathbf{x}))]$.

We consider SGD updates of the form:

$$\theta_{t+1} = \theta_t - \eta_t \left( \nabla_\theta \left( L_{\mathcal{S}_t}(h_{\theta_t}, f) + R(\theta_t) \right) + \xi_t \right),$$

for some sample $\mathcal{S}_t$, step size $\eta_t$, regularizer $R(\cdot)$, and adversarial noise $\xi_t \in [-\tau, \tau]^r$. For normalization, we assume that $\|\nabla h_\theta(\mathbf{x})\|_\infty \leq 1$ for all $\theta$ and $\mathbf{x} \in \mathcal{X}$.

Denote by 0 the constant function, mapping all inputs to 0. Let $\theta_t^\star$ be the following trajectory of

SGD that is independent of the target. Define $\theta_t^\star$ recursively s.t. $\theta_0^\star = \theta_0$ and

$$\theta_{t+1}^\star = \theta_t^\star - \eta_t \nabla_\theta \left( L_{\mathcal{D}}(h_{\theta_t^\star}, 0) + R(\theta_t^\star) \right)$$

**Assumption 6.2** (Bounded error for gradient estimator)**.** For all $t$, suppose that

$$\left\| \nabla L_{\mathcal{S}_t}(h_{\theta_t^\star}, 0) - \nabla L_{\mathcal{D}}(h_{\theta_t^\star}, 0) \right\|_\infty \leq \tau/2.$$

*Remark:* If $m = \tilde{O}(1/\tau^2)$,* Assumption 6.2 is satisfied w.h.p. for: 1) $\mathcal{S}_t \sim \mathcal{D}^m$ ("online" SGD), 2) $\mathcal{S}_t = \mathcal{S}$ for some $\mathcal{S} \sim \mathcal{D}^m$ ("offline" GD) and 3) $\mathcal{S}_t$ is a batch of size $m$ sampled uniformly at random from $\mathcal{S}$, where $\mathcal{S} \sim \mathcal{D}^M$ and $M \geq m$ ("offline" SGD). Indeed, in all these cases we have $\mathbb{E}_{\mathcal{S}_t}\left[\nabla L_{\mathcal{S}_t}(h_{\theta_t^\star}, 0)\right] = \nabla L_{\mathcal{D}}(h_{\theta_t^\star}, 0)$, and the above follows from standard concentration bounds.

The lower bound in this section uses standard statistical query arguments to show that gradient-based algorithms, *without sufficient resources*, will fail to learn $(n, k)$-parities. We therefore start by stating the four types of resources that impact learnability:

- The *number of parameters* $r$ – equivalently, the number of parallel "queries".
- The *number of gradient updates* $T$ – i.e. the serial running time of the training algorithm.
- The *gradient precision* $\tau$ – i.e. how close the empirical gradient is to the population gradient. As discussed above, $\tilde{O}(1/\tau^2)$ samples suffice to obtain such an estimator.
- The probability of success $\delta$.

The following theorem ties these four resources together, showing that without a sufficient allocation of these resources, gradient descent will fail to learn:

*We use the notation $\tilde{O}$ to hide logarithmic factors.

**Proposition 6.3.** *Assume that $\theta_0$ is randomly drawn from some distribution. For every $r, T, \delta, \tau > 0$, if $\frac{rT}{\tau^2\delta} \leq \frac{1}{2}\binom{n}{k}$, then there exists some $(n,k)$-parity s.t. with probability at least $1-\delta$ over the choice of $\theta_0$, the first $T$ iterates of SGD are statistically independent of the target function.*

The proof follows standard SQ lower bound arguments (Kearns, 1998; Feldman, 2008), and for completeness is given in Appendix C.2. The core idea of the proof is the observation that any gradient step has very little correlation (roughly $n^{-k}$) with most sparse-parity functions, and so there exists a parity function that has small correlation with all steps. In this case, the noise can force the gradient iterates to follow the trajectory $\theta_1^\star, \ldots, \theta_T^\star$, which is independent of the true target. We note that, while the focus of this paper is on sparse parities, similar analysis applies for a broader class of functions that are characterized by large SQ dimension (see Blum et al. (1994)).

Observe that there are various ways for an algorithm to escape the lower bound of Theorem 6.3. We can scale a single resource with $\binom{n}{k}$, keeping the rest small, e.g. by training a network of size $n^k$, or using a sample size of size $n^k$. Crucially, we note the possibility of *interpolating* between these extremes: one can spread this homogenized "cost" across multiple resources, by (e.g.) training a network of size $n^{k/2}$ using $n^{k/2}$ samples. In the next section, we show how neural networks can be tailored to solve the task in these interpolated resource scaling regimes.

### 6.3.2 Upper bounds: many ways to trade off the terms

As a warmup, we discuss some simple SQ algorithms that succeed in learning parities by properly scaling the different resources. First, consider deterministic exhaustive search, which computes the error of all possible $(n,k)$-parities, choosing the one with smallest error. This can be done with constant $\tau$ (thus, sample complexity logarithmic in $n$), but takes $T = \Theta(n^k)$ time. Since querying different parities can be done in parallel, we can reduce the number of steps $T$ by increasing the number of parallel queries $r$. Alternatively, it is possible to query only a randomly selected subset of parities,

which reduces the overall number of queries but increases the probability of failure.

The above algorithms give a rough understanding of the frontier of algorithms that succeed at learning parities. However, at first glance, they do not seem to reflect algorithms used in deep learning, and are specialized to the parity problem. In this section, we will explore the ability of neural networks to achieve similar tradeoffs between the different resources. In particular, we focus on the interaction between sample complexity and network size, establishing learning guarantees with *interpolatable mixtures* of these resources.

Before introducing our main positive theoretical results, we discuss some prior theoretical results on learning with neural networks, and their limitations in the context of learning parities. Positive results on learning with neural networks can generally be classified into two categories: those that reduce the problem to convex learning of linear predictors over a predefined set of features (e.g. the NTK), and those that involve neural networks departing from the kernel regime by modifying the fixed features of the initialization, known as the feature learning regime.

Kernel regime. When neural networks trained with gradient descent stay close to their initial weights, optimization behaves like kernel regression on the neural tangent kernel (Jacot et al., 2018; Du et al., 2018): the resulting function is approximately of the form $\mathbf{x} \mapsto \langle \psi(\mathbf{x}), \mathbf{w} \rangle$, where $\psi$ is a *data-independent* infinite-dimensional embedding of the input, and $\mathbf{w}$ is some weighting of the features. However, it has been shown that the NTK (more generally, any fixed kernel) cannot achieve low $\ell_2$ error on the $(n, k)$-parity problem, unless the sample complexity grows as $\Omega(n^k)$ (see Kamath et al. (2020)). Thus, no matter how we scale the network size or training time, neural networks trained in the NTK regime cannot learn parities with low sample complexity, and thus do not enjoy the flexibility of resource allocation discussed above.

Feature learning regime. Due to the limitation of neural networks trained in the kernel regime, some works study learning in the "rich" regime, quantifying how hidden-layer features adapt to the data. Among these, Barak et al. (2022) analyze a feature learning mechanism requiring exceptionally small network width: SGD on 2-layer MLPs can solve the *online* sparse parity learning problem with network width *independent* of $n$ (dependent only on $k$), at the expense of requiring a suboptimal ($\approx n^k$) number of examples. This mechanism is *Fourier gap amplification*, by which SGD through a *single neuron* $\sigma(w^\top x)$ can perform feature selection in this setting, via exploiting a small gap between the relevant and irrelevant coordinates in the population gradient. The proof of Theorem 6.4 below relies on a similar analysis, extended to the offline regime (i.e., multiple passes over a dataset of limited size).

### "Data × model size" success frontier for sparsely-initialized MLPs

In this section, we analyze a 2-layer MLP with ReLU ($\sigma(x) = \max(0, x)$) activation, trained with batch ("offline") gradient-descent over a sample $\mathcal{S}$ with $\ell_2$-regularized updates*:

$$\theta^{(t+1)} = (1 - \lambda^{(t)})\theta^{(t)} - \eta^{(t)} \nabla L_{\mathcal{S}}(h_{\theta^{(t)}})$$

We allow learning rates $\eta$, and weight decay coefficients $\lambda$ to differ between layers and iterations. For simplicity, we analyze the case where no additional noise is added to each update; however, we believe that similar results can be obtained in the noisy case (e.g., using the techniques in Feldman et al. (2017)). Finally, we focus on ReLU networks with $s$-sparse initialization of the first layer: every weight $\mathbf{w}_i$ has $s$ randomly chosen coordinates set to 1, and the rest set to 0. Note that after initialization, all of the network's weights are allowed to move, so sparsity is not necessarily preserved during training.

*For simplicity, we do not assume adversarial noise in the gradients as in the lower bound. However, similar results can be shown under bounded noise.

Over-sparse initialization ($s > \Omega(k)$). The following theorem demonstrates a "data$\times$ width" success frontier when learning $(n, k)$-parities with sparsely initialized ReLU MLPs at sparsity levels $s > \Omega(k)$.

**Theorem 6.4.** *Let $k$ be an even integer, and $\varepsilon \in (0, 1/2)$. Assume that $n \geq \Omega(1/\varepsilon^2)$. For constants $c_1, c_2, c_3, c_4$ depending only on $k$, choose the following: (1) sparsity level: $s \geq c_1/\varepsilon^2$, for some odd $s$, (2) width of the network: $r = c_2(n/s)^k$, (3) sample size: $m \geq c_3(s/k)^{k-1} n^2 \log n$, and (4) number of iterations: $T \geq c_4/\varepsilon^2$. Then, for every $(n, k)$-parity distribution $\mathcal{D}$, with probability at least $0.99$ over the random samples and initialization, gradient descent with these parameter settings returns a function $h_T$ s.t. $L_{\mathcal{D}}(h_T) \leq \varepsilon$.*

Intuitively, by varying the sparsity parameter in Theorem 6.4, we obtain a family of algorithms which smoothly interpolate between the *small-data/large-width* and *large-data/small-width* regimes of tractability. First, consider a sparsity level linear in $n$ (i.e. $s = \alpha \cdot n$). In this case, a small network (with width $r$ independent of the input dimension) is sufficient for solving the problem, but the sample size must be large ($\Omega(n^{k+1})$) for successful learning; this recovers the result of Barak et al. (2022). At the other extreme, if the sparsity is independent of $n$, the sample complexity grows only as $O(n^2 \log n)$*, but the requisite width becomes $\Omega(n^k)$.

*Proof sketch.* The proof of Theorem 6.4 relies on establishing a Fourier anti-concentration condition, separating the relevant (i.e. indices in $S$) and irrelevant weights in the initial population gradient, similarly as the main result in Barak et al. (2022). When we initialize an $s$-sparse neuron, there is a probability of $\gtrsim (s/n)^k$ that the subset of activated weights contains the "correct" subset $S$. In this case, to detect the subset $S$ via the Fourier gap, it is sufficient to observe $s^{k-1}$ examples instead of $n^{k-1}$. Initializing the neurons more sparsely makes it less probable to draw a *lucky* neuron, but

*We note that the additional $n^2$ factor in the sample complexity can be removed if we apply gradient truncation, thus allowing only a logarithmic dependence on $n$ in the small-sample case.

once we draw a lucky neuron, it requires fewer samples to find the right features. Thus, increasing the width reduces overall sample complexity, by sampling a large number of "lottery tickets".

Under-sparse initialization ($s < k$). The sparsity parameter can modulate similar data vs. width tradeoffs for feature learning in the "under-sparse" regime. We provide a partial analysis for this more challenging case, showing that one step of gradient descent can recover a correct subnetwork. Appendix C.2.3 discusses the mathematical obstructions towards obtaining an end-to-end guarantee of global convergence.

**Theorem 6.5.** *For even $k, s$, sparsity level $s < k$, network width $r = O((n/k)^s)$, and $\varepsilon$-perturbed[*] $s$-sparse random initialization scheme s.t. for every $(n, k)$-parity distribution $\mathcal{D}$, with probability at least 0.99 over the choice of sample and initialization after one step of batch gradient descent (with gradient clipping) with sample size $m = O((n/k)^{k-s-1})$ and appropriate learning rate, there is a subnetwork in the ReLU MLP that approximately computes the parity function.*

Here, the sample complexity can be improved by a factor of $n^s$, at the cost of requiring the width to be $n^s$ times larger. The proof of Theorem 6.5 relies on a novel analysis of improved Fourier gaps with "partial progress": intuitively, if a neuron is randomly initialized with a subset of the relevant indices $S$, it only needs to identify $k-s$ more coordinates, inheriting the improved sample complexity for the $(n-s, k-s)$-parity problem. Note that the probability of finding such a lucky neuron scales as $(n/k)^{-s}$, which governs how wide (number of lottery tickets) the network needs to be.

Remarks on the exact-sparsity regime. Our theoretical analyses do not extend straightforwardly to the case of $s = \Theta(k)$. Observe that if the value of $k$ is known, initializing a network of size

[*]For ease of analysis, we use a close variant of the sparse initialization scheme: $s$ coordinates out of the $n$ coordinates are chosen randomly and set to 1, and the rest of the coordinates are set to $\varepsilon < (n-s)^{-1}$. Without the small norm dense component in the initialization, the population gradient will be 0 at initialization.

$r = \widetilde{O}\left(n^k\right)$ with sparsity $s = k$ gives w.h.p. a subnetwork with good first-layer features at initialization. We believe that with a proper choice of regularization and training scheme, it is possible to show that such a network learns to select this subnetwork with low sample complexity. We leave the exact details of this construction and end-to-end proofs for future work.

Analogous results for dense initialization schemes? We believe that the principle of "parallel search with randomized per-subnetwork sample complexities" extends to other initialization schemes (including those more commonly used in practice), and leads to analogous success frontiers. To support this, our experiments investigate both sparse and uniform initialization, with qualitatively similar findings. For dense initializations, the mathematical challenge lies in analyzing the Fourier anti-concentration of general halfspaces (see the discussion and experiments in Appendix C.1 of (Barak et al., 2022)). The axis-aligned inductive biases imparted by sparse initialization may also be of independent practical interest.

## 6.4 Experiments

A high-level takeaway from Section 5.4 is that when a learning problem is computationally difficult but statistically easy, a complex frontier of resource tradeoffs can emerge; moreover, it is possible to interpolate between extremes along this frontier using ubiquitous algorithmic choices in deep learning, such as overparameterization, random initialization, and weight decay. In this section, we explore the nature of the frontier with an empirical lens—first with end-to-end sparse parity learning, then with natural tabular datasets.

### 6.4.1 Empirical Pareto frontiers for offline sparse parity learning

We launch a large-scale (∼200K GPU training runs) exploration of resource tradeoffs when training neural networks to solve the offline sparse parity problem. While Section 5.4 analyzes idealized

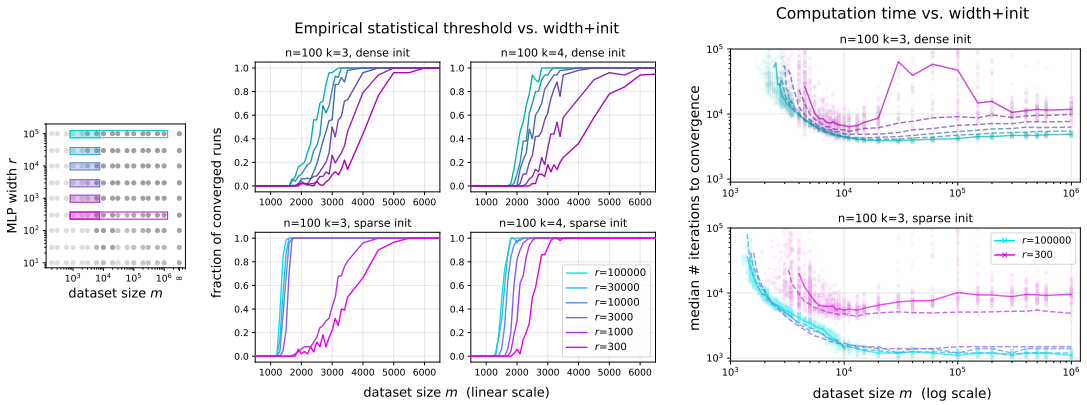



**Figure 6.2:** Zoomed-in views of the interactions between width, data, time, and luck. *Left:* Locations of these runs in the larger parameter space. *Center:* Success probability vs. dataset size. In accordance with our theory, **width buys luck, and improves end-to-end sample efficiency**, despite increasing the network's capacity. *Right:* Number of iterations to convergence vs. dataset size. We observe a **data vs. time tradeoff** in the grokking regime (at the edge of feasibility), as well as a **"sample-wise double descent"** performance drop with more data (which can also be seen in Figure 6.1, and disappears with larger widths). Comparing dense vs. sparse initializations (upper vs. lower plots; sparse inits are colored more brightly), we see computational and statistical benefits of the sparse initialization scheme from Section 6.3.2.

variants of SGD on MLPs which interpolate along the problem's resource tradeoff frontier, in this section we ask whether the same can be observed end-to-end with standard training and regularization.

On various instances of the $(n, k)$-sparse parity learning problem, we train a 2-layer MLP with identical hyperparameters, varying the network width $r \in \{10, 30, 100, \ldots, 10^5\}$ and the dataset size $m \in \{100, 200, 300, 600, 1000, \ldots, 10^6\}$. Alongside standard algorithmic choices, we consider one non-standard augmentation of SGD: the *under-sparse* initialization scheme from Section 6.3.2; we have proven that these give rise to "lottery ticket" neurons which learn the influential coordinates more sample-efficiently. Figure 6.1 (in the introduction) and Figure 6.2 illustrate our findings at a high level; details and additional discussion are in Appendix C.3.1). We list our key findings below:

(1) **A "success frontier": large width can compensate for small datasets.** We observe convergence and perfect generalization when $m \ll n^k$. In such regimes, which are far outside

the online setting considered by Barak et al. (2022), high-probability sample-efficient learning is enabled by large width. This can be seen in Figure 6.1 (left), and analogous plots in Appendix C.3.1.

(2) **Width is monotonically beneficial, and buys data, time, and luck.** In this setting, increasing the model size yields exclusively positive effects on success probability, sample efficiency, and the number of serial steps to convergence (see Figure 6.2). This is a striking example where end-to-end generalization behavior runs *opposite* to uniform convergence-based upper bounds, which predict that enlarging the model's capacity *worsens* generalization.

(3) **Sparse axis-aligned initialization buys data, time, and luck.** Used in conjunction with a wide network, we observe that a sparse, axis-aligned initialization scheme yields strong improvements on all of these axes; see Figure 6.2 (bottom row). In smaller hyperparameter sweeps, we find that $s = 2$ (i.e. initialize every hidden-layer neuron with a random 2-hot weight vector) works best.

(4) **Intriguing effects of dataset size.** As we vary the sample size $m$, we note two interesting phenomena; see Figure 6.2 (right). The first is grokking (Power et al., 2021), which has been previously documented in this setting (Barak et al., 2022; Merrill et al., 2023). This entails a *data vs. time* tradeoff: for small $m$ where learning is marginally feasible, optimization requires significantly more training iterations $T$. Our second observation is a "sample-wise double descent" (Nakkiran et al., 2021): success probability and convergence times can worsen with *increasing* data. Both of these effects are also evident in Figure 6.1).

Lottery ticket neurons. The above findings are consistent with the viewpoint taken by the theoretical analysis, where randomly-initialized SGD plays the role of *parallel search*, and a large width increases the number of random subnetworks available for this process—in particular, the

"winning lottery ticket" neurons, for which feature learning occurs more sample-efficiently. To provide further evidence that sparse subnetworks are responsible for learning the parities, we perform a smaller-scale study of network prunability in Appendix C.3.2.

### 6.4.2 Sample-efficient deep learning on natural tabular datasets

Sparse parity learning is a toy problem, in that it is defined by an idealized distribution, averting the ambiguities inherent in reasoning about real-life datasets. However, due to its provable hardness (Theorem 6.3, as well as the discussion in Section 6.2.1), it is a *maximally hard* toy problem in a rigorous sense*. In this section, we perform a preliminary investigation of how the empirical and algorithmic insights gleaned from Section 6.4.1 can be transferred to more realistic learning scenarios.

To this end, we use the benchmark assembled by Grinsztajn et al. (2022), a work which specifically investigates the performance gap between neural networks and tree-based classifiers (e.g. random forests, gradient-boosted trees), and includes a standardized suite of 16 classification benchmarks with numerical input features. The authors identify three common aspects of tabular† tasks which present difficulties for neural networks, especially vanilla MLPs:

(i) **The feature spaces are not rotationally invariant.** In state-of-the-art deep learning, MLPs are often tasked with function representation in rotation-invariant domains (token embedding spaces, convolutional channels, etc.).

(ii) **Many of the features are uninformative.** In order to generalize effectively, especially from a limited amount of data, it is essential to avoid overfitting to these features.

(iii) **There are meaningful high-frequency/non-smooth patterns in the target function.**

*Namely, its SQ dimension is equal to the number of hypotheses, which is what leads to Theorem 6.3.

†"Tabular data" refers to the catch-all term for data sources where each coordinate has a distinct semantic meaning which is consistent across points.

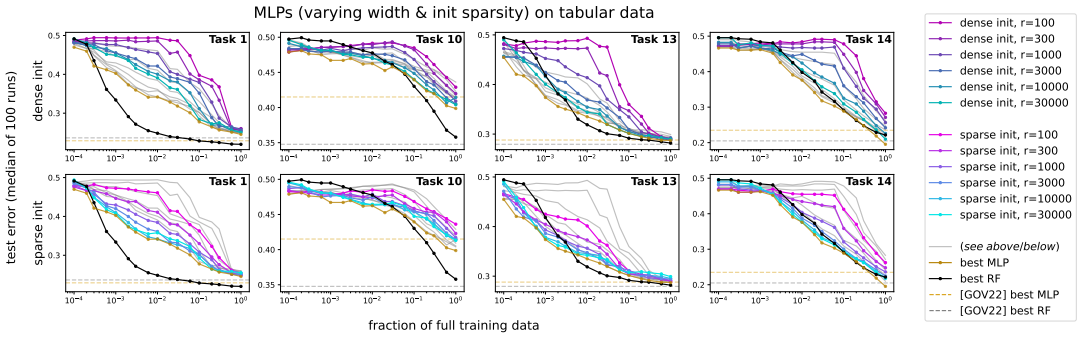



**Figure 6.3:** Analogous investigation of MLP width $r$ and sparse initialization for real-world tabular datasets (OpenML benchmarks assembled by Grinsztajn et al. (2022)), varying the dataset size $m$ via downsampling. Large width and sparse initialization tend to improve generalization, in accordance with the theory and experiments for synthetic parity tasks. In some settings, our best MLPs outperform tuned random forests. Dotted lines denote the test errors reported by Grinsztajn et al. (2022) of tuned MLPs and RFs on the full datasets. Full results on all 16 tasks are in Appendix C.3.3.

> Combined with property (i), decision tree-based methods (which typically split on axis-aligned features) can appear to have the ideal inductive bias for tabular modalities of data.

Noting that the sparse parity task possesses all three of the above qualities, we conduct a preliminary investigation on whether our empirical findings in the synthetic case carry over to natural tabular data. In order to study the impact of algorithmic choices (mainly width and sparse initialization) on sample efficiency, we create low-data problem instances by subsampling varying fractions of each dataset for training. Figure 6.3 provides a selection of our results. We note the following empirical findings, which are the tabular data counterparts of results (2) and (3) in Section 6.4.1:

(2T) **Wide networks generalize on small tabular datasets.** Like in the synthetic experiments, width yields nearly monotonic end-to-end benefits for learning. This suggests that the "parallel search + pruning" mechanisms analyzed in our paper are also at play in these settings. In some (but not all) cases, these MLPs perform competitively with tuned tree-based classifiers.

(3T) **Sparse axis-aligned initialization sometimes improves end-to-end performance.** This

effect is especially pronounced on datasets which are downsampled to be orders of magnitude smaller. We believe that this class of drop-in replacements for standard initialization merits further investigation, and may contribute to closing the remaining performance gap between deep learning and tree ensembles on small tabular datasets.

## 6.5 Conclusion

We have presented a theoretical and empirical study of offline sparse parity learning with neural networks; this is a provably hard problem which admits a multi-resource lower bound in the SQ model. We have shown that the lower bound can be surmounted using varied mixtures of these resources, which correspond to natural algorithmic choices and scaling axes in deep learning. By investigating how these choices influence the empirical "success frontier" for this hard synthetic problem, we have arrived at some promising improvements for MLP models of tabular data (namely, large width and sparse initialization). These preliminary experiments suggest that a more intensive, exhaustive study of algorithmic improvements for MLPs on tabular data has a chance of reaping significant rewards, perhaps even surpassing the performance of decision tree ensembles.

Broader impacts and limitations. The nature of this work is foundational; the aim of our theoretical and empirical investigations is to contribute to the fundamental understanding of neural feature learning, and the influences of scaling relevant resources. A key limitation is that our benchmarks on tabular data are only preliminary; it is a significant and perennial methodological challenge to devise fair and comprehensive comparisons between neural networks and tree-based learning paradigms.

*This chapter is based on "Feature Emergence via Margin Maximization: Case Studies in Algebraic Tasks" (Morwani et al., 2023b), written in collaboration with Depen Morwani, Costin-Andrei Oncescu, Rosie Zhao, and Sham Kakade.*

# 7

# Feature Emergence

Understanding the internal representations learned by neural networks is a cornerstone challenge in the science of machine learning. While there have been significant recent strides in some cases towards understanding *how* neural networks implement specific target functions, this paper explores a complementary question – *why* do networks arrive at particular computational strategies? Our inquiry focuses on the algebraic learning tasks of modular addition, sparse parities, and finite group operations. Our primary theoretical findings analytically characterize the features learned by stylized

neural networks for these algebraic tasks. Notably, our main technique demonstrates how the principle of margin maximization alone can be used to fully specify the features learned by the network. Specifically, we prove that the trained networks utilize Fourier features to perform modular addition and employ features corresponding to irreducible group-theoretic representations to perform compositions in general groups, aligning closely with the empirical observations of Nanda et al. (2023) and Chughtai et al. (2023). More generally, we hope our techniques can help to foster a deeper understanding of why neural networks adopt specific computational strategies.

## 7.1 Introduction

Opening the black box of neural networks has the potential to enable safer and more reliable deployments, justifications for model outputs, and clarity on how model behavior will be affected by changes in the input distribution. The research area of mechanistic interpretability (Olah et al., 2020; Elhage et al., 2021; Olsson et al., 2022; Elhage et al., 2022) aims to dissect individual trained neural networks in order to shed light on internal representations, identifying and interpreting subcircuits that contribute to the networks' functional behavior. Mechanistic interpretability analyses typically leave open the question of *why* the observed representations arise as a result of training.

Meanwhile, the theoretical literature on inductive biases in neural networks (Soudry et al., 2018; Shalev-Shwartz & Ben-David, 2014; Vardi, 2023) aims to derive general principles governing which solutions will be preferred by trained neural networks—in particular, in the presence of underspecification, where there are many distinct ways a network with a given architecture could perform well on the training data. Most work on inductive bias in deep learning is motivated by the question of understanding why networks generalize from their training data to unobserved test data. It can be non-obvious how to apply the results from this literature to understand what solution will be found when a particular architecture is trained on a particular type of dataset.

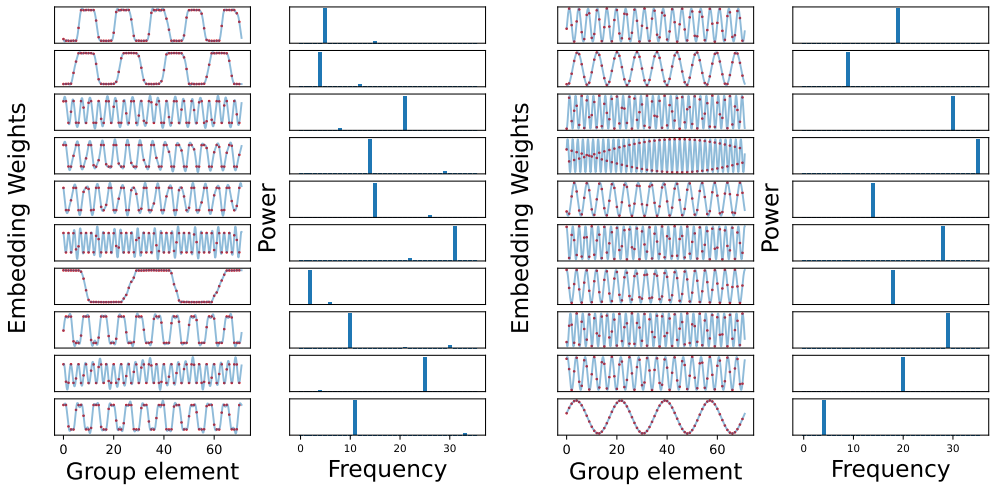
In this work, we show that the empirical findings of Nanda et al. (2023) and Chughtai et al. (2023), about the representations found by networks trained to perform finite group operations, can be analytically explained by the inductive bias of the regularized optimization trajectory towards *margin maximization*. Informally, the network maximizes the margin if it attains a given confidence level on all the points in the dataset, with the smallest total parameter norm possible. Perhaps surprisingly, the margin maximization property alone — typically used for the study of generalization — is sufficient to *comprehensively and precisely* characterize the richly structured features that are actually learned by neural networks in these settings. Let's begin by reviewing the case of learning modular addition with neural networks, first studied in Power et al. (2021) in their study of "grokking".

NANDA ET AL.'S STRIKING OBSERVATIONS. Nanda et al. (2023) investigated the problem of how neural networks learn modular addition (using a 1-layer transformer); they consider the problem of computing $a + b \bmod p$, where $p$ is a prime number. The findings were unexpected and intriguing: SGD not only reliably solves this problem (as originally seen in Power et al. (2021)) but also consistently learns to execute a particular algorithm, as illustrated by the learned embedding weights in Figure 7.1. This geometric algorithm simplifies the task to composing integer rotations around a circle [*].

The algorithm above fundamentally relies on the following identity: for any $a, b \in \mathbb{Z}_p$ and $k \in \mathbb{Z}_p \setminus \{0\}$,

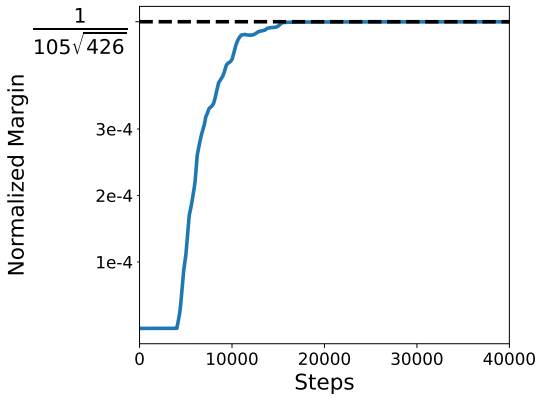
$$(a+b) \bmod p = \arg\max_{c \in \mathbb{Z}_p} \left\{ \cos\left( \frac{2\pi k(a+b-c)}{p} \right) \right\}.$$

[*]The algorithm identified by Nanda et al. (2023) can be seen as a real-valued implementation of the following procedure: Choose a fixed $k$. Embed $a \mapsto e^{2\pi i k a}$, $b \mapsto e^{2\pi i k b}$, representing rotations by $ka$ and $kb$. Multiply these (i.e. compose the rotations) to obtain $e^{2\pi i k(a+b)}$. Then, for each $c \in \mathbb{Z}_p$, multiply by $e^{-2\pi i k c}$ and take the real part to obtain the logit for $c$. Moreover, averaging the result over neurons with different frequencies $k$ results in destructive interference when $c \neq a + b$, accentuating the correct answer.




**(a)** ReLU activation

**(b)** Quadratic activation




**(c)** Normalized $L_{2,3}$ Margin

**Figure 7.1:** (a) Final trained embeddings and their Fourier power spectrum for a 1-hidden layer ReLU network trained on a mod-71 addition dataset with $L_2$ regularization. Each row corresponds to an arbitrary neuron from the trained network. The red dots represent the actual value of the weights, while the light blue interpolation is obtained by finding the function over the reals with the same Fourier spectrum as the weight vector. (b) Similar plot for 1-hidden layer quadratic activation, trained with $L_{2,3}$ regularization (Section 7.2) (c) For the quadratic activation, the network asymptotically reaches the maximum $L_{2,3}$ margin predicted by our analysis.

This identity also leads to other natural algorithms (still relying on sinusoidal features) that are generally implemented by neural networks, as shown in Zhong et al. (2023).

These findings prompt the question: why does the network consistently prefer such Fourier-based circuits, amidst other potential circuits capable of performing the same modular addition function?

Our Contributions.

- We formulate general techniques for analytically characterizing the maximum margin solutions for tasks exhibiting symmetry.
- For sufficiently wide one-hidden layer MLPs with quadratic activations, we use these techniques to characterize the structure of the weights of max-margin solutions for certain algebraic tasks including modular addition, sparse parities and general group operations.
- We empirically validate that neural networks trained using gradient descent with small regularization approach the maximum margin solution (Theorem 7.1), and the weights of trained networks match those predicted by our theory (Figure 7.1).

Our theorem for modular addition shows that Fourier features are indeed the global maximum margin solution:

*Informal Theorem* (Modular addition). Consider a single hidden layer neural network of width $m$ with $x^2$ activations trained on the modular addition task (modulo $p$). For $m \geq 4(p-1)$, any maximum margin solution for the full population dataset satisfies the following:

- For every neuron, there exists a frequency such that the Fourier spectra of the input and output weight vectors are supported only on that frequency.
- There exists at least one neuron of each frequency in the network.

Note that even with this activation function, there are solutions that fit all the data points, but where the weights do not exhibit any sparsity in Fourier space—see Appendix D.4 for an example construction. Such solutions, however, have lower margin and thus are not reached by training.

In the case of $k$-sparse parity learning with an $x^k$-activation network, we show margin maximization implies that the weights assigned to all relevant bits are of the same magnitude, and the sign pattern of the weights satisfies a certain condition.

For learning on the symmetric group (or other groups with real representations), we use the machinery of representation theory (Kosmann-Schwarzbach et al., 2010) to show that learned features correspond to the irreducible representations of the group, as observed by Chughtai et al. (2023).

Two closely related works to ours are Gromov (2023) and Bronstein et al. (2022). Gromov (2023) provides an analytic construction of various two-layer quadratic networks that can solve the modular addition task. The construction used in the proof of Theorem 7.7 is a special case of the given scheme. Bronstein et al. (2022) shows that all max margin solutions of a one-hidden-layer ReLU network (with fixed top weights) trained on read-once DNFs have neurons which align with clauses. However, their proof technique for characterizing max margin solutions is very different. For more details, refer to Appendix 5.1.2.

**Paper organization:** In section 6.1, we delineate our contributions and discuss a few related works. In section 7.2, we state preliminary definitions. In section 7.3, we sketch our theoretical methodology, and state general lemmas which will be applied in all three case studies. In sections 7.4, 7.5, and 7.6, we use the above lemmas to characterize the max margin features for the modular addition, sparse parity and group operation tasks respectively. We discuss and conclude the paper in section 5.5. Further related work, full proofs, hyperparameter choices, and additional experimental results can be found in the Appendix.

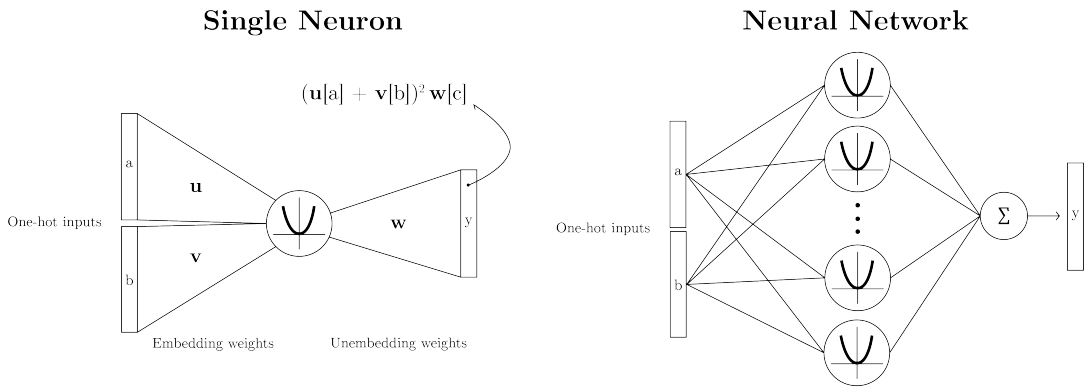



**Figure 7.2:** An illustration of an individual neuron $\varphi(\{u, v, w\}, a, b)$ (left) and the resulting one hidden layer neural network $f(\theta, a, b)$ (right) with quadratic activations.

## 7.2 Preliminaries

In this work, we will consider one-hidden layer neural networks with homogeneous polynomial activations, such as $x^2$, and no biases. The network output for a given input $x$ will be represented as $f(\theta, x)$, where $\theta \in \Theta$ represents the parameters of the neural network. The homogeneity constant of the network is defined as a constant $\nu$ such that for any scaling factor $\lambda > 0, f(\lambda\theta, x) = \lambda^\nu f(\theta, x)$ for all inputs $x$.

In the case of 1-hidden layer networks, $f$ can be further decomposed as: $f(\theta, x) = \sum_{i=1}^{m} \varphi(\omega_i, x)$, where $\theta = \{\omega_1, \ldots, \omega_m\}$, $\varphi$ represents an individual neuron within the network, and $\omega_i \in \Omega$ denotes the weights from the input to the $i^{\text{th}}$ neuron and from the neuron to the output. $\theta = \{\omega_1, \ldots, \omega_m\}$ is said to have directional support on $\Omega' \subseteq \Omega$ if for all $i \in \{1, \ldots, m\}$, either $\omega_i = 0$ or $\lambda_i \omega_i \in \Omega'$ for some $\lambda_i > 0$. In this work, we will be primarily concerned with networks that have homogeneous neurons, i.e, $\varphi(\lambda\omega_i, x) = \lambda^\nu \varphi(\omega_i, x)$ for any scaling constant $\lambda > 0$.

For Sections 7.4 and 7.6 corresponding to cyclic and general finite groups respectively, we will consider neural networks with quadratic activations (Figure 7.2). A single neuron will be repre-

sented as $\varphi(\{u, v, w\}, x^{(1)}, x^{(2)}) = (u^\top x^{(1)} + v^\top x^{(2)})^2 w$, where $u, v, w \in \mathbb{R}^d$ are the weights associated with a neuron and $x^{(1)}, x^{(2)} \in \mathbb{R}^d$ are the inputs provided to the network (note that $\varphi(\{u, v, w\}, x^{(1)}, x^{(2)}) \in \mathbb{R}^d$). For these tasks, we set $d = |G|$, where $G$ refers to either the cyclic group or a general group. We will also consider the inputs $x^{(1)}$ and $x^{(2)}$ to be one-hot vectors, representing the group elements being provided as inputs. Thus, for given input elements $(a, b)$, a single neuron can be simplified as $\varphi(\{u, v, w\}, a, b) = (u_a + v_b)^2 w$, where $u_a$ and $v_b$ represent the $a^{th}$ and $b^{th}$ component of $u$ and $v$ respectively. Overall, the network will be given by

$$f(\theta, a, b) = \sum_{i=1}^{m} \varphi(\{u_i, v_i, w_i\}, a, b),$$

with $\theta = \{u_i, v_i, w_i\}_{i=1}^m$ (note that $f(\theta, a, b) \in \mathbb{R}^d$).

For Section 7.5, we will consider the $(n, k)$-sparse parity problem, where the parity is computed on $k$ bits out of $n$. For this task, we will consider a neural network with the activation function $x^k$. A single neuron within the neural network will be represented as $\varphi(\{u, w\}, x) = (u^\top x)^k w$, where $u \in \mathbb{R}^n$, $w \in \mathbb{R}^2$ are the weights associated with a neuron and $x \in \mathbb{R}^n$ is the input provided to the network. The overall network will represented as

$$f(\theta, x) = \sum_{i=1}^{m} \varphi(\{u_i, w_i\}, x),$$

where $\theta = \{u_i, w_i\}_{i=1}^m$.

For any vector $v$ and $k \geq 1$, $\|v\|_k$ represents $\left(\sum |v_i|^k\right)^{1/k}$. For a given neural network with parameters $\theta = \{\omega_i\}_{i=1}^m$, the $L_{a,b}$ norm of $\theta$ is given by $\|\theta\|_{a,b} = \left(\sum_{i=1}^m \|\omega_i\|_a^b\right)^{1/b}$. Here $\{\omega_i\}$ represents the concatenated vector of parameters corresponding to a single neuron.

## 7.3 Theoretical Approach

Suppose we have a dataset $D \subseteq \mathcal{X} \times \mathcal{Y}$, a norm $\|\cdot\|$ and a class of parameterized functions $\{f(\theta, \cdot) \mid \theta \in \mathbb{R}^U\}$, where $f : \mathbb{R}^U \times \mathcal{X} \to \mathbb{R}^{\mathcal{Y}}$ and $\Theta = \{\|\theta\| \leq 1\}$. We define the margin function $g : \mathbb{R}^U \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ as being, for a given datapoint $(x, y) \in D$,

$$g(\theta, x, y) = f(\theta, x)[y] - \max_{y' \in \mathcal{Y} \setminus \{y\}} f(\theta, x)[y'].$$

Then, the margin of the dataset $D$ is given by $h : \mathbb{R}^U \to \mathbb{R}$ defined as

$$h(\theta) = \min_{(x,y) \in D} g(\theta, x, y).$$

Similarly, we define the normalized margin for a given $\theta$ as $h(\theta/\|\theta\|)$.

We train using the regularized objective

$$\mathcal{L}_\lambda(\theta) = \frac{1}{|D|} \sum_{(x,y) \in D} l(f(\theta, x), y) + \lambda \|\theta\|^r$$

where $l$ is the cross-entropy loss. Let $\theta_\lambda \in \arg\min_{\theta \in \mathbb{R}^U} \mathcal{L}_\lambda(\theta)$ be a minimum of this objective, and let $\gamma_\lambda = h(\theta_\lambda/\|\theta_\lambda\|)$ be the normalized margin of $\theta_\lambda$. Let $\gamma^* = \max_{\theta \in \Theta} h(\theta)$ be the maximum normalized margin. The following theorem of Wei et al. (2019b) states that, when using vanishingly small regularization $\lambda$, the normalized margin of global optimizers of $\mathcal{L}_\lambda$ converges to $\gamma^*$.

**Theorem 7.1** (Wei et al. (2019b), Theorem 4.1). *For any norm $\|\cdot\|$, a fixed $r > 0$ and any homogeneous function $f$ with homogeneity constant $\nu > 0$, if $\gamma^* > 0$, then $\lim_{\lambda \to 0} \gamma_\lambda = \gamma^*$.*

This provides the motivation behind studying maximum margin classifiers as a proxy for understanding the global minimizers of $\mathcal{L}_\lambda$ as $\lambda \to 0$. Henceforth, we will focus on characterizing the maximum margin solution: $\Theta^* := \arg\max_{\theta \in \Theta} h(\theta)$.

Note that the maximum margin $\gamma^*$ is given by

$$\begin{aligned}\gamma^* &= \max_{\theta \in \Theta} \min_{(x,y) \in D} g(\theta, x, y) \\ &= \max_{\theta \in \Theta} \min_{q \in \mathcal{P}(D)} \mathop{\mathbb{E}}_{(x,y) \sim q} [g(\theta, x, y)]\end{aligned}$$

where $q$ represents a distribution over data points in $D$. The primary approach in this work for characterizing the maximum margin solution is to exhibit a pair $(\theta^*, q^*)$ such that

$$q^* \in \arg\min_{q \in \mathcal{P}(D)} \mathop{\mathbb{E}}_{(x,y) \sim q} [g(\theta^*, x, y)] \tag{7.1}$$

$$\theta^* \in \arg\max_{\theta \in \Theta} \mathop{\mathbb{E}}_{(x,y) \sim q^*} [g(\theta, x, y)] \tag{7.2}$$

That is, $q^*$ is one of the minimizers of the expected margin with respect to $\theta^*$ and $\theta^*$ is one of the maximizers of the expected margin with respect to $q^*$. The lemma below uses the max-min inequality (Boyd & Vandenberghe, 2004) to show that exhibiting such a pair is sufficient for establishing that $\theta^*$ is indeed a maximum margin solution. The proof for the lemma can be found in Appendix D.5.

**Lemma 7.2.** *If a pair $(\theta^*, q^*)$ satisfies Equations 7.1 and 7.2, then*

$$\theta^* \in \arg\max_{\theta \in \Theta} \min_{(x,y) \in D} g(\theta, x, y)$$

In the following subsections, we will describe our approach for finding such a pair for 1-hidden layer homogeneous neural networks. Furthermore, we will show how exhibiting just a single pair of the above form can enable us to characterize the set of *all* maximum margin solutions. We start off with the case of binary classification, and then extend the techniques to multi-class classification.

### 7.3.1 Binary Classification

In the context of binary classification where $|\mathcal{Y}| = 2$, the margin function $g$ for a given datapoint $(x, y) \in D$ is given by

$$g(\theta, x, y) = f(\theta, x)[y] - f(\theta, x)[y'],$$

where $y' \neq y$. For 1-hidden layer neural networks, by linearity of expectation, the expected margin is given by

$$\mathop{\mathbb{E}}_{(x,y)\sim q}[g(\theta, x, y)] = \sum_{i=1}^{m} \mathop{\mathbb{E}}_{(x,y)\sim q}\left[\varphi(\omega_i, x)[y] - \varphi(\omega_i, x)[y']\right],$$

where $y' \neq y$ and $\theta = \{\omega_i\}_{i=1}^m$. Since the expected margin of the network decomposes into the sum of expected margin of individual neurons, finding a maximum expected margin network simplifies to finding maximum expected margin neurons. Denoting $\psi(\omega, x, y) = \varphi(\omega, x)[y] - \varphi(\omega, x)[y']$, the following lemma holds:

**Lemma 7.3.** *Let* $\Theta = \{\theta : \|\theta\|_{a,b} \leq 1\}$ *and* $\Theta_q^* = \arg\max_{\theta\in\Theta} \mathbb{E}_{(x,y)\sim q}[g(\theta, x, y)]$. *Similarly, let* $\Omega = \{\omega : \|\omega\|_a \leq 1\}$ *and* $\Omega_q^* = \arg\max_{\omega\in\Omega} \mathbb{E}_{(x,y)\sim q}[\psi(\omega, x, y)]$. *For binary classification:*

- ***Single neuron optimization**: Any $\theta \in \Theta_q^*$ has directional support only on $\Omega_q^*$.*
- ***Combining neurons**: If $b = \nu$ (the homogeneity constant of the network) and $\omega_1^*, \ldots, \omega_m^* \in \Omega_q^*$, then for any neuron scaling factors $\sum \lambda_i^\nu = 1, \lambda_i \geq 0$, we have that $\theta = \{\lambda_i \omega_i^*\}_{i=1}^m$ belongs to $\Theta_q^*$.*

The proof for the above lemma can be found in Appendix D.5.1.

To find a $(\theta^*, q^*)$ pair, we will start with a guess for $q^*$ (which will be the uniform distribution in our case as the datasets are symmetric). Then, using the first part of Lemma 7.3, we will find all neurons which can be in the support of $\theta^*$ satisfying Equation 7.2 for the given $q^*$. Finally, for specific

norms of the form $\|\cdot\|_{a,v}$, we will combine the obtained neurons using the second part of Lemma 7.3 to obtain a $\theta^*$ such that $q^*$ satisfies Equation 7.1.

We think of $(\theta^*, q^*)$ as a "certificate pair". By just identifying this single solution, we can characterize the set of *all* maximum margin solutions. Denoting $\text{spt}(q) = \{(x,y) \in D \mid q(x,y) > 0\}$, the following lemma holds:

**Lemma 7.4.** *Let* $\Theta = \{\theta : \|\theta\|_{a,b} \leq 1\}$ *and* $\Theta^*_q = \arg\max_{\theta\in\Theta} \mathbb{E}_{(x,y)\sim q}[g(\theta,x,y)]$. *Similarly, let* $\Omega = \{\omega : \|\omega\|_a \leq 1\}$ *and* $\Omega^*_q = \arg\max_{\omega\in\Omega} \mathbb{E}_{(x,y)\sim q}[\psi(\omega,x,y)]$. *For the task of binary classification, if there exists* $\{\theta^*, q^*\}$ *satisfying Equation 7.1 and 7.2, then any*

$$\hat{\theta} \in \arg\max_{\theta\in\Theta} \min_{(x,y)\in D} g(\theta,x,y)$$

*satisfies the following:*

- $\hat{\theta}$ *has directional support only on* $\Omega^*_{q^*}$.
- *For any* $(x,y) \in spt(q^*), f(\hat{\theta},x,y) - f(\hat{\theta},x,y') = \gamma^*$, *where* $y' \neq y$; *i.e., all points in the support of* $q^*$ *are "on the margin" for any maximum margin solution.*

The proof for the above lemma can be found in Appendix D.5.1.

Thus, we can say that the neurons found by Lemma 7.3 are indeed the exhaustive set of neurons for any maximum margin network. Moreover, any maximum margin solution will have the support of $q^*$ on the margin.

### 7.3.2 Multi-Class Classification

The modular addition and general finite group tasks are multi-class classification problems. For multi-class classification, the margin function $g$ for a given datapoint $(x, y) \in D$ is given by

$$g(\theta, x, y) = f(\theta, x)[y] - \max_{y' \in \mathcal{Y} \setminus \{y\}} f(\theta, x)[y'],$$

For 1-hidden layer networks, the expected margin is given by

$$\mathop{\mathbb{E}}_{(x,y)\sim q}[g(\theta, x, y)] = \mathop{\mathbb{E}}_{(x,y)\sim q}\left[\sum_{i=1}^{m} \varphi(\omega_i, x)[y] - \max_{y' \in \mathcal{Y} \setminus \{y\}} \sum_{i=1}^{m} \varphi(\omega_i, x)[y']\right],$$

Here, due to the max operation, we cannot swap the summation and expectation, and thus the expected margin of the network does not decompose into the expected margins of the neurons as it did in the binary classification case.

To circumvent this issue, we will introduce the notion of *class-weighted margin*. Consider some $\tau : D \to \Delta(\mathcal{Y})$ that assigns a weighting of incorrect labels to every datapoint. For any $(x, y) \in D$, let $\tau$ satisfy the properties that $\sum_{y' \in \mathcal{Y} \setminus \{y\}} \tau(x, y)[y'] = 1$ and $\tau(x, y)[y'] \geq 0$ for all $y' \in \mathcal{Y}$. Using this, we define the class-weighted margin $g'$ for a given datapoint $(x, y) \in D$ as

$$g'(\theta, x, y) = f(\theta, x)[y] - \sum_{y' \in \mathcal{Y} \setminus \{y\}} \tau(x, y)[y'] f(\theta, x)[y'].$$

Note that $g'(\theta, x, y) \geq g(\theta, x, y)$ as $g'$ replaces the max by a weighted sum. Moreover, by linearity of expectation we can say that

$$\mathop{\mathbb{E}}_{(x,y)\sim q}[g'(\theta, x, y)] = \sum_{i=1}^{m} \mathop{\mathbb{E}}_{(x,y)\sim q}\left[\varphi(\omega_i, x)[y] - \sum_{y' \in \mathcal{Y} \setminus \{y\}} \tau(x, y)[y'] \varphi(\omega_i, x)[y']\right],$$

Denoting $\psi'(\omega, x, y) = \varphi(\omega, x)[y] - \sum_{y' \in \mathcal{Y} \setminus \{y\}} \tau(x, y)[y'] \varphi(\omega, x)[y']$, a result analogous to Lemma 7.3 holds for the class-weighted margin (proof can be found in Appendix D.5.2):

**Lemma 7.5.** *Let* $\Theta = \{\theta : \|\theta\|_{a,b} \leq 1\}$ *and* $\Theta'^*_q = \arg\max_{\theta \in \Theta} \mathbb{E}_{(x,y)\sim q} [g'(\theta, x, y)]$. *Similarly, let* $\Omega = \{\omega : \|\omega\|_a \leq 1\}$ *and* $\Omega'^*_q = \arg\max_{\omega \in \Omega} \mathbb{E}_{(x,y)\sim q} [\psi'(\omega, x, y)]$. *Then:*

- ***Single neuron optimization:*** *Any* $\theta \in \Theta'^*_q$ *has directional support only on* $\Omega'^*_q$.
- ***Combining neurons:*** *If* $b = \nu$ *and* $\omega_1^*, \ldots, \omega_m^* \in \Omega'^*_q$, *then for any neuron scaling factors* $\sum \lambda_i^\nu = 1, \lambda_i \geq 0$, *we have that* $\theta = \{\lambda_i \omega_i^*\}_{i=1}^m$ *belongs to* $\Theta'^*_q$.

The above lemma helps us characterize $\Theta'^*_q$ for a given distribution $q$. Thus, applying it to a given $q^*$, we can find

$$\theta^* \in \arg\max_{\theta \in \Theta} \mathop{\mathbb{E}}_{(x,y)\sim q^*} \left[ g'(\theta, x, y) \right]. \tag{7.3}$$

To further ensure that $\theta^*$ also satisfies the corresponding equation for $g$ (i.e., Equation 7.2) we will consider the following condition:

**C.1** For any $(x, y) \in \operatorname{spt}(q^*)$, it holds that $g'(\theta^*, x, y) = g(\theta^*, x, y)$. This translates to any label with non-zero weight being one of the incorrect labels where $f$ is maximized: $\{\ell \in \mathcal{Y} \setminus \{y\} : \tau(x, y)[\ell] > 0\} \subseteq \arg\max_{\ell \in \mathcal{Y} \setminus \{y\}} f(\theta^*, x)[\ell]$.

The main lemma used for finding the maximum margin solutions for multi-class classification is stated below:

**Lemma 7.6.** *Let* $\Theta = \{\theta : \|\theta\|_{a,b} \leq 1\}$ *and* $\Theta'^*_q = \arg\max_{\theta \in \Theta} \mathbb{E}_{(x,y)\sim q} [g'(\theta, x, y)]$. *Similarly, let* $\Omega = \{\omega : \|\omega\|_a \leq 1\}$ *and* $\Omega'^*_q = \arg\max_{\omega \in \Omega} \mathbb{E}_{(x,y)\sim q} [\psi'(\omega, x, y)]$. *If* $\exists \{\theta^*, q^*\}$ *satisfying Equations 7.1 and 7.3, and* ***C.1*** *holds, then:*

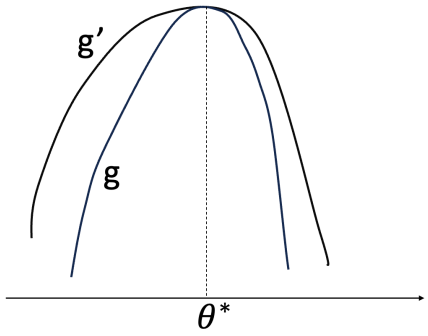



**Figure 7.3:** A schematic illustration of the relation between class-weighted margin $g'$ and maximum margin $g$.

- $\theta^* \in \arg\max_{\theta \in \Theta} g(\theta, x, y)$
- *Any* $\hat{\theta} \in \arg\max_{\theta \in \Theta} \min_{(x,y) \in D} g(\theta, x, y)$ *satisfies the following:*
  - $\hat{\theta}$ *has directional support only on* $\Omega'^{*}_{q^*}$.
  - *For any* $(x, y) \in spt(q^*), f(\hat{\theta}, x, y) - \max_{y' \in \mathcal{Y} \setminus \{y\}} f(\hat{\theta}, x, y') = \gamma^*$*, i.e, all points in the support of* $q^*$ *are on the margin for any maximum margin solution.*

The first part of the above lemma follows from the fact that $g'(\theta, x, y) \geq g(\theta, x, y)$. Thus, any maximizer of $g'$ satisfying $g' = g$ is also a maximizer of $g$ (See Figure 7.3). The second part states that the neurons found using Lemma 7.5 are indeed the exhaustive set of neurons for any maximum margin network. Moreover, any maximum margin solution has the support of $q^*$ on margin. The proof for the lemma can be found in Appendix D.5.2.

Overall, to find a $(\theta^*, q^*)$ pair, we will start with a guess of $q^*$ (which will be uniform in our case as the datasets are symmetric) and a guess of the weighing $\tau$ (which will be uniform for the modular addition case). Then, using the first part of Lemma 7.5, we will find all neurons which can be in the support of $\theta^*$ satisfying Equation 7.3 for the given $q^*$. Finally, for specific norms of the form $\|\cdot\|_{a,\nu}$, we will combine the obtained neurons using the second part of Lemma 7.5 to obtain a $\theta^*$ such that it satisfies **C.1** and $q^*$ satisfies Equation 7.1. Thus, we will primarily focus on maximum margin

with respect to $L_{2,\nu}$ norm in this work.

### 7.3.3 Blueprint for the case studies

In each case study, we want to find a certificate pair: a network $\theta^*$ and a distribution on the input data points $q^*$, such that Equation 7.1 and 7.2 are satisfied. Informally, these are the main steps involved in the proof approach:

1. As the datasets we considered are symmetric, we consider $q^*$ to be uniformly distributed on the input data points.

2. Using the **Single neuron optimization** part of Lemma 7.5, we find all neurons that maximize the expected class-weighted margin. Only these neurons can be part of a network $\theta^*$ satisfying Equation 7.3.

3. Using the **Combining neurons** part of Lemma 7.5, we combine the above neurons into a network $\theta^*$ such that

   (a) All input points are on the margin, i.e, $q^*$ satisfies Equation 7.1.

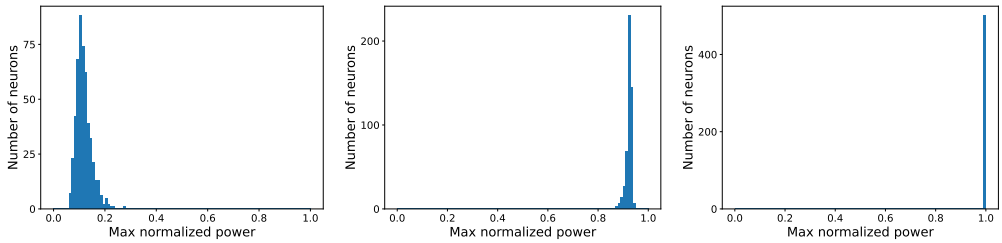
   (b) The class-weighted margin is equal to the maximum margin, i.e, $\theta^*$ satisfies **C.1**.

Then, using Lemma 7.6, we can say that the network $\theta^*$ maximizes the margin.

## 7.4 Cyclic groups (modular addition)

For a prime $p > 2$, let $\mathbb{Z}_p$ denote the cyclic group on $p$ elements. For a function $f : \mathbb{Z}_p \to \mathbb{C}$, the discrete Fourier transform of $f$ at a frequency $j \in \mathbb{Z}_p$ is defined as

$$\hat{f}(j) := \sum_{k \in \mathbb{Z}_p} f(k) \exp(-2\pi i \cdot jk/p).$$




**(a)** Initial distribution **(b)** ReLU activation **(c)** Quadratic activation

**Figure 7.4:** The maximum normalized power of the embedding vector of a neuron is given by $\max_i |\hat{u}[i]|^2/(\sum |\hat{u}[j]|^2)$, where $\hat{u}[i]$ represents the $i^{th}$ component of the Fourier transform of $u$. (a) Initially, the maximum power is randomly distributed. (b) For 1-hidden layer ReLU network trained with $L_2$ regularization, the final distribution of maximum power seems to be concentrated around 0.9, meaning neurons are nearly 1-sparse in frequency space but not quite. (c) For 1-hidden layer quadratic network trained with $L_{2,3}$ regularization, the final maximum power is almost exactly 1 for all the neurons, so the embeddings are 1-sparse in frequency space, as predicted by the maximum margin analysis.

Note that we can treat a vector $v \in \mathbb{C}^p$ as a function $v : \mathbb{Z}_p \to \mathbb{C}$, thereby endowing it with a Fourier transform. Consider the input space $\mathcal{X} := \mathbb{Z}_p \times \mathbb{Z}_p$ and output space $\mathcal{Y} := \mathbb{Z}_p$. Let the dataset $D_p := \{((a,b), a+b) : a, b \in \mathbb{Z}_p\}$.

**Theorem 7.7.** *Consider one-hidden layer networks $f(\theta, a, b)$ of the form given in section 7.2 with $m \geq 4(p-1)$ neurons. The maximum $L_{2,3}$-margin of such a network on the dataset $D_p$ is:*

$$\gamma^* = \sqrt{\frac{2}{27}} \cdot \frac{1}{p^{1/2}(p-1)}.$$

*Any network achieving this margin satisfies the following conditions:*

1. *for each neuron $\varphi(\{u, v, w\}; a, b)$ in the network, there exists a scaling constant $\lambda \in \mathbb{R}$ and a frequency $\zeta \in \{1, \ldots, \frac{p-1}{2}\}$ such that*

$$u(a) = \lambda \cos(\theta_u^* + 2\pi\zeta a/p)$$
$$v(b) = \lambda \cos(\theta_v^* + 2\pi\zeta b/p)$$
$$w(c) = \lambda \cos(\theta_w^* + 2\pi\zeta c/p)$$

*for some phase offsets* $\theta_u^*, \theta_v^*, \theta_w^* \in \mathbb{R}$ *satisfying* $\theta_u^* + \theta_v^* = \theta_w^*$.

2. *For every frequency* $\zeta \in \{1, \ldots, \frac{p-1}{2}\}$, *at least one neuron in the network uses this frequency.*

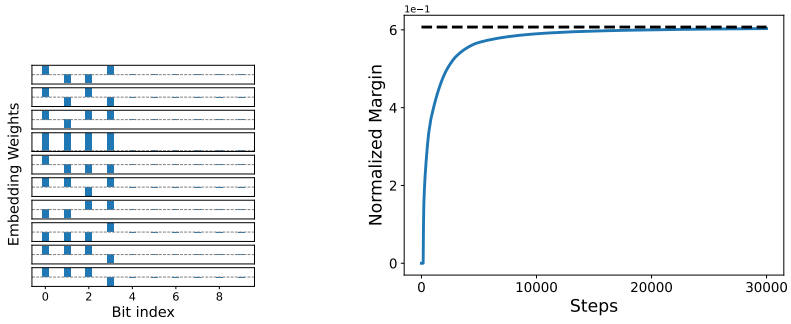
*Proof outline.* Following the blueprint described in the previous section, we first prove that neurons of the form above (and only these neurons) maximize the expected class-weighted margin $\mathbb{E}_{a,b}[\psi'(u, v, w)]$ with respect to the uniform distribution $q^* = \text{unif}(\mathcal{X})$. We will use the uniform class weighting: $\tau(a, b)[c'] := 1/(p-1)$ for all $c' \neq a + b$. As a crucial intermediate step, we prove that

$$\mathop{\mathbb{E}}_{a,b}[\psi'(\{u, v, w\}, a, b)] = \frac{2}{(p-1)p^2} \sum_{j \neq 0} \hat{u}(j)\hat{v}(j)\hat{w}(-j),$$

Maximizing the above expression subject to the max-margin norm constraint $\sum_{j \neq 0} \left(|\hat{u}(j)|^2 + |\hat{v}(j)|^2 + |\hat{w}(j)|^2\right) \leq 1$ leads to sparsity in Fourier space.

Then, we describe a network $\theta^*$ (of width $4(p-1)$) composed of such neurons, and that satisfies Equation 7.1 and condition **C.1**. By Lemma 7.6, part (1) of Theorem 7.7 will follow, and $\theta^*$ will be an example of a max-margin network. Finally, in order to show that all frequencies are used, we introduce the multidimensional discrete Fourier transform. We prove that each neuron only contributes a single frequency to the multi-dimensional DFT of the network; but that second part of Lemma 7.6 implies that all frequencies are present in the full network's multidimensional DFT. The full proof can be found in Appendix D.6. □

As demonstrated in Figure 5.1 and 7.4, empirical networks trained with gradient descent with $L_{2,3}$ regularization approach the theoretical maximum margin, and have single frequency neurons. Figure D.1 in the Appendix verifies that all frequencies are present in the network.




**(a)** Embeddings

**(b)** Normalized margin

**Figure 7.5:** Final neurons with highest norm and the evolution of normalized $L_{2,5}$ margin over training of a 1-hidden layer quartic network (activation $x^4$) on $(10, 4)$ sparse parity dataset with $L_{2,5}$ regularization. The network approaches the theoretical maximum margin that we predict.

## 7.5 Sparse parity

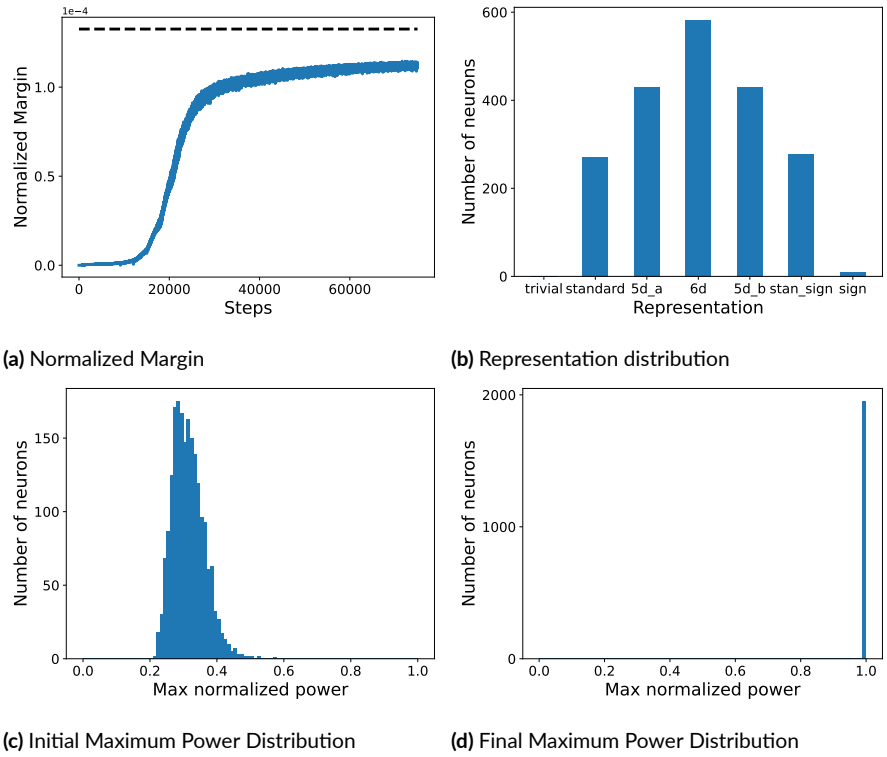
In this section, we will establish the max margin features that emerge when training a neural network on the sparse parity task. Consider the $(n, k)$-sparse parity problem, where the parity is computed over $k$ bits out of $n$. To be precise, consider inputs $x_1, \ldots, x_n \in \{\pm 1\}$. For a given subset $S \subseteq [n]$ such that $|S| = k$, the parity function is given by $\Pi_{j \in S} x_j$.

**Theorem 7.8.** *Consider a single hidden layer neural network of width $m$ with the activation function given by $x^k$, i.e, $f(x) = \sum_{i=1}^{m} (u_i^\top x)^k w_i$, where $u_i \in \mathbb{R}^n$ and $w_i \in \mathbb{R}^2$, trained on the $(n, k)-$sparse parity task. Without loss of generality, assume that the first coordinate of $w_i$ corresponds to the output for class $y = +1$. Denote the vector $[1, -1]$ by $\boldsymbol{b}$. Provided $m \geq 2^{k-1}$, the $L_{2,k+1}$ maximum margin is:*

$$\gamma^* = k! \sqrt{2(k+1)^{-(k+1)}}.$$

*Any network achieving this margin satisfies the following conditions:*

1. *For every $i$ having $\|u_i\| > 0$, $spt(u_i) = S$, $w_i$ lies in the span of $\boldsymbol{b}$ and $\forall j \in S$, $|u_i[j]| = \|w_i\|$.*




**(a)** Normalized Margin

**(b)** Representation distribution

**(c)** Initial Maximum Power Distribution

**(d)** Final Maximum Power Distribution

**Figure 7.6:** This figure demonstrates the training of a 1-hidden layer quadratic network on the symmetric group $S5$ with $L_{2,3}$ regularization. (a) Evolution of the normalized $L_{2,3}$ margin of the network with training. It approaches the theoretical maximum margin that we predict. (b) Distribution of neurons spanned by a given representation. Higher dimensional representations have more neurons as given by our construction. (c) and (d) Maximum normalized power is given by $\max_i \hat{u}[i]^2 / (\sum_j \hat{u}[j]^2)$ where $\hat{u}[i]$ refers to the component of weight vector $u$ spanned by the basis vectors corresponding to $i^{th}$ representation. This is random at initialization, but towards the end of training, all neurons are concentrated in a single representation, as predicted by maximum margin.

2. *For every $i$,* $\left(\Pi_{j \in S} u_i[j]\right) \left(w_i^\top \boldsymbol{b}\right) \geq 0$.

As shown in Figure 7.5, a network trained with gradient descent and $L_{2,k+1}$ regularization exhibits these properties, and approaches the theoretically-predicted maximum margin. The proof for Theorem 7.8 can be found in Appendix D.7.

## 7.6 Finite Groups with Real Representations

We conclude our case study on algebraic tasks by studying group composition on finite groups $G$. Namely, here we set $\mathcal{X} := G \times G$ and output space $\mathcal{Y} := G$. Given inputs $a, b \in G$ we train the network to predict $c = ab$. We wish to characterize the maximum margin features similarly to the case of modular addition; here, our analysis relies on principles from group representation theory.

### 7.6.1 Brief Background and Notation

The following definitions and notation are essential for stating our main result, and further results are presented with more rigor in Appendix D.8.

A real representation of a group $G$ is a finite dimensional real vector space $V = \mathbb{R}^d$ and a group homomorphism (i.e. a map preserving the group structure) $R : G \to GL(V)$. We denote such a representation by $(R, V)$ or just by $R$. The dimension of a representation $R$, denoted $d_R$, is the dimension of $V$. Our analysis focuses on *unitary, irreducible, real* representations of $G$. The number of such representations is precisely equal to the number of conjugacy classes of $G$ where the conjugacy class of $a \in G$ is defined as $C(a) = \{gag^{-1} : g \in G\}$.

A quantity important to our analysis is the *character* of a representation $R$, denoted $\chi_R : G \to \mathbb{R}$ given by $\chi_R(g) = \mathrm{tr}(R(g))$. It was previously observed by Chughtai et al. (2023) that one-layer ReLU MLPs and transformers learn the task by mapping inputs $a, b$ to their respective matrices $R(a), R(b)$ for some irreducible representation $R$ and performing matrix multiplication with $R(c^{-1})$ to output logits proportional to the character $\chi_R(abc^{-1}) = \mathrm{tr}(R(a)R(b)R(c^{-1}))$, which is in particular maximized when $c = ab$. They also find evidence of network weights being *spanned by representations*, which we establish rigorously here.

For each representation $R$ we will consider the $|G|$-dimensional vectors by fixing one index in the matrices outputted by $R$, i.e. vectors $(R(g)_{(i,j)})_{g \in G}$ for some $i, j \in [d_R]$. For each $R$, this gives ${d_R}^2$

vectors. Letting $K$ be the number of conjugacy classes and $R_1, \ldots, R_K$ be the corresponding irreducible representations, since $|G| = \sum_{n=1}^{K} d_{R_n}^2$, taking all such vectors for each representation will form a set of $|G|$ vectors which we will denote $\rho_1, \ldots, \rho_{|G|}$ ($\rho_1$ is always the vector corresponding to the trivial representation). These vectors are in fact orthogonal, which follows from orthogonality relations of the representation matrix elements $R(g)_{(i,j)}$ (see Appendix D.8 for details). Thus, we refer to this set of vectors as *basis vectors* for $\mathbb{R}^{|G|}$. One can ask whether the maximum margin solution in this case has neurons which are spanned only by basis vectors corresponding to a single representation $R$, and if all representations are present in the network— the analogous result we obtained for modular addition in Theorem 7.7. We show that this is indeed the case.

### 7.6.2 The Main Result

Our main result characterizing the max margin features for group composition is as follows.

**Theorem 7.9.** *Consider a single hidden layer neural network of width $m$ with quadratic activation trained on learning group composition for $G$ with real irreducible representations. Provided $m \geq 2\sum_{n=2}^{K} {d_{R_n}}^3$ and $\sum_{n=2}^{K} {d_{R_n}}^{1.5} \chi_{R_n}(C) < 0$ for every non-trivial conjugacy class $C$, the $L_{2,3}$ maximum margin is:*

$$\gamma^* = \frac{2}{3\sqrt{3|G|}} \frac{1}{\left(\sum_{n=2}^{K} d_{R_n}^{2.5}\right)}.$$

*Any network achieving this margin satisfies the following conditions:*

1. *For every neuron, there exists a non-trivial representation such that the input and output weight vectors are spanned only by that representation.*
2. *There exists at least one neuron spanned by each representation (except for the trivial representation) in the network.*

The complete proof for Theorem 7.9 can be found in Appendix D.9.

The condition that $\sum_{n=2}^{K} {d_{R_n}}^{1.5} \chi_{R_n}(C) < 0$ for every non-trivial conjugacy class $C$ holds for the symmetric group $S_k$ up until $k = 5$. In this case, as shown in Figure 7.6, network weights trained with gradient descent and $L_{2,3}$ regularization exhibit similar properties. The maximum margin of the network approaches what we have predicted in theory. Analogous results for training on $S_3$ and $S_4$ in Figures D.2 and D.3 are in the Appendix.

Although Theorem 7.9 does not apply to all finite groups with real representations, it can be extended to apply more generally. The theorem posits that *every* representation is present in the network, and *every* conjugacy class is present on the margin. Instead, for general finite groups, each neuron still satisfies the characteristics of max margin solutions in that it is only spanned by one non-trivial representation, but only a *subset* of representations are present in the network; moreover, only a subset of conjugacy classes are present on the margin. More details are given in Appendix D.9.2.

## 7.7 Discussion

We have shown that the simple condition of margin maximization can, in certain algebraic learning settings, imply very strong conditions on the representations learned by neural networks. The mathematical techniques we introduce are general, and may be able to be adapted to other settings than the ones we consider. Our proof holds for the case of $x^2$ activations ($x^k$ activations, in the $k$-sparse parity case) and $L_{2,\nu}$ norm, where $\nu$ is the homogeneity constant of the network. Empirical findings suggest that the results may be transferable to other architectures and norms. In general, we think explaining how neural networks adapt their representations to symmetries and other structure in data is an important subject for future theoretical and experimental inquiry.

*This chapter is based on "The Evolution of Statistical Induction Heads: In-Context Learning Markov Chains" (Edelman et al., 2024b), written in collaboration with Ezra Edelman, Surbhi Goel, Eran Malach, and Nikolaos Tsilivis .*

# 8

# Induction Heads

Large language models have the ability to generate text that mimics patterns in their inputs. We introduce a simple Markov Chain sequence modeling task in order to study how this in-context learning (ICL) capability emerges. In our setting, each example is sampled from a Markov chain drawn from a prior distribution over Markov chains. Transformers trained on this task form *statistical induction heads* which compute accurate next-token probabilities given the bigram statistics of the context. During the course of training, models pass through multiple phases: after an initial stage

in which predictions are uniform, they learn to sub-optimally predict using in-context single-token statistics (unigrams); then, there is a rapid phase transition to the correct in-context bigram solution. We conduct an empirical and theoretical investigation of this multi-phase process, showing how successful learning results from the interaction between the transformer's layers, and uncovering evidence that the presence of the simpler unigram solution may delay formation of the final bigram solution. We examine how learning is affected by varying the prior distribution over Markov chains, and consider the generalization of our in-context learning of Markov chains (ICL-MC) task to $n$-grams for $n > 2$.

## 8.1 Introduction

Large language models (LLMs) exhibit a remarkable ability to perform *in-context learning* (ICL): learning from patterns in their input context (Brown et al., 2020; Dong et al., 2022). The ability of LLMs to adaptively learn from context is profoundly useful, yet the underlying mechanisms of this emergent capability are not fully understood.

In an effort to better understand ICL, some recent works propose to study ICL in controlled synthetic settings—in particular, training transformers on mathematically defined tasks which require learning from the input context. For example, a recent line of works studies the ability of transformers to perform ICL of standard supervised learning problems such as linear regression (Garg et al., 2022; Akyürek et al., 2022; Li et al., 2023; Wu et al., 2023). Studying these well-understood synthetic learning tasks enables fine-grained control over the data distribution, allows for comparisons with established supervised learning algorithms, and facilitates the examination of the in-context "algorithm" implemented by the network. That said, these supervised settings are reflective specifically of *few-shot learning*, which is only a special case of the more general phenomenon of networks incorporating patterns from their context into their predictions. A few re-

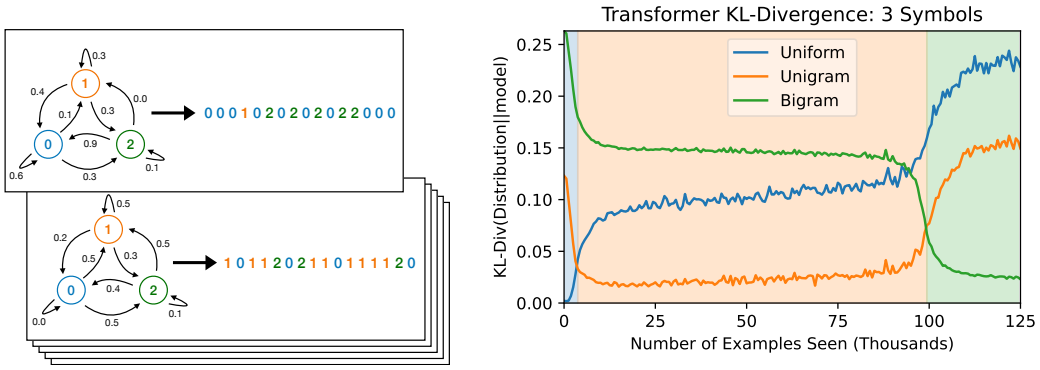



**Figure 8.1:** (**left**) We train small transformers to perform in-context learning of Markov chains (ICL-MC). Each training sequence is generated by sampling a transition matrix from a prior distribution, and then sampling a sequence from this Markov chain. (**right**) Distance of a transformer's output distribution to several well-defined strategies over the course of training on our in-context Markov chain task. The model passes through three stages: (1) predicting a uniform distribution, (2) predicting based on in-context unigram statistics, (3) predicting based on in-context bigram statistics. Shading is based on the minimum of the curves.

cent works (Bietti et al., 2023; Xie et al., 2022) go beyond the case of cleanly separated in-context inputs and outputs, studying in-context learning on distributions based on discrete stochastic processes.

The goal of this work is to propose and analyze a simple synthetic setting for studying ICL. To achieve this, we consider $n$-gram models (Brown et al., 1992; Shannon, 1948; Chomsky, 1956), one of the simplest and oldest methods for language modeling. An $n$-gram language model predicts the probability of a token based on the preceding $n-1$ tokens, using fixed-size chunks ($n$-grams) of text data to capture linguistic patterns. Our work studies ICL of $n$-gram models, where the network needs to compute the conditional probability of the next token based on the statistics of the tokens observed in the input context, rather than on the statistics of the entire training data. We mainly focus on the simple case of $n = 2$; i.e., bigram models, which can be represented as Markov chains. We therefore consider ICL of Markov chains (ICL-MC): we train a two layer attention-only transformer on sequences of tokens, where each sequence is produced by a different Markov chain, generated using a different transition matrix (see Figure 8.1 (left)).

By studying ICL-MC, we are able to replicate and study multiple phenomena that have been observed in ICL for LLMs, and identify new ones. We demonstrate our findings using a combination of empirical observations on transformers trained from scratch on ICL-MC and theoretical analysis of a simplified linear transformer. Our key findings are summarized below:

- **Transformers learn statistical induction heads to optimally solve ICL-MC**. Prior work studying ICL in transformers revealed the formation of *induction heads* (Elhage et al., 2021), a circuit that looks for recent occurrence(s) of the current token, and boosts the probabilities of tokens which followed in the input context. We show that in order to solve ICL-MC, transformers learn *statistical* induction heads that are able to compute the correct *conditional (posterior) probability* of the next token given all previous occurrences of the prior token (see the attention patterns in Figure 8.2). We show that these statistical induction heads lead to the transformers achieving performance approaching that of the Bayes-optimal predictor.

- **Transformers learn predictors of increasing complexity and undergo a phase transition when increasing complexity.** We observe that transformers display *phase transitions* when learning Markov chains—learning appears to be separated into phases, with fast drops in loss between the phases. We are able to show that different phases correspond to learning models of increased complexity—unigrams, then bigrams (see Figure 8.1)—and characterize the transition between the phases. We also consider the $n$-gram generalization of our setting where the next token is generated based on the previous $n-1$ tokens.

- **Simplicity bias may slow down learning.** We provide evidence that the model's inherent bias towards simpler solutions (in particular, in-context unigrams) causes learning of the optimal solution to be delayed. Changing the distribution of the in-context examples to remove the usefulness of in-context unigrams leads to faster convergence, even when evaluated on the original distribution.

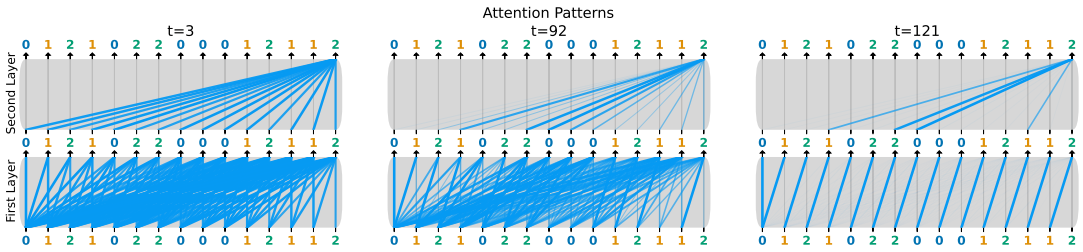



**Figure 8.2:** Attention for a fixed input at various time steps in training. These diagrams show where the attention heads are attending to at each layer. In the second layer, only the last token attention is shown. Tokens on top attend to tokens below them. Attention starts off uniform, but by the end of training, the layers are clearly acting the same as the induction head construction. Specifically, in the first layer each token is attending to the previous token. In the second layer, the current token, a 2, is attending to tokens that followed 2s, allowing bigram statistics to be calculated. Figure E.5 shows the full attention matrices as heatmaps.

- **Alignment of layers is crucial.** We show that the transition from a phase of learning the simple-but-inadequate solution to the complex-and-correct solution happens due to an alignment between the layers of the model: the learning signal for the first layer is tied to the extent to which the second layer approaches its correct weights.

- **Alternating patterns in positional embeddings.** When we train transformers with relative position embeddings, the theoretical optimization analysis of our simplified model indicates that along the way to the correct solution, the first layer develops a bias toward looking back an *odd* number of tokens, even though only looking back by 1 is clearly useful. We empirically observe this curious phenomenon in real transformers as well.

### 8.1.1 Related Work

In-context Learning. Recently, many works have focused on understanding how ICL emerges in language models. In (Chan et al., 2022), the authors discuss how properties of the data distribution promote ICL, with a focus on empirical observations. Xie et al. (2022) studies a data model similar to ours, demonstrating that language models trained on Hidden Markov Models (HMMs)

can learn in-context HMMs not found in the training data. Abernethy et al. (2023) study the ability of transformers to segment the context into pairs of examples and labels and provide learning guarantees when the labeling is of the form of a sparse function. The work of Bietti et al. (2023) studies the dynamics of training transformers on a task that is reminiscent of our Markov chain setting but has additional complexities. Instead of drawing a fresh Markov chain for each sequence, in their task all sequences are sampled from the same Markov chain; after certain 'trigger' tokens, the following 'output' token is chosen deterministically within a sequence. Thus, successful prediction requires incorporating both global bigram statistics and in-context deterministic bigram copying, unlike in our setting where the patterns computed by *statistical* induction heads are necessary and sufficient. As in our work, they identify multiple distinct stages of training and show how multiple top-down gradient steps lead to a solution.

Other works observe that ICL is possible due to the ability of transformers to implement gradient descent as a "meta learning" algorithm, and show some evidence that this indeed corresponds to how transformers learn in-context (Von Oswald et al., 2023; Dai et al., 2022). The work of Li et al. (2023) presents a theoretical framework for studying ICL, providing some risk bounds on ICL of supervised learning algorithms. Guo et al. (2023) construct synthetic in-context learning problems with a compositional structure, studying the representation capacity of transformers to learn these problems in-context. In (Hendel et al., 2023), the authors demonstrate that transformers learn to represent task vectors, providing a mechanistic analysis of ICL in LLMs. Kirsch et al. (2022) view ICL as a broad meta-learning paradigm, and observe that transformers meta-trained on real image classification tasks undergo similar phase transitions as the ones we observe in this work.

Induction Heads. Elhage et al. (2021) studies the formation of induction heads, sub-components of transformers that match previous occurrences of the current token, retrieving the token that succeeds the most recent occurrence. Olsson et al. (2022) studies in-context learning by analyzing the

formation of induction heads in language models, showing empirical evidence that both large and small transformers display a phase transition in the ability to learn in-context. Reddy (2023) also studies the formation of induction heads and their role in ICL, showing empirically that a three layer network exhibits a sudden formation of induction heads towards solving some ICL problem of interest. Bietti et al. (2023) study the effect of specific trigger tokens on the formation of induction heads.

Phase Transitions. It has been observed in different contexts that neural networks and language models display a sudden drop in loss during their training process. This phase transition is often related to emergence of new capabilities in the network. The work of (Power et al., 2021) observed the "grokking" phenomena, where the test loss of neural networks sharply drops, long after the network overfits the training data. (Chen et al., 2023) shows another example of a phase transition in language model training, where the formation of specific attention mechanisms happen suddenly in training, causing the loss to quickly drop. Barak et al. (2022) observe that neural networks trained on complex learning problems display a phase transition when converging to the correct solution. Several works (Kumar et al., 2023; Lyu et al., 2023) attribute these phase transitions to rapid changes in the inductive bias of networks, while Merrill et al. (2023) argue that the models are sparser after the phase change. Schaeffer et al. (2023) warn that phenomena in deep learning that seem to be discontinuous can actually be understood to evolve continuously once seen through the right lens.

Simplicity Bias. Various works observed that neural networks have a "simplicity bias", which causes them to "prioritize" learning simple patterns first (Arpit et al., 2017; Valle-Perez et al., 2018). The work of (Kalimeris et al., 2019) shows that SGD learns functions of increased complexity, first fitting a linear concept to the data before moving to more complex functions. (Shah et al., 2020)

shows that the simplicity bias of neural networks can sometimes be harmful, causing them to ignore important features of the data. Chen et al. (2023) demonstrate the effect of simplicity bias on language tasks that require understanding of syntactic structure. Abbe et al. (2023) provide a theoretical framework for understanding how the simplicity of the target function can govern the convergence time of SGD, describing how simple partial solutions can speed up learning; in contrast, in our setting, the unigram solution appears likely to be a distractor which delays learning of the correct solution.

Concurrent works In parallel to this work, there have been a number of papers devoted to the study of similar questions regarding in-context learning or Markov chains: Akyürek et al. (2024) empirically compare the ability of different architectures to perform in-context learning of regular languages. Hoogland et al. (2024) observe similar stage-wise learning behaviors on transformers trained on language or synthetic linear regression tasks. Makkuva et al. (2024) study the loss landscape of transformers trained on sequences sampled from a single Markov Chain.

## 8.2 Setup

In this section, we describe our learning problem and present the neural networks that we will use for learning.

### 8.2.1 ICL-MC Task

Our learning task consists of Markov Chains with random transition matrices. The goal is to in-context estimate the transition probabilities from sampled sequences, in order to predict the next state. Formally, a sample is a Markov Chain with state space $S = \{1, \dots, k\}$ and a transition matrix $\mathcal{P}$ randomly sampled from some prior distribution, with $x_1$ drawn from some other prior distribution (potentially dependent on $\mathcal{P}$), and the rest of $\boldsymbol{x} = (x_1, \dots, x_t)$ drawn from the Markov Chain.

We primarily focus on the case where each row of the matrix is sampled from the Dirichlet distribution with concentration parameter $\alpha$, i.e. $\mathcal{P}_{i,:} \sim \mathrm{Dir}(\alpha)$. We want to learn a predictor that, given context $x_1, \ldots, x_t$, predicts the next token, $x_{t+1}$. Note that this is an inherently non-deterministic task, even provided full information about the transition matrix, and as such it can better capture certain properties of language than previous in-context learning modeling approaches (Garg et al., 2022).

We focus on the case of the *flat* Dirichlet distribution, with $\alpha = (1, \ldots, 1)^\top$, that corresponds to uniformly random transition probabilities between states. We draw the initial state $x_1$ from the stationary distribution $\pi$ of the chain (which exists almost surely). We primarily consider the case where the number of states $k$ is 2 or 3.

In subsection 8.3.3, we consider the generalization of this setting to $n$-grams for $n > 2$. Instead of $\Pr(x_t)$ being determined by $x_{t-1}$, we let $\Pr(x_t)$ be determined by $x_{t-n+1}, \ldots, x_{t-1}$, according to a conditional distribution $\mathcal{P}$ drawn from some prior. In particular, for each tuple of $n-1$ tokens, we sample the vector of conditional probabilities for the next state from a flat Dirichlet distribution.

### 8.2.2 Potential Strategies for (Partially) Solving ICL-MC

We adopt the Bayesian interpretation of in-context learning (Xie et al., 2022), in which a prior distribution is given by the training data, and, at test time, the model updates this prior given the in-context sequence. In this framework, we focus on two strategies for Bayesian inference: a *unigram* strategy which assumes tokens in each sequence are i.i.d. samples, and the *bigram* strategy which correctly takes into account dependencies among adjacent tokens.

1ST STRATEGY: UNIGRAMS Since we let the Markov chain reach its stationary distribution (which exists a.s.), the optimal strategy across unigrams is just to count frequency of states and form a posterior belief about the stationary distribution. Unfortunately, the stationary distribution of

this random Markov chain does not admit a simple analytical characterization when there is a finite number of states, but it can be estimated approximately. At the limit of $k \to \infty$, the stationary distribution converges to the uniform distribution (Bordenave et al., 2008).

2ND STRATEGY: BIGRAMS For any pair of states $i$ and $j$, let $\mathcal{P}_{ij}$ be probability of transitioning from $i$ to $j$. On each sample $\boldsymbol{x}$, we can focus on the transitions from the $i$-th state, which follow a categorical distribution with probabilities equal to $(\mathcal{P}_{i1}, \ldots, \mathcal{P}_{ik})$. If we observe the in-context empirical counts $\{c_{ij}\}_{j=1}^{k}$ of the transitions, then $\mathcal{P}_{ij}$ is given by:

$$(\mathcal{P}_{i1}, \ldots, \mathcal{P}_{ik}) \,|\boldsymbol{x} \sim \mathrm{Dir}(k, c_{i1} + \alpha_1, \ldots, c_{ik} + \alpha_k), \tag{8.1}$$

where, recall, $\alpha_1, \ldots, \alpha_k$ are the Dirichlet concentration parameters of the prior. Hence, each $\mathcal{P}_{ij}$ has a (marginal) distribution that is actually a Beta distribution:

$$\mathcal{P}_{ij}|\boldsymbol{x} \sim \mathrm{Beta}\left(c_{ij} + \alpha_j, \sum_j \alpha_j + N_i - \alpha_j - c_{ij}\right), \tag{8.2}$$

where $N_i$ is the total number of observed transitions from state $i$. As such, our best (point) estimate for each state $j$ is given by:

$$\mathbb{E}\left[\mathcal{P}_{ij}|\boldsymbol{x}\right] = \frac{c_{ij} + \alpha_j}{N + \sum_i \alpha_i}. \tag{8.3}$$

For the uniform Dirichlet, $\alpha = (1, \ldots, 1)^\top$, it is $\mathbb{E}\left[\mathcal{P}_{ij}|\boldsymbol{x}\right] = \frac{c_{ij}+1}{N_i+k}$.

*Remark* 8.1. The bigram strategy implicitly assumes that the first token $x_1$ is sampled uniformly, as opposed to being sampled from the stationary distribution (which is used in our experiments and theoretical results). As the context length grows, the bigram statistics approach the Bayes optimal solution either way and this difference becomes negligible.

### 8.2.3 Architectures: Transformers and Simplifications

We are mainly interested in investigating how transformers (Vaswani et al., 2017) can succeed in in-context learning this task. We focus on attention-only transformers with 2 layers with causal masking which is a popular architecture for language modeling. Given an input sequence $\boldsymbol{x}$, the output of an $n$-layer attention-only transformer* is:

$$TF(\boldsymbol{x}) = P \circ (Attn_n + I) \cdots \circ (Attn_1 + I) \tag{8.4}$$

Where $\mathbf{e}_{\boldsymbol{x}} \in \mathbb{R}^{t \times d}$ is an embedding of $\boldsymbol{x}$, and $Attn(\boldsymbol{x})$ is masked self attention with relative position embeddings (Shaw et al., 2018), which is parameterized by $W_Q, W_K, W_V \in \mathbb{R}^{d \times d}, v \in \mathbb{R}^{t \times d}$:

$$\begin{aligned} Attn(z) &= \text{softmax}(\text{mask}(A))zW_V \\ A_{i,j} = A_{i,j} &= \frac{(z_i W_Q)(z_j W_K + v_{i-j+1})^\top}{\sqrt{d}}. \end{aligned} \tag{8.5}$$

During training, we minimize this loss:

$$L(\theta) = \mathop{\mathbb{E}}_{\substack{\boldsymbol{x} \sim \mathcal{P} \\ \mathcal{P} \sim \text{Dir}(\alpha)^{\otimes k}}} \left[ \frac{1}{t} \sum_{p=1}^{t} l\left(TF(\boldsymbol{x}; \theta)_p, x_{p+1}\right) \right], \tag{8.6}$$

where $\theta$ denotes the parameters of the model and $l$ is a loss function such as the cross entropy or margin loss. For our experiments, we run on the standard cross-entropy loss. For our theoretical results, we analyze training under the margin loss that is a generalization of the hinge loss to the case

*For simplicity of notation we assume embedding dimension equals the hidden dimension, in general there can be a hidden dimension

of more than 2 classes: for hyperparameter $\Delta > 0$,

$$l_M(f(e)_{p,:}, x_{p+1}) = \frac{1}{k} \sum_{\substack{i=1, \\ i \neq x_{p+1}}}^{k} \max\left\{0, \Delta + f(e)_{p,i} - f(e)_{p,x_{p+1}}\right\}. \tag{8.7}$$

We now show how a two-layer transformer can represent the optimal bigrams solution.

**Proposition 8.2** (Transformer Construction)**.** *A single-head two layer attention-only transformer can find the bigram statistics in the in-context learning markov chain task.*

*Proof.* Set the internal dimension $d = 3k$, and choose $\mathbf{e}_{\boldsymbol{x}}$ to be one-hot embeddings—that is, $\mathbf{e}_{\boldsymbol{x}_i} = \delta_{\boldsymbol{x}_i}$, where $\delta$ is the Kronecker delta. We will call the parameters of attention layer $i$, $W_Q^{(i)}, W_K^{(i)}, W_V^{(i)}, v^{(i)}$. Let

$$v^{(1)} = \begin{pmatrix} \delta_2 1_k^\top \\ 0 \\ 0 \end{pmatrix} \quad W_Q^{(1)} = \begin{pmatrix} cI^{k\times k} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad W_K^{(1)} = 0 \quad W_V^{(1)} = \begin{pmatrix} 0 & I^{k\times k} & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

So,

$$A_{i,j}^{(1)} = \frac{(e_i W_Q^{(1)})(v_{i-j+1}^{(1)})^\top}{\sqrt{d}}.$$

Notice that $A_{i,j}^{(1)} = c1[j = i-1]$.So, $\text{softmax}(\text{mask}(A))_{i,j}^{(1)} \approx 1[j = i-1]$ for large enough $c$. So, for any $2 \leq i < t, 1 \leq j < k, Attn_1(e)_{i,j+k} = e_{i-1,j}$. Effectively, the first layer appends the embedding of the previous token after the embedding of the current token, so that the output at position $i$ is approximately $\begin{pmatrix} e_{x_i} & e_{x_{i-1}} & 0 \end{pmatrix}$.

The second layer is defined as follows:

$$v^{(2)} = 0 \quad W_Q^{(2)} = \begin{pmatrix} cI^{k\times k} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad W_K^{(2)} = \begin{pmatrix} 0 & 0 & 0 \\ cI^{k\times k} & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad W_V^{(2)} = \begin{pmatrix} 0 & 0 & I^{k\times k} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Note that $z = e + Attn_1(e)$, then

$$A_{i,j}^{(2)} = \frac{(z_i W_Q^{(2)})(z_j W_k^{(2)})^\top}{\sqrt{d}} = \frac{ce_i(e_{j-1})^\top}{\sqrt{d}} = \frac{c}{\sqrt{d}} 1[x_{j-1} = x_i].$$

So, for all $j < i$, $\text{softmax}(\text{mask}(A))_{i,j} \approx 1[j = i-1]$ for large enough $c$. For any $2 \le i < t, 1 \le j < k$,

$$Attn_2(e)_{i,j+2k} = \sum_{h=1}^{3k} 1[x_{h-1} = x_i](zW_V^{(2)})_{h,j} = \sum_{h=1}^{k}\sum_{g=1}^{k} 1[x_{h-1} = x_i]1[x_h = j].$$

Which is exactly the empircal bigram statistics (that is, the number of times $x_i \to j$ appears before position $i$), so to make this the output, $P = \begin{pmatrix} 0 \\ 0 \\ I^{k\times k} \end{pmatrix}$ * □

Simplified Transformer Architecture. As we see from the construction, there are two main ingredients in the solution realized by the transformer; (1st layer) the ability to look one token back and (2nd layer) the ability to attend to itself. For this reason, we define a *minimal model* that is expressive enough to be able to represent such a solution, but also simple enough to be amenable to

*Technically, the output of this construction is not the log probabilities as generally cross-entropy loss assumes. These can be approximated linearly by setting $P = \begin{pmatrix} b1^\top 1 \\ 0 \\ aI^{k\times k} \end{pmatrix}$ to change the output from $x$ to $ax + b$. In practice, this approximation can achieve close to Bayes optimal loss.

analysis. Let $e_{x_p}$ denote the one-hot embedding that corresponds to the state at position $p \in [t]$, and let $E$ be the $\mathbb{R}^{t\times k}$ one-hot embedding matrix. Then the model is defined as:

$$f(E) = \text{mask}\left(EW_k(ME)^T\right)E, \text{ where } M = \begin{pmatrix} v_1 & 0 & \dots & 0 \\ v_2 & v_1 & \dots & 0 \\ \vdots & \vdots & \dots & \vdots \\ v_t & v_{t-1} & \dots & v_1 \end{pmatrix} \in \mathbb{R}^{t\times t} \text{ and } W_k \in \mathbb{R}^{k\times k}. \tag{8.8}$$

Here mask $(\cdot)$ is a causal mask. Notice that the role of $W_k$ is to mimic the attention mechanism of the second layer and the role of $v$ is that of the positional embeddings.

*Fact* 8.3. Both the bigrams strategy and the unigrams strategy can be expressed by the minimal model with a simple choice of weights.

- *Bigrams:* For $v = (0, 1, 0 \dots, 0)^\top$, $W_k = I_{k\times k}$, we have $f(E)_{p,i} = \sum_{t'=2}^{p} 1\{x_{t'} = i\}\, 1\{x_{t'-1} = x_p\}$.
- *Unigrams:* For $v = (1, 0, 0 \dots, 0)^\top$, $W_k = 11^T$, we have $f(E)_{p,i} = \sum_{t'=1}^{p} 1\{x_{t'} = i\}$.

## 8.3 Empirical Findings and Theoretical Validation

In this section, we present our empirical findings on how transformers succeed in in-context learning Markov Chains, we demonstrate the different learning stages during training and the sudden transitions between them, and draw analytical and empirical insights from the minimal model.

### 8.3.1 Transformers In-Context Learn Markov Chains Hierarchically

As can be seen in Figure 8.3, all the models converge near the Bayesian optimal solution, suggesting that they learn to implement the bigram strategy. Curiously, however, the learning seems to be happening in stages; there is an initial rapid drop and the model quickly finds a better than random

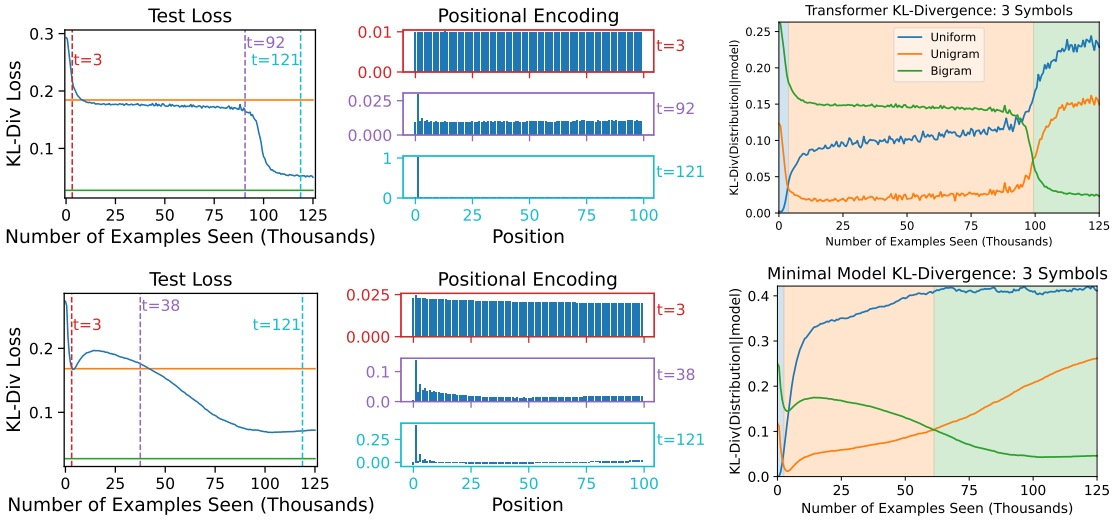



**Figure 8.3:** A two layer transformer (**top**) and a minimal model (**bottom**) trained on our in-context Markov Chain task. A comparison of the two layer attention-only transformer and minimal model (8.8) (with $v$ having constant uniform initialization, and $W_K$ initialized to 0). The graphs on the left are test loss measured by KL-Divergence from the underlying truth. The green line shows the loss of the unigram strategy, and the orange line shows the loss of the bigram strategy. The middle graph shows the effective positional encoding (for the transformer, these are for the first layer, and averaged over all tokens). The graph on the right shows the KL-divergence between the outputs of the models and three strategy. The lower the KL-divergence, the more similar the model is to that strategy.

solution. Afterwards, there is a long period of only slight improvement before a second rapid drop brings the model close to the Bayes optimal loss. We observe that training a 1-layer transformer fails to undergo a phase transition or converge to the right solution - see Figure E.8.

Interestingly, as can be seen from the horizontal lines in Figure 8.3, the intermediate plateau corresponds to a phase when the model reaches the unigram baseline. We provide evidence that this is not a coincidence, and that after the initial drop in loss, the model's strategy is very similar to the unigram strategy, before eventually being overtaken by the bigram strategy. Some of the strongest such evidence is on the right in Figure 8.3, where we plot the KL divergence between model's prediction and the two different strategies. For both the strategies, their KL divergence from the model quickly goes down, with the unigram solution being significantly lower. Around the point of the second loss drop, the KL divergence between the model and the bigram solution decreases, while the other

one increases, making it clear that the model transitions from the one solution to the other. This final drop is what has been associated to prior work with *induction heads* formation (Olsson et al., 2022); special dedicated heads inside a transformer are suddenly being formed to facilitate in-context learning.

Mechanistic Evidence For Solutions Found By Transformer. To confirm how the two layer attention-only transformer solves ICL-MC, we inspected the attention in each layer throughout training. Figure 8.2 shows the attention for a particular input during different parts of training. By the end of training, the attention consistently matches that of our construction, with the first layer attending to tokens one in the past, and the second layer attending to tokens that follow the same token as the current one. Note that even if the second layer of the transformer is mostly the same as at the end of training, if the first layer is different, then the weights shown for the second layer attention could differ dramatically. See also Figure E.6 in the Appendix that displays how the models perform on different parts of the distribution during training.

Varying the data distribution - Unigrams slow down learning There are several interesting phenomena in the learning scenario that we just described, but it is the second drop (and the preceding plateau) that warrants the most investigation. In particular, one can ask the question: is the unigram solution helpful for the eventual convergence of the model, or is it perhaps just a by-product of the learning procedure?

To answer these questions, we define distributions over Markov chains that are in between the distribution where unigrams is Bayes optimal, and the distribution where unigrams is as good as uniform. As we see in Figure 8.4a, the transformers that are being trained on the distribution where there is no unigrams "signal" train much faster.

### 8.3.2 Theoretical Insights from the Minimal Model

To abstract away some of the many complicated components from the transformer architecture, we focus our attention now to the minimal model of Section 8.2.3. We train minimal models of eq. (8.8), starting from a deterministic constant initialization, by minimizing the cross entropy loss with sgd. Full experimental details can be found in the Appendix. Figure 8.3 (bottom) displays the training curves for the minimal model. Similar to the transformer, the model learns to converge to the bigrams solution, spending however significantly less time, if any, to the unigram solution - even though they can represent it.

We now provide theoretical insights on how training progresses stage by stage and how this is achieved by the synergy between the two layers. As it turns out, there need to be at least two steps of gradient descent in order for both elements of the solution to be formed. The following lemma quantifies this.

**Lemma 8.4.** *Let the model defined as in eq.* (E.2) *and initialized with* $W_k = c11^T, v = c1^T$. *Then, the after one step of stochastic gradient descent on the margin loss of eq.* (8.7) *we have:*

$$W_k^{(1)} = \begin{pmatrix} c & c \\ c & c \end{pmatrix} + c\eta \left[ O(t^2) \begin{pmatrix} B & A \\ A & B \end{pmatrix} + O(t) \right] \qquad v_j^{(1)} = c + \frac{c\eta}{t} \left[ \frac{(t-j+1)(t-j+2)}{2} D + O(t) \right], j \in [t]$$

*where* $A, B, D > 0$ *with* $B \approx 4A$ *(diagonal bias) and* $\eta$ *is the learning rate. After the second step,* $v_2^{(2)}$ *becomes dominant, i.e.* $v_2^{(2)} > v_j^{(2)}, j = 1, 3, 4, \ldots, t$.

See Appendix E.1 for the proof. We see that the gradient of $W_k$ at initialization has a clear diagonal bias, and so it starts driving the 2nd layer towards the correct solution - see Constructions in subsection 8.2.3. The gradient of the positional embeddings, $v$, however, is rather "uninformative" to the correct coordinate that contains the one back position. Instead, it mostly favors the first po-

sition (look at current token), and has a quadratic decay. It is only after the second step that the 1st layer also starts realizing the solution, by growing $v_2^{(2)}$ - see Figure 8.4b.

There are several interesting observations that follow from Lemma 8.4 and can help us understand better the two stages of learning - both in the toy model and in the transformer:

2ND LAYER IS LEARNT FIRST It has been observed before in a similar bigram learning setting with a two-layer transformer that the model might be learning first the 2nd layer (Bietti et al., 2023). We also make similar observations in our experiments with the minimal model and the transformers (see figure 8.2), although the evidence is not so clear in the latter case. For the minimal model, the gradient calculations, clearly suggest that starting from a default initialization, it is only the 2nd layer that quickly "picks up" the right solution.

PASSING BY UNIGRAMS. The gradient of the positional embeddings $v$ at initialization delivers no information about the optimal solution, but has a quadratic structure instead that mostly favors $v_1$. If the gradient of the 2nd layer is small in scale (which can happen either due to the initialization of $v$ or to a small learning rate), then $W_k$ will not deviate much from the uniform initialization. Notice, then, that a uniform $W_k$ together with a biased first coordinate of $v$ correspond precisely to the unigram solution in this model. Thus, it is clear in this case that $W_k$ not being aligned with the solution yet, not only slows down $v$, but also biases it towards a simpler solution. We believe that the same mechanism, perhaps manifested differently, is also responsible for the hierarchical learning in the transformer.

EVEN/ODD PATTERN. An interesting observation that comes out from the calculations is the form of $v_j$ after 2 steps; the gradient amplifies much more the even coordinates than the odd ones, with a scale that follows a geometric series. In fact, the ratio is closely related to the moments of the eigenvalue of the transition matrix. This way the second coordinate starts growing larger than the

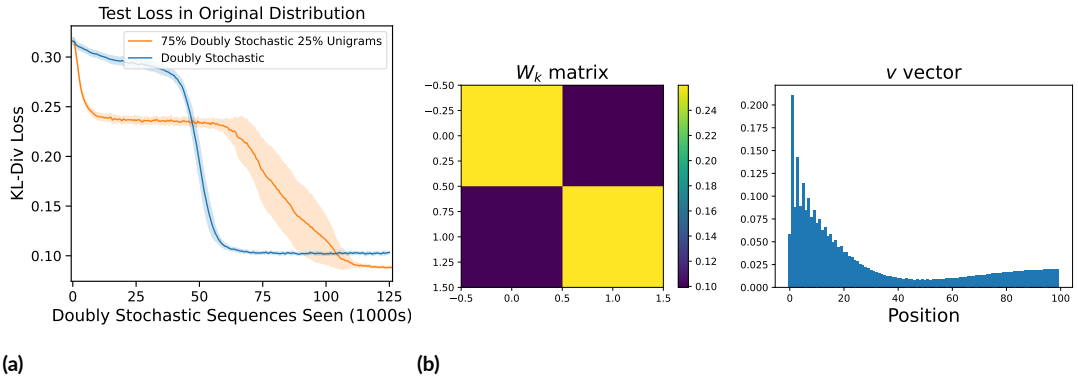



**Figure 8.4:** (**a**) Training the transformer on two different distributions (with $k = 4$ symbols). In one distribution, the transition matrix is a uniformly random doubly stochastic matrix, leading to the unigram algorithm being as bad as random, and in the Unigrams distribution each row of the matrix is the same uniformly random sample on the probability simplex, so the unigram algorithm is Bayes optimal (which is when every row of the transition matrix is the same). The second distribution is a mixture of the doubly stochastic and unigrams distribution. We then compare their loss on the full distribution, with the x-axis as the number of doubly-stochastic samples seen. (**b**) Training of the minimal model in In-Context Learning Markov Chains with $k = 2$ states. (**left**) The heatmap of the 2nd layer ($W_k$ matrix) that learns to be close to diagonal. (**right**) The values of the positional embeddings (1st layer) that display the curious even/odd pattern. Timestep corresponds to a phase when the model has started implementing the bigrams solution, but has not converged yet.

rest and the model eventually learns to represent the bigrams solution. As a byproduct, however, the rest of the even coordinates also grow in magnitude, despite not being part of the optimal solution. Perhaps surprisingly, we are able to also identify the same spurious pattern in the transformers! In the transformer, if the positional embeddings are dominating attention (that is, if $W_Q$ is unimportant in that layer), we can define an analogue of the minimal model: $\hat{v}_i = \text{softmax}(e_{x_i} W_k v^T)$. Unlike $v$ in the minimal model, $\hat{v}_i$ depends on the token at position $i$, but in practice, during most of training, $\hat{v}_i$ is similar regardless of the value of $x_i$. As can be seen on the top of Figure 8.3 (see also Figure E.7), shortly before the second drop in the loss as the induction head forms around $t = 92$, the positional embeddings start showing the same even/odd pattern for the first positions.

UNIGRAM SOLUTION SLOWS DOWN LEARNING FULL SOLUTION. By tweaking the data distribution, we can probe different parts of learning. In particular, when we focus on a special case of

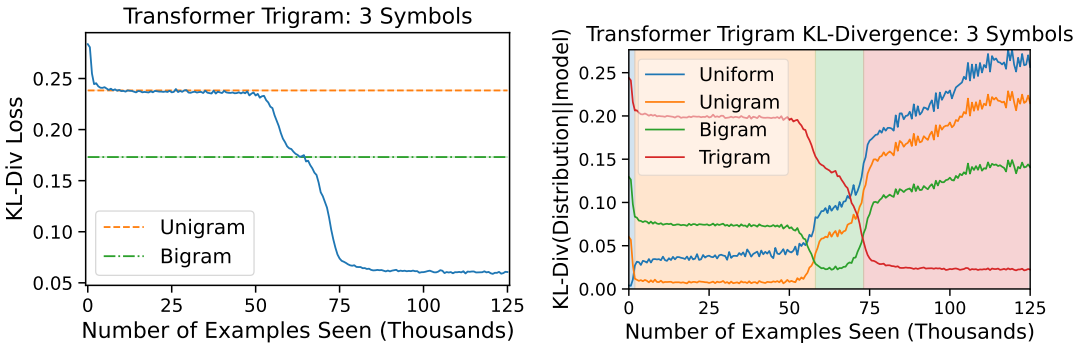



**Figure 8.5:** Three-headed transformer trained on In-Context Learning 3-grams (trigrams), with context length 200. **Left:** Loss during training. The model hierarchically converges close to the Bayes optimal solution. **Right:** KL divergence between the model and different strategies during training. As we observe, there are 4 stages of learning, each of them corresponding to a different algorithm implemented by the model.

distribution where there is no unigrams solution, transformers learn faster and do not plateau for long in the intermediate stage, and even more telling, giving additional "unigram samples" for free slows down learning - see Figure 8.4a. The calculations also seem to confirm the first observation (Corollary E.2 in the Appendix): the gradient of $v$ at the first step evaluated on this special distribution shows that, in contrast to the default case, there is no "heavy" preference towards the first coordinate. Then, once there is enough signal from the second layer, training progresses much more smoothly.

We now show that two-stage learning is not only necessary in some sense, but also sufficient to reach the bigrams solution. We analyze the optimization process of the minimal model and discover, that much like in the experiments with the transformers, there are separate phases that correspond to learning of different parts of the model. In particular, our analysis is in a simplified setting where the data distribution changes in the second step; the transition matrices are no longer sampled uniformly, but they are instead the "hard" examples of this task and contain the most "information" about the solution. This idea is closely related to the technique of curriculum learning (Bengio et al., 2009) and is naturally motivated in many settings. In this setting, we are able to show that two

steps of gradient descent will reach the bigrams solution.

**Proposition 8.5.** *(Informal) Consider the minimal model of eq.* (8.8) *being trained with online SGD on the margin loss* (8.7) *with in-context loss* (8.6)*. With appropriate choice of learning rates and weight decay in the 2nd layer, two steps of gradient descent with curriculum learning reach the bigrams solution.*

The idea of the proof is that a first step of gradient descent with a small learning rate can align the 2nd layer, while a second step on "hard" examples can learn to identify perfectly the relevant relative embedding. The formal statement and its proof can be found in Appendix E.1. An interesting by-product of the analysis is that the learning rate at the first step needs to be an order of magnitude larger than at the second step, perhaps explaining why the second phase of learning in the experiments takes significantly more time. It is worth noting that, while this is a simplified setting, it goes beyond NTK-based (Jacot et al., 2018) analyses where the representations do not change much and it crucially involves more than one step which has been a standard tool in the analysis of feature learning in deep learning (Ba et al., 2022).

### 8.3.3 Beyond Bigrams: $n$-gram Statistics

We also investigated the performance of transformers on learning in-context $n$-grams for $n > 2$; in particular, 3-grams. We trained attention-only transformers with three heads in each layer by minimizing the in-context cross entropy loss with the Adam optimizer. As can be seen in Figure 8.5 (left), the model eventually converges to the Bayes optimal solution. Interestingly, as in the case of Markov Chains, the model displays a "hierarchical learning" behavior characterized by long plateaus and sudden drops. In this setup, the different strategies correspond to unigrams, bigrams and trigrams, respectively. This is presented clearly on the right of Figure 8.5, where we plot the similarity of the model with the different strategies and it exhibits the same clear pattern as in the case of

$n = 2$. Single attention headed models could not achieve better performance than bigrams. We leave a more thorough investigation for future work.

## 8.4 Conclusion and Discussion

In this work, we have introduced a simple learning problem which serves as a controlled setting for understanding in-context learning and the emergence of (statistical) induction heads. Induction heads are a common motif in LLMs, which suggests the natural direction of exploring to what extent our various findings extend to natural language training data. Notably, on the way to solving the ICL-MC task, networks pass through a sequence of well-characterized discrete incomplete solutions. Simple but incomplete solutions may be commonplace in language modeling and other rich learning settings; for any such solution, one can ask to what extent its presence speeds up or slows down the formation of more complex circuits with higher accuracy.

# A

# Variable Creation

## A.1 Proofs of capacity bounds

In this section we present the full proofs (including the omitted proofs) of our capacity bounds. We also cover relevant background and useful technical lemmas.

### A.1.1 Rademacher complexity and generalization bounds

Here we briefly review Rademacher complexity and its relationship to covering numbers and generalization bounds. We refer the reader to Bartlett & Mendelson (2002) for a more detailed exposition.

**Definition A.1** (Empirical Rademacher complexity)**.** For a given class of functions $\mathcal{F} = \{f : \mathcal{X} \to \mathbb{R}\}$ and $\{z^{(i)} \in \mathcal{X}\}_{i=1}^m$, the empirical Rademacher complexity $\widehat{\mathcal{R}}(\mathcal{F}; z^{(1)}, \dots, z^{(m)})$ is defined as

$$\widehat{\mathcal{R}}(\mathcal{F}; z^{(1)}, \dots, z^{(m)}) = \frac{1}{m} \mathbb{E}_\varepsilon \left[ \sup_{f \in \mathcal{F}} \sum_{i=1}^m \varepsilon_i f(z^{(i)}) \right],$$

where $\varepsilon$ is a vector of $m$ i.i.d. Rademacher random variables ($\Pr[\varepsilon_i = 1] = \Pr[\varepsilon_i = -1] = 1/2$).

In order to relate the Rademacher complexity and $\ell_\infty$-covering numbers, we use a modified version of Dudley's metric entropy.

**Lemma A.2** (Dudley (1967); modified)**.** *Consider a real-valued function class $\mathcal{F}$ such that $|f| \le A$ for all $f \in \mathcal{F}$. Then*

$$\widehat{\mathcal{R}}(\mathcal{F}; z^{(1)}, \dots, z^{(m)}) \le c \cdot \inf_{\delta \ge 0} \left( \delta + \int_\delta^A \sqrt{\frac{\log \mathcal{N}_\infty(\mathcal{F}; \varepsilon; z^{(1)}, \dots, z^{(m)})}{m}} \, d\varepsilon \right)$$

*for some constant $c > 0$.*

*Proof sketch.* The original statement is for 2-norm covering number, but the $\infty$-norm case reduces to the 2-norm case because $N_2(\cdot) \le N_\infty(\cdot)$. The original statement also fixes $\delta = 0$ rather than

taking an infimum. Also, the standard statement has the integral going from 0 to $\infty$, but these are easily replaced with $\delta$ and $A$. □

For our paper, we will instantiate the above lemma for log covering numbers scaling as $1/\varepsilon^2$.

**Corollary A.3** (Rademacher complexity via covering number)**.** *Consider a real-valued function class $\mathcal{F}$ such that $|f| \leq A$ for all $f \in \mathcal{F}$. Suppose $\log \mathcal{N}_\infty(\mathcal{F}; \varepsilon; z^{(1)}, \ldots, z^{(m)}) \leq C_\mathcal{F}/\varepsilon^2$, then*

$$\widehat{\mathcal{R}}(\mathcal{F}; z^{(1)}, \ldots, z^{(m)}) \leq c \cdot \sqrt{\frac{C_\mathcal{F}}{m}} \cdot \left(1 + \log\left(A\sqrt{m/C_\mathcal{F}}\right)\right)$$

*for some constant $c > 0$.*

*Proof.* Using Lemma A.2, we have for some constant $c > 0$,

$$\begin{aligned}
\widehat{\mathcal{R}}(\mathcal{F}; z^{(1)}, \ldots, z^{(m)}) &\leq c \inf_{\delta \geq 0} \left( \delta + \int_\delta^A \sqrt{\frac{\log \mathcal{N}_\infty(\mathcal{F}; \varepsilon; z^{(1)}, \ldots, z^{(m)})}{m}}\, d\varepsilon \right) \\
&\leq c \inf_{\delta \geq 0} \left( \delta + \int_\delta^A \sqrt{\frac{C_\mathcal{F}}{\varepsilon^2 m}}\, d\varepsilon \right) \\
&= c \inf_{\delta \geq 0} \left( \delta + \sqrt{\frac{C_\mathcal{F}}{m}} \int_\delta^A \frac{1}{\varepsilon}\, d\varepsilon \right) \\
&= c \inf_{\delta \geq 0} \left( \delta + \sqrt{\frac{C_\mathcal{F}}{m}} \log(A/\delta) \right) \\
&= c\sqrt{\frac{C_\mathcal{F}}{m}} \left(1 + \log\left(A\sqrt{m/C_\mathcal{F}}\right)\right).
\end{aligned}$$

□

We can now obtain a generalization guarantee from the Rademacher complexity of a function class:

**Theorem A.4** (Bartlett & Mendelson (2002)). *Let $\mathcal{D}$ be a distribution over $\mathcal{X} \times \mathbb{R}$ and let $\ell : \mathbb{R} \times \mathbb{R}$ be a $b$-bounded loss function that is $L$-Lipschitz in its first argument. For a given function class $\mathcal{F}$ and $f \in \mathcal{F}$, let $\mathsf{risk}(f; \mathcal{D}) := \mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell(f(x), y)]$ and $\widehat{\mathsf{risk}}\left(f; (z^{(i)}, y^{(i)})_{i=1}^m\right) := \frac{1}{m}\sum_{i=1}^m \ell(f(z^{(i)}), y^{(i)})$. Then for any $\delta > 0$, with probability at least $1 - \delta$, simultaneously for all $f \in \mathcal{F}$,*

$$\left|\mathsf{risk}(f; \mathcal{D}) - \widehat{\mathsf{risk}}\left(f; (z^{(i)}, y^{(i)})_{i=1}^m\right)\right| \leq 4L\,\widehat{\mathcal{R}}\left(\mathcal{F}; z^{(1)}, \ldots, z^{(m)}\right) + 2b\sqrt{\frac{\log(1/\delta)}{2m}}.$$

Combining the above, we get:

**Lemma A.5** (Lemma 4.2 (restated)). *Consider a function class $\mathcal{F}$ such that $|f| \leq A$ for all $f \in \mathcal{F}$ and $\log \mathcal{N}_\infty(\mathcal{F}; \varepsilon; x^{(1)}, \ldots, x^{(m)}) \leq C_\mathcal{F}/\varepsilon^2$ for all $x^{(1)}, \ldots, x^{(m)} \in \mathcal{X}^m$. Then for any $\delta > 0$, with probability at least $1 - \delta$, simultaneously for all $f \in \mathcal{F}$,*

$$\left|\mathsf{risk}(f; \mathcal{D}) - \widehat{\mathsf{risk}}\left(f; (x^{(i)}, y^{(i)})_{i=1}^m\right)\right| \leq 4cL\sqrt{\frac{C_\mathcal{F}}{m}}\left(1 + \log\left(A\sqrt{m/C_\mathcal{F}}\right)\right) + 2b\sqrt{\frac{\log(1/\delta)}{2m}},$$

*for some constant $c > 0$.*

### A.1.2 Useful lemmas

**Lemma A.6.** *Consider function $f : \mathbb{R}^d \to \Delta^{d-1}$ such that the Jacobian of the function satisfies $\|Jf(\theta)\|_{1,1} \leq c_f$ for all $\theta \in \mathbb{R}^d$, then for any vectors $\theta_1, \theta_2 \in \mathbb{R}^p$,*

$$\|f(\theta_1) - f(\theta_2)\|_1 \leq c_f\|\theta_1 - \theta_2\|_\infty.$$

*Proof.* By the fundamental theorem of calculus applied to $g(t) = f(t\theta_1 + (1-t)\theta_2)$, followed by a change of variables:

$$f(\theta_1) - f(\theta_2) = \left(\int_0^1 J(t\theta_1 + (1-t)\theta_2)\,dt\right)(\theta_1 - \theta_2),$$

We have

$$\|f(\theta_1) - f(\theta_2)\|_1 = \left\| \int_0^1 J(t\theta_1 + (1-t)\theta_2)\,(\theta_1 - \theta_2) dt \right\|_1$$

By Jensen's inequality:

$$\leq \int_0^1 \|J(t\theta_1 + (1-t)\theta_2)\,(\theta_1 - \theta_2)\|_1 \, dt$$

Using $\|Ax\|_1 \leq \|A\|_{1,1} \|x\|_\infty$:

$$\leq \int_0^1 \|J(t\theta_1 + (1-t)\theta_2)\|_{1,1} \|\theta_1 - \theta_2\|_\infty \, dt$$

By assumption on the Jacobian:

$$\leq c_f \|\theta_1 - \theta_2\|_\infty .$$

□

**Corollary A.7.** *For vectors $\theta_1, \theta_2 \in \mathbb{R}^p$, $\|\mathsf{softmax}(\theta_1) - \mathsf{softmax}(\theta_2)\|_1 \leq 2\|\theta_1 - \theta_2\|_\infty$.*

*Proof.* Observe that for softmax, the Jacobian satisfies:

$$J(\theta) = \mathrm{diag}(\mathsf{softmax}(\theta)) - \mathsf{softmax}(\theta)\mathsf{softmax}(\theta)^\top.$$

We have for all $\theta$, $h$,

$$\begin{aligned}
\|J(\theta)\|_{1,1} &= \sum_{i=1}^{p} \sum_{j=1}^{p} \left| \mathsf{softmax}(\theta)_i (1[i=j] - \mathsf{softmax}(\theta)_j) \right| \\
&= \sum_{i=1}^{p} \mathsf{softmax}(\theta)_i \left( 1 - \mathsf{softmax}(\theta)_i + \sum_{j \neq i} \mathsf{softmax}(\theta)_j \right) \\
&= 2 \sum_{i=1}^{p} \mathsf{softmax}(\theta)_i \left(1 - \mathsf{softmax}(\theta)_i\right) \\
&\leq 2.
\end{aligned}$$

Combining the above with Lemma A.6 gives the desired result. □

**Lemma A.8.** *For $\alpha_i, \beta_i \geq 0$, the solution to the following optimization*

$$\min_{x_1,\ldots,x_n} \sum_{i=1}^{n} \frac{\alpha_i}{x_i^2}$$
$$\text{subject to} \sum_{i=1}^{n} \beta_i x_i = C$$

*is $\frac{\gamma^3}{C^2}$ and is achieved at $x_i = \frac{C}{\gamma}\left(\frac{\alpha_i}{\beta_i}\right)^{1/3}$ where $\gamma = \sum_{i=1}^{n} \alpha_i^{1/3}\beta_i^{\frac{2}{3}}$.*

*Proof.* The proof follows by a standard Lagrangian analysis. □

**Lemma A.9** (Contractivity of $\Pi_{\mathsf{norm}}$)**.** *Let $\Pi_{\mathsf{norm}}$ be the projection operator onto the unit norm ball. For any vectors $u, v$, we have $\|\Pi_{\mathsf{norm}}(u) - \Pi_{\mathsf{norm}}(v)\| \leq \|u - v\|$.*

*Proof.* If $u, v$ are both in the unit ball then this follows trivially. Let us assume that $\|u\| \geq \|v\|$ and $\|u\| \geq 1$ WLOG. First suppose $\|v\| \leq 1$. Let $B_V^{(1)} = \alpha u$ be the projection of $v$ in the direction of $u$, and let $B_V^2 = v - B_V^{(1)}$. Then

$$\begin{aligned}
\|\Pi_{\mathsf{norm}}(u) - \Pi_{\mathsf{norm}}(v)\|^2 &= \|u/\|u\| - v\|^2 \\
&= \|u/\|u\| - (\alpha u + B_V^2)\|^2 \\
&= \|(\|u\|^{-1} - \alpha)u - B_V^2\|^2 \\
&= (\|u\|^{-1} - \alpha)^2\|u\|^2 + \|B_V^2\|^2 \\
&\leq (1 - \alpha^2)\|u\|^2 + \|B_V^2\|^2 \qquad \text{since } \|u\|^{-1} < \alpha < 1 \\
&= \|u - (\alpha u + B_V^2)\|^2 \\
&= \|u - v\|^2
\end{aligned}$$

If $\|v\| > 1$, then

$$\|\Pi_{\mathsf{norm}}(u) - \Pi_{\mathsf{norm}}(v)\| = \|\Pi_{\mathsf{norm}}(u/\|v\|) - \Pi_{\mathsf{norm}}(v/\|v\|)\| \leq \|u/\|v\| - v/\|v\|\| < \|u - v\|.$$

where the second-to-last inequality follows from the $\|v\| < 1$ case. □

**Lemma A.10** (Zhang (2002), Theorem 4). *Let* $\mathcal{V} : \{v : v \in \mathbb{R}^{d_1}, \|v\| \leq B_1\}$ *and* $\mathcal{F}_{linear} = \{x \mapsto v^\top x : v \in \mathcal{V}\}$. *For any* $\delta > 0$ *and* $x^{(1)}, \ldots, x^{(N)}$ *satisfying* $\|x^{(i)}\| \leq B_2 \; \forall i$,

$$\log \mathcal{N}_\infty(\mathcal{F}_{linear}; \varepsilon; x^{(1)}, \cdots, x^{(N)}) \leq 36 \frac{B_1^2 B_2^2}{\varepsilon^2} \log(2\lceil 4B_1B_2/\varepsilon + 2\rceil N + 1).$$

### A.1.3 Pseudo-dimension lower bound

Recall that a Transformer self-attention head is of the form

$$f_{\mathsf{tf\text{-}head}}(X; W_V, W_{QK}) := \sigma\left(W_V^\top X^\top \mathsf{softmax}\left(X W_{QK}^\top x_\tau\right)\right)$$

and that the *pseudo-dimension* of a function class $\mathcal{F}$ is defined as

$$\mathrm{Pdim}(\mathcal{F}) := \max_{c \in \mathbb{R}} \mathrm{VCdim}(\{h(\cdot) = \mathrm{sign}(f(\cdot) - c) : f \in \mathcal{F}\})$$

For the purposes of this proof, we will treat $x_\tau$ as a fixed vector—so either it is not treated as part of the input, or it is set to the same value for all of the inputs. Thus, $W_{QK}^\top x_\tau$ is a fixed vector, which we call $w_{QK}$. Moreover, we will set $W_V$ to be a $d \times 1$ matrix, so we will treat it as a vector $w_V \in \mathbb{R}^d$. Also, so long as the activation function is non-decreasing and non-constant, it does not affect the pseudo-dimension and can be safely ignored. Thus, we will deal with the simplified class $\mathcal{F}$ of functions of the form:

$$f(X; w_V, w_{QK}) := w_V^\top X^\top \mathsf{softmax}\left(X w_{QK}\right)$$

for $w_V, w_{QK} \in \mathbb{R}^d$.

*Proof of Proposition 4.8.* For simplicity, we consider the case where $T$ is a power of 2. We will construct a set of $\log T$ inputs $\{X^{(i)}\}_{i=1}^{\log T}$ in $\mathbb{R}^T$ that are shattered by $\mathcal{F}$.

We will construct a shattering using the threshold $c = 1/2$, which gives us a lower bound on the pseudo-dimension:

$$\text{Pdim}(\mathcal{F}) \geq \text{VCdim}(\{h(\cdot) = \text{sign}(f(\cdot) - 1/2) : f \in \mathcal{F}\}).$$

In particular, we will shatter the set $\mathcal{X} = \{X^{(i)} : i \in [\log T]\}$ of sequences defined by

$$x_t^{(i)} := (\cos(2\pi t/T), \sin(2\pi t/T), \text{bin}(t)_i).$$

Here we have indexed $t = 0, \ldots, T-1$, and $\text{bin}(t)_i$ is defined to be the $i$th bit of the binary expansion of the integer $t$ (padded with 0s in the front such that the expansion is length $\log T$). We can think of the first two coordinates of $x_t^{(i)}$ as a (fixed) circular "positional encoding", and the third coordinate as the "token embedding". The token embedding is designed such that $\mathcal{X}$ is shattered by the function class $\mathcal{P} = \{\text{proj}^{(s)}(X) : s = 0, \ldots T-1\}$ of projection functions

$$\text{proj}^{(s)}(X) = x_s[3].$$

We will design the attention matrices to approximate the functions in $\mathcal{P}$.

For $s = 0, \ldots T-1$, let

$$w_{QK}^{(s)} := (T^2 \cos(2\pi t/T), T^2 \sin(2\pi t/T), 0).$$

The $t$th attention weight ${w_{QK}^{(s)}}^\top x_t^{(i)}$ is maximized when $t = s$. In fact, $w_{QK}^{(s)}$ is of sufficiently large

magnitude that the token $x_s^{(i)}$ is attended to more strongly than all the other positions combined.

Also, let

$$w_V^{(s)} := (0, 0, 1).$$

This projects onto the token embedding coordinate.

Observe that

$$w_{QK}^{(s)\top} x_t^{(i)} = T^2 \cos(2\pi(s-t)/T),$$

so we have

$$\mathsf{softmax}(Xw_{QK}^{(s)})[t] = \frac{\exp(T^2\cos(2\pi(s-t)/T))}{\sum_{\tau=0}^{T-1}\exp(T^2\cos(2\pi(s-\tau)/T))} = \frac{\exp(T^2\cos(2\pi(s-t)/T))}{\sum_{\tau=0}^{T-1}\exp(T^2\cos(2\pi\tau/T))}$$

Let us bound the denominator using the fact that $\cos(\theta) \le 1 - \frac{2}{\pi^2}\theta^2$ for $\theta \in [0, \pi]$:

$$\begin{aligned}
\sum_{\tau=0}^{T-1}\exp(T^2\cos(2\pi\tau/T)) &\le \exp(T^2) + 2\sum_{\tau=1}^{\lfloor (T-1)/2\rfloor}\exp(T^2\cos(2\pi\tau/T)) \\
&\le \exp(T^2) + 2\sum_{\tau=0}^{\lfloor (T-1)/2\rfloor}\exp\left(T^2\left(1 - \frac{2}{\pi^2}(2\pi\tau/T)^2\right)\right) \\
&= e^{T^2} + 2e^{T^2}\sum_{\tau=1}^{\lfloor (T-1)/2\rfloor} e^{-8\tau^2} \\
&\le e^{T^2} + 2e^{T^2}\int_{\rho=0}^{\lfloor (T-1)/2\rfloor} e^{-8\rho^2}d\rho \\
&\le e^{T^2} + 2e^{T^2}\int_{\rho=0}^{\infty} e^{-8\rho^2}d\rho \\
&= e^{T^2} + \sqrt{\frac{\pi}{8}}e^{T^2}
\end{aligned}$$

Hence, for each $s$,

$$\mathsf{softmax}(Xw_{QK}^{(s)})[s] \ge \frac{1}{1+\sqrt{\frac{\pi}{8}}} > \frac{1}{2}$$

and

$$\sum_{t \neq s} \mathsf{softmax}(X w_{QK}^{(s)})[t] \leq 1 - \frac{1}{1 + \sqrt{\frac{\pi}{8}}} < \frac{1}{2}$$

Then we claim that the set $\{f(\cdot; w_V^{(s)}, w_{QK}^{(s)})\}_{s=0}^{T-1}$ shatters $\{X^{(i)}\}_{i=1}^{\log T}$:

$$\begin{aligned} f(X^{(i)}; w_V^{(s)}, w_{QK}^{(s)}) &= {w_V^{(s)}}^\top X^\top \mathsf{softmax}\left(X^{(i)} w_{QK}^{(s)}\right) \\ &= {w_V^{(s)}}^\top \sum_{t=0}^{T-1} \mathsf{softmax}\left(X^{(i)} w_{QK}^{(s)}\right)[t] \cdot \mathrm{bin}(t)_i \end{aligned}$$

which is greater than $1/2$ if $\mathrm{bin}(s)_i = 1$ and is less than $1/2$ if $\mathrm{bin}(s)_i = 0$, since the $t = s$ term in the sum dominates all the other terms. Thus,

$$\{h(\cdot) = \mathrm{sign}(f(\cdot) - 1/2) : f \in \mathcal{F}\} = \mathcal{P},$$

so the different choices of $s$ induce a shattering. □

### A.1.4 Covering number upper bounds

*Proof of Lemma 4.7.* Observe that,

$$\begin{aligned} &\left\| f_{\mathsf{head}}(X, z; \theta_s, \theta_{\mathrm{in}}) - f_{\mathsf{head}}(X, z; \widehat{\theta}_s, \widehat{\theta}_{\mathrm{in}}) \right\| \\ &= \left\| \varphi_{\mathrm{out}}\left(\varphi_{\mathrm{in}}(X; \theta_{\mathrm{in}})^\top \mathsf{Norm}(\mathsf{Score}(X, z; \theta_s))\right) - \varphi_{\mathrm{out}}\left(\varphi_{\mathrm{in}}(X; \widehat{\theta}_{\mathrm{in}})^\top \mathsf{Norm}(\mathsf{Score}(X, z; \widehat{\theta}_s))\right) \right\| \end{aligned}$$

By $L_{\mathrm{out}}$-Lipschitzness of $\varphi_{\mathrm{out}}$ and bound on $\|w\|$:

$$\leq L_{\mathrm{out}} \left\| \varphi_{\mathrm{in}}(X; \theta_{\mathrm{in}})^\top \mathsf{Norm}(\mathsf{Score}(X, z; \theta_s)) - \varphi_{\mathrm{in}}(X; \widehat{\theta}_{\mathrm{in}})^\top \mathsf{Norm}(\mathsf{Score}(X, z; \widehat{\theta}_s)) \right\|$$

By triangle inequality:

$$\leq L_{\text{out}} \left\| \varphi_{\text{in}}(X;\theta_{\text{in}})^\top \left( \mathsf{Norm}(\mathsf{Score}(X,z;\theta_s)) - \mathsf{Norm}(\mathsf{Score}(X,z;\widehat{\theta}_s)) \right) \right\|$$
$$+ L_{\text{out}} \left\| \left( \varphi_{\text{in}}(X;\theta_{\text{in}}) - \varphi_{\text{in}}(X;\widehat{\theta}_{\text{in}}) \right)^\top \mathsf{Norm}(\mathsf{Score}(X,z;\widehat{\theta}_s)) \right\|$$

Using $\|Pv\| \leq \|P\|_{2,\infty}\|v\|_1$ and $B_{\text{in}}$-boundedness of $\varphi_{\text{in}}$:

$$\leq L_{\text{out}} B_{\text{in}} \left\| \mathsf{Norm}(\mathsf{Score}(X,z;\theta_s)) - \mathsf{Norm}(\mathsf{Score}(X,z;\widehat{\theta}_s)) \right\|_1$$
$$+ L_{\text{out}} \left\| \left( \varphi_{\text{in}}(X;\theta_{\text{in}}) - \varphi_{\text{in}}(X;\widehat{\theta}_{\text{in}}) \right)^\top \right\|_{2,\infty} \left\| \mathsf{Norm}(\mathsf{Score}(X,z;\widehat{\theta}_s)) \right\|_1$$

By Lemma A.6 and the assumption on Norm:

$$\leq L_{\text{out}} C_{\mathsf{Norm}} \left\| \varphi_{\text{in}}(X;\theta_{\text{in}})^\top \right\|_{2,\infty} \left\| \mathsf{Score}(X,z;\theta_s) - \mathsf{Score}(X,z;\widehat{\theta}_s) \right\|_\infty + L_{\text{out}} \left\| \left( \varphi_{\text{in}}(X;\theta_{\text{in}}) - \varphi_{\text{in}}(X;\widehat{\theta}_{\text{in}}) \right)^\top \right\|_{2,\infty}$$

By boundedness of $\varphi_{\text{in}}$ and $\left\|X^\top\right\|_{2,\infty} \leq B_X$:

$$\leq L_{\text{out}} C_{\mathsf{Norm}} B_{\text{in}} B_X \left\| \mathsf{Score}(X,z;\theta_s) - \mathsf{Score}(X,z;\widehat{\theta}_s) \right\|_\infty + L_{\text{out}} \left\| \left( \varphi_{\text{in}}(X;\theta_{\text{in}}) - \varphi_{\text{in}}(X;\widehat{\theta}_{\text{in}}) \right)^\top \right\|_{2,\infty}.$$

□

*Proof of Theorem 4.6.* Our goal is to show that for every $\varepsilon > 0$, collection of inputs $(X^{(1)}, z^{(1)}), \ldots, (X^{(m)}, z^{(m)})$, there is a cover $\mathcal{C}_{\mathsf{head}}$ such that for all $\theta_s \in \Theta_s, \theta_{\text{in}} \in \Theta_{\text{in}}$, there is some $(\widehat{\theta}_s, \widehat{\theta}_{\text{in}}) \in \mathcal{C}_{\mathsf{head}}$ such that $\max_i \left\| f_{\mathsf{head}}(X^{(i)}, z^{(i)}; \theta_s, \theta_{\text{in}}) - f_{\mathsf{head}}(X^{(i)}, z^{(i)}; \widehat{\theta}_s, \widehat{\theta}_{\text{in}}) \right\| \leq \varepsilon$.

Observe that for all $\theta_s, \widehat{\theta}_s$,

$$\max_{i\in[m]} \|\mathsf{Score}(X^{(i)}, z^{(i)}; \theta_s) - \mathsf{Score}(X^{(i)}, z^{(i)}; \widehat{\theta}_s)\|_\infty = \max_{i\in[m], t\in[T]} \left| \mathsf{Score}(x_t^{(i)}, z^{(i)}; \theta_s) - \mathsf{Score}(x_t^{(i)}, z^{(i)}; \widehat{\theta}_s) \right|.$$

Similarly, for all $\theta_{\text{in}}, \widehat{\theta}_{\text{in}}$,

$$\max_{i\in[m]} \left\| \left( \varphi_{\text{in}}(X^{(i)};\theta_{\text{in}}) - \varphi_{\text{in}}(X^{(i)};\widehat{\theta}_{\text{in}}) \right)^\top \right\|_{2,\infty} = \max_{i\in[m], t\in[T]} \left\| \varphi_{\text{in}}(x_t^{(i)};\theta_{\text{in}}) - \varphi_{\text{in}}(x_t^{(i)};\widehat{\theta}_{\text{in}}) \right\|.$$

This crucially allows us to aggregate over the $i$ and $t$ dimensions together.* Therefore, we can consider $\mathcal{N}_\infty$ covers for the above to bound the overall covering number.

Let $\mathcal{C}_{\mathsf{Score}}$ be the $\varepsilon_{\mathsf{Score}}$-cover $(\infty)$ for $\mathcal{F}_{\mathsf{Score}}$ over inputs $\left\{(x_t^{(i)}, z^{(i)})\right\}_{i\in[m],t\in[T]}$ of size

$$\mathcal{N}_\infty\left(\mathcal{F}_{\mathsf{Score}}; \varepsilon_{\mathsf{Score}}; \{(x_t^{(i)}, z^{(i)})\}_{i\in[m],t\in[T]}\right).$$

Also, Let $\mathcal{C}_{\text{in}}$ be the $\varepsilon_{\text{in}}$-cover $(\infty)$ for $\mathcal{F}_{\text{in}}$ over inputs $\{x_t^{(i)}\}_{i\in[m],t\in[T]}$ of size

$$\mathcal{N}_\infty\left(\mathcal{F}_{\text{in}}; \varepsilon_{\text{in}}; \{x_t^{(i)}\}_{i\in[m],t\in[T]}; \|\cdot\|_2\right).$$

We are ready to construct the cover for $\mathcal{F}_{\mathsf{head}}$. Set $\mathcal{C}_{\mathsf{head}} = \{f_{\mathsf{head}}(\cdot; \widehat{\theta}_s, \widehat{\theta}_{\text{in}}))_{i\in[m]} : \widehat{\theta}_s \in \mathcal{C}_{\mathsf{Score}}, \widehat{\theta}_{\text{in}} \in \mathcal{C}_{\text{in}}\}$. Then for any $\theta_s \in \Theta_s, \theta_{\text{in}} \in \Theta_{\text{in}}$, there exists $\widehat{\theta}_s, \widehat{\theta}_{\text{in}} \in \mathcal{C}_{\mathsf{head}}$, such that for all $i \in [m]$, using Lemma 4.7:

$$\left\|f_{\mathsf{head}}(X^{(i)}, z^{(i)}; \theta_s, \theta_{\text{in}}) - f_{\mathsf{head}}(X^{(i)}, z^{(i)}; \widehat{\theta}_s, \widehat{\theta}_{\text{in}})\right\| \le C_{\mathsf{Norm}} L_{\text{out}} B_{\text{in}} B_X \varepsilon_{\mathsf{Score}} + L_{\text{out}}\varepsilon_{\text{in}}.$$

The size of the cover we have constructed is,

$$\begin{aligned}\log|\mathcal{C}_{\mathsf{head}}| &= \log|\mathcal{C}_{\mathsf{Score}}| + \log|\mathcal{C}_{\text{in}}| \\ &= \log\mathcal{N}_\infty\left(\mathcal{F}_{\mathsf{Score}}; \varepsilon_{\mathsf{Score}}; \{(x_t^{(i)}, z^{(i)})\}_{i\in[m],t\in[T]}\right) + \log\mathcal{N}_\infty\left(\mathcal{F}_{\text{in}}; \varepsilon_{\text{in}}; \{x_t^{(i)}\}_{i\in[m],t\in[T]}; \|\cdot\|_2\right)\end{aligned}$$

and we are done. □

*In the case of the Transformer self-attention mechanism, we will obtain $\infty$-norm covering numbers for Score and $\varphi_{\text{in}}$ that have only logarithmic dependence on the number of examples. Because of this aggregation trick, the resulting covering number for the whole layer will have merely logarithmic dependence on the context length $T$.

*Proof of Corollary 4.9.* By Theorem 4.6, the covering number of $\mathcal{F}_{\text{tf-head}}$ satisfies

$$\begin{aligned}
&\log \mathcal{N}_\infty \left( \mathcal{F}_{\text{tf-head}}; \varepsilon; \left\{ (X^{(i)}, z^{(i)}) \right\}_{i=1}^m \right) \\
&\leq \inf_{\alpha \in [0,1]} \left[ \log \mathcal{N}_\infty \left( \mathcal{F}_{QK}; \frac{\alpha \varepsilon}{2 L_\sigma B_V B_X}; \{ (x_t^{(i)}, z^{(i)}) \}_{i \in [m], t \in [T]} \right) \right. \\
&\quad \left. + \log \mathcal{N}_\infty \left( \mathcal{F}_V; \frac{(1-\alpha)\varepsilon}{L_\sigma}; \{ x_t^{(i)} \}_{i \in [m], t \in [T]}; \| \cdot \|_2 \right) \right].
\end{aligned}$$

where we have used the fact that for a scalar-output Transformer layer:

- softmax satisfies the Jacobian assumption with $C_{\text{softmax}} = 2$ using Corollary A.7.
- $L_{\text{out}}$ is the Lipschitz constant of $\sigma$: $L_\sigma$.
- $B_{\text{in}}$ is a bound on the norm of $W_V^\top x$ with respect to norm of $x$: $B_V$.

By Lemma 4.10, for any $\varepsilon_{QK}, \varepsilon_V > 0$:

$$\log \mathcal{N}_\infty \left( \mathcal{F}_{QK}; \varepsilon_{QK}; \{ (x_t^{(i)}, z^{(i)}) \}_{i \in [m], t \in [T]} \right) \lesssim \frac{(B_{QK}^{2,1} B_X)^2 \log(dmT)}{\varepsilon_{QK}^2}$$

$$\log \mathcal{N}_\infty \left( \mathcal{F}_V; \varepsilon_V; \{ (x_t^{(i)}, z^{(i)}) \}_{i \in [m], t \in [T]}; \| \cdot \|_2 \right) \lesssim \frac{(B_V^{2,1} B_X)^2 \log(dmT)}{\varepsilon_V^2}$$

since $W_{QK}, W_V \in \mathbb{R}^{d \times d}$ ($k = d$). We want to choose $\varepsilon_{QK}$ and $\varepsilon_V$ to minimize the sum of the above two terms, subject to

$$2 L_\sigma B_V B_X \varepsilon_{QK} + L_\sigma \varepsilon_V \leq \varepsilon.$$

By Lemma A.8, the solution to this optimization leads to an optimal bound of:

$$\log \mathcal{N}_\infty (\mathcal{F}_{\text{tf-head}}; \varepsilon; X^{(1)}, \ldots, X^{(M)}) \lesssim (L_\sigma B_X)^2 \cdot \frac{\left( (B_V^{2,1})^{\frac{2}{3}} + (B_{QK}^{2,1} B_V B_X)^{\frac{2}{3}} \right)^3}{\varepsilon^2} \cdot \log(dmT).$$

□

*Proof of Lemma 4.10.* Our approach will be to construct a cover by decomposing the problem into two separate cover problems, (1) $\ell_2$-cover over the possible norms of the rows of $W$, and (2) $\ell_\infty$-cover of the set $\mathcal{W}$ constrained to the norms dictated by the first cover. More formally, let us define:

$$\mathcal{W}_{:,2} = \left\{ \begin{bmatrix} \|w_1\| \\ \vdots \\ \|w_{d_1}\| \end{bmatrix} : W = \begin{bmatrix} w_1 \\ \vdots \\ w_{d_1} \end{bmatrix} \in \mathcal{W} \right\} = \left\{ v \in \mathbb{R}^{d_1} : \|v\|_1 \leq B_W \right\}, \text{and}$$

$$\mathcal{F}_v = \left\{ x \to Wx : W = \begin{bmatrix} w_1 \\ \vdots \\ w_{d_1} \end{bmatrix} \in \mathbb{R}^{d_1 \times d_2}, \forall i \, \|w_i\| \leq v_i \right\}.$$

Denote $\mathcal{C}_\mathcal{W}$ to be the $\varepsilon_1$-cover (in terms of $\ell_2$-norm) for $\mathcal{W}_{:,2}$*, and for any $v \in \mathbb{R}^{d_1}$, denote $\mathcal{C}_v := \mathcal{N}_\infty\left(\mathcal{F}_v, \varepsilon_2; x^{(1)}, \ldots, x^{(N)}; \|\cdot\|_2\right)$. We will set $\varepsilon_1$ and $\varepsilon_2$ later.

We will first show that $\mathcal{C} := \{f : v \in \mathcal{C}_\mathcal{W}, f \in \mathcal{F}_v\}$ is an $(\varepsilon_2 + B_X\varepsilon_1)$-cover (in terms of $\ell_\infty$) for $\mathcal{F}$. Consider $f \in \mathcal{F}$ parameterized by some $W = \begin{bmatrix} w_1 \\ \vdots \\ w_{d_1} \end{bmatrix}$. Since $W \in \mathcal{W}$, we know that there is a $\bar{v} \in \mathcal{C}_\mathcal{W}$, such that,

$$\sqrt{\sum_{i=1}^{d_1} (\|w_i\| - \bar{v}_i)^2} \leq \varepsilon_1.$$

*Here the cover is for a set and not a function.

Define $\bar{W} = \begin{bmatrix} \bar{v}_1 \cdot \frac{w_1}{\|w_1\|} \\ \vdots \\ \bar{v}_{d_1} \cdot \frac{w_{d_1}}{\|w_{d_1}\|} \end{bmatrix}$. Then there exists $f \in \mathcal{F}_{\bar{v}} \subseteq \mathcal{C}$ such that

$$\max_{i \in [N]} \|\bar{W}x^{(i)} - f(x^{(i)})\| \leq \varepsilon_2.$$

Now we have, for all $i \in [N]$,

$$\begin{aligned}
\|Wx^{(i)} - f(x^{(i)})\| &\leq \|\bar{W}x^{(i)} - f(x^{(i)})\| + \|(\bar{W} - W)x^{(i)}\| \\
&\leq \varepsilon_2 + \sqrt{\sum_{j=1}^{d_1} (\bar{v}_j - \|w_j\|)^2 \left(\frac{w_j \cdot x^{(j)}}{\|w_j\|^2}\right)^2} \\
&\leq \varepsilon_2 + B_X \sqrt{\sum_{j=1}^{d_1} (\bar{v}_j - \|w_j\|)^2)} \\
&\leq \varepsilon_2 + B_X \varepsilon_1.
\end{aligned}$$

Thus, we get the desired cover. Note that the size of the cover satisfies

$$\log |\mathcal{C}| \leq \log |\mathcal{C}_W| + \max_{v \in \mathcal{C}_W} \log |\mathcal{C}_v|.$$

Now we need to construct the sub-covers and bound the size of $\mathcal{C}$. Let us first construct the cover $\mathcal{C}_W$. Using Maurey's sparsification (see Theorem 3 in Zhang (2002)), we can find a proper cover of $\mathcal{C}_W$ which satisfies,

$$\log |\mathcal{C}_W| \leq \frac{B_W^2}{\varepsilon_1^2} \log(2d_1 + 1).$$

Let us now construct $\mathcal{C}_v$ for a fixed $v$. The approach will be to cover each of the rows of $W$ independently, treating each as specifying a linear function from $\mathbb{R}^{d_2} \to \mathbb{R}$. By Lemma A.10, letting

$\mathcal{Z}(b) : \{z : z \in \mathbb{R}^{d_2}, \|z\| \le b\}$ and $\mathcal{F}_{\text{linear}}(b) = \{x \mapsto z^\top x : z \in \mathcal{Z}(b)\}$, for any $\delta > 0$

$$\log \mathcal{N}_\infty(\mathcal{F}_{\text{linear}}(b); \delta; x^{(1)}, \cdots, x^{(N)}) \le \frac{cb^2 B_X^2 \log((1 + bB_X/\delta)N)}{\delta^2}.$$

In fact the cover, which we denote by $\bar{\mathcal{F}}_{\text{linear}}(b; \delta)$, is proper: $\bar{\mathcal{F}}_{\text{linear}}(b; \delta) = \{x \mapsto \bar{z}^\top x : \bar{z} \in \bar{Z}\}$ for some finite subset $\bar{Z} \subset \mathcal{Z}(b)$. Then the cover for the matrix can be constructed as,

$$\mathcal{C}_v = \left\{ x \mapsto \bar{Z}x : \bar{Z} = \begin{bmatrix} \bar{z}_1 \\ \vdots \\ \bar{z}_{d_1} \end{bmatrix} : \bar{z}_i \in \widehat{\mathcal{F}}_{\text{linear}}\left(v_i; \varepsilon_2 \sqrt{\frac{v_i}{\|v\|_1}}\right) \text{ for all } i \right\}.$$

Observe that this forms a cover for $\mathcal{F}_v$. For any $f \in \mathcal{F}_v$ parameterized by $W$, let $\bar{w}_i$ be the closest element in the corresponding row covers, then we have

$$\|Wx^{(i)} - \bar{W}x^{(i)}\| = \sqrt{\sum_{j=1}^{d_2} (w_i^\top x^{(i)} - \bar{w}_i^\top x^{(i)})^2} \le \sqrt{\sum_{j=1}^{d_2} \varepsilon_2^2 \cdot \frac{v_i}{\|v\|_1}} = \varepsilon_2.$$

□

Note that the size of $\mathcal{C}_v$ can be bounded as,

$$\begin{aligned}
\log |\mathcal{C}_v| &= \sum_{i=1}^{d_2} \log \mathcal{N}_\infty \left( \mathcal{F}_{\text{linear}}(v_i); \varepsilon_2 \sqrt{\frac{v_i}{\|v\|_1}}; x^{(1)}, \cdots, x^{(N)} \right) \\
&\le \sum_{i=1}^{d_2} \frac{c v_i^2 \|v\|_1 B_X^2 \log((1 + \|v\|_1 B_X/\varepsilon_2)N)}{\varepsilon_2^2 v_i} \\
&= \frac{c\|v\|_1^2 B_X^2 \log((1 + \|v\|_1 B_X/\varepsilon_2)N)}{\varepsilon_2^2} \le \frac{cB_W^2 B_X^2 \log((1 + B_W B_X/\varepsilon_2)N)}{\varepsilon_2^2}.
\end{aligned}$$

Combining the above and setting $\varepsilon_1, \varepsilon_2$ appropriately, we get

$$\log |\mathcal{C}| \leq \frac{B_W^2 B_X^2}{\varepsilon^2} \log(d_1 N).$$

### A.1.5 Capacity with positional embeddings

Since the Transformer architecture is permutation invariant for all $t \neq \tau$, positional embeddings (fixed or trainable) are typically added to the inputs to distinguish the different positions of the tokens. These positional embeddings are matrices $P \in \mathbb{R}^{T \times d}$ such that $P = [p_1 \dots p_T]^\top$ for $p_i \in \mathbb{R}^d$. Accounting for the positional embeddings as input, a single Transformer attention head can be expressed as:

$$f_{\text{tf-pos}}(X, P; W_V, W_{QK}) := \sigma\left(W_V^\top (X+P)^\top \mathsf{softmax}\left((X+P) W_{QK}^\top (x_\tau + p_\tau)\right)\right).$$

For a fixed positional embedding $P$, let us define

$$\mathcal{F}_{\text{tf-pos}}(P) := \{X \to f_{\text{tf-pos}}(X, P; W_V, W_{QK}) : \|W_V\|_{2,1} \leq B_V^{2,1}, \|W_V\| \leq B_V, \|W_{QK}^\top\|_{2,1} \leq B_{QK}^{2,1}\}$$

. Position embedding just impacts the input into the covering bound argument which effects the bound in terms of the $\left\|P^\top\right\|_{2,\infty}$ as given below,

**Lemma A.11.** *For all $X^{(1)}, \dots, X^{(m)} \in \mathbb{R}^{T \times d}$ such that $\left\|X^{(i)\top}\right\|_{2,\infty} \leq B_X$ for all $i \in [m]$, and $P \in \mathbb{R}^{T \times d}$ such that $\|P^\top\|_{2,\infty} \leq B_P$, the covering number of $\mathcal{F}_{\text{tf-pos}}(P)$ satisfies*

$$\log \mathcal{N}_\infty(\mathcal{F}_{\text{tf-pos}}(P); \varepsilon; X^{(1)}, \dots, X^{(m)}, \|\cdot\|_2) \lesssim (L_\sigma(B_X + B_P))^2 \cdot \frac{\left((B_V^{2,1})^{\frac{2}{3}} + (2B_{QK}^{2,1} B_V (B_X + B_P))^{\frac{2}{3}}\right)^3}{\varepsilon^2} \cdot \log(dmT).$$

*Proof.* Observe that $f_{\text{tf-pos}}(X, P; W_V, W_{QK}) = f_{\text{tf-head}}(X + P; W_V, W_{QK})$. Thus we have,

$$\log \mathcal{N}_\infty \left( \mathcal{F}_{\text{tf-pos}}(P); \varepsilon; \left\{ (X^{(i)}) \right\}_{i=1}^m, \|\cdot\|_2 \right) = \log \mathcal{N}_\infty \left( \mathcal{F}_{\text{tf-head}}; \varepsilon; \left\{ X^{(i)} + P \right\}_{i=1}^m, \|\cdot\|_2 \right).$$

For all $i \in [m]$, $\left\| (X^{(i)} + P)^\top \right\|_{2,\infty} \le \left\| X^{(i)\top} \right\|_{2,\infty} + \left\| P^\top \right\|_{2,\infty} \le B_X + B_P$. Therefore, using Corollary 4.9, we get the desired result. □

Therefore our bounds go through for fixed positional embeddings. If we were to train the embeddings, we would need a much finer cover on the embeddings which could incur a $T$ dependence.

### A.1.6 Capacity of multiple parallel heads

In virtually all practical applications of Transformers since their inception, instead of using one set of weights for an attention head, there are parallel attention heads, which have separate identically-shaped parameters; their outputs are concatenated. For the purposes of this analysis, suppose we have

$$f_{\text{tf-heads}} \left( X; \left\{ W_V^{[h]}, W_{QK}^{[h]} \right\}_{h=1}^H \right) := \sum_{h=1}^H f_{\text{tf-head}} \left( X; W_V^{[h]}, W_{QK}^{[h]} \right).$$

Let us define the class of multi-head self-attention with $H$ heads as

$$\mathcal{F}_{\text{tf-heads}} := \Bigg\{ X \mapsto f_{\text{tf-heads}} \left( X; \left\{ W_V^{[h]}, W_{QK}^{[h]} \right\}_{h=1}^H \right) : \\ \forall h \in [H], \left\| W_V^{[h]} \right\|_{2,1} \le B_V^{2,1\,[h]}, \left\| W_V^{[h]} \right\| \le B_V^{[h]}, \left\| W_{QK}^{[h]\top} \right\|_{2,1} \le B_{QK}^{2,1\,[h]} \Bigg\}.$$

**Lemma A.12.** *For all $X^{(1)}, \ldots, X^{(m)} \in \mathbb{R}^{T \times d}$ such that $\left\| X^{(i)\top} \right\|_{2,\infty} \le B_X$ for all $i \in [m]$, the covering number of $\mathcal{F}_{\text{tf-heads}}$ satisfies*

$$\log \mathcal{N}_\infty(\mathcal{F}_{\text{tf-heads}}; \varepsilon; X^{(1)}, \ldots, X^{(m)}, \|\cdot\|_2) \lesssim (L_\sigma B_X)^2 \cdot \frac{\left( \sum_{h=1}^H (B_V^{2,1\,[h]})^{\frac{2}{3}} + (2 B_{QK}^{2,1\,[h]} B_V^{[h]})^{\frac{2}{3}} \right)^3}{\varepsilon^2} \cdot \log(dmT).$$

*Proof.* For all $h \in [H]$, let $\mathcal{C}_h$ be an $\varepsilon_h$-covering of $\mathcal{F}_{\text{tf-head}}$ with weight bounds corresponding to head $h$. Since $f_{\text{tf-heads}}\left(X; \left\{W_V^{[h]}, W_{QK}^{[h]}\right\}_{h=1}^{H}\right) = \sum_{h=1}^{H} f_{\text{tf-head}}\left(X; W_V^{[h]}, W_{QK}^{[h]}\right)$, we have $\mathcal{C} := \mathcal{C}_1 \times \ldots \times \mathcal{C}_H$* is an $\left(\sum_{h=1}^{H} \varepsilon_h\right)$-covering for $\mathcal{F}_{\text{tf-heads}}$. Using Corollary 4.9 (and optimizing for $\varepsilon_h$ using Lemma A.8, by breaking them into individual errors for each head), we have

$$\log|\mathcal{C}| = \sum_{h=1}^{H} \log|\mathcal{C}_h| \leq \sum_{h=1}^{H} \leq (L_\sigma B_X)^2 \cdot \frac{\left(\sum_{h=1}^{H} (B_V^{2,1\,[h]})^{\frac{2}{3}} + (2B_{QK}^{2,1\,[h]} B_V^{[h]})^{\frac{2}{3}}\right)^3}{\varepsilon^2} \cdot \log(dmT).$$

□

To see the dependence on $H$, consider the setting where the weight bounds are the same for each head (dropping the $[h]$ subscript), then we get,

$$\log \mathcal{N}_\infty(\mathcal{F}_{\text{tf-heads}}; \varepsilon; X^{(1)}, \ldots, X^{(m)}, \|\cdot\|_2) \lesssim (L_\sigma B_X)^2 \cdot H^3 \cdot \frac{\left((B_V^{2,1})^{\frac{2}{3}} + (2B_{QK}^{2,1} B_V)^{\frac{2}{3}}\right)^3}{\varepsilon^2} \cdot \log(dmT).$$

### A.1.7 Capacity of multi-layer Transformers

This section analyzes the capacity of an $L$-layer Transformer. Let us denote the weights of layer $i$ by $W^{(i)} := \left\{W_Q^{(i)}, W_K^{(i)}, W_V^{(i)}, W_C^{(i)}\right\}$ such that $\left\|W_K^{(i)} W_Q^{(i)\top}\right\|_2 \leq B_{QK}^{(i)}$, $\left\|W_V^{(i)}\right\|_2 \leq B_V^{(i)}$, $\left\|W_C^{(i)}\right\|_2 \leq B_C^{(i)}$ and $\left\|W_K^{(i)\top} W_Q^{(i)}\right\|_{2,1} \leq B_{QK}^{2,1\,(i)}$, $\left\|W_V^{(i)}\right\|_{2,1} \leq B_V^{2,1(i)}$ and $\left\|W_C^{(i)}\right\|_{2,1} \leq B_C^{2,1(i)}$. Let us further denote the set of weights up to layer $i$ by $W^{1:i} = (W^{(1)}, \ldots, W^{i-1})$. Let the input representation of layer $i$ be $g_{\text{tf-head}}^{(i)}(X; W^{1:i})$. We inductively define $g$ with $g_{\text{tf-head}}^{(1)}(X; W^{1:1}) =$

*Here, $\times$ denotes the Cartesian product: the functions obtained by using the every combination of parameters of each individual cover.

$X$

$$g^{(i+1)}_{\text{tf-head}}\left(X; W^{1:i+1}\right) = \Pi_{\text{norm}}\left(\sigma\left(\Pi_{\text{norm}}\left(f\left(g^{(i)}_{\text{tf-head}}\left(X; W^{1:i}\right); W^{(i)}\right)\right)\right) W_C^{(i)}\right) \text{ with}$$
$$f\left(Z; \{W_Q, W_K, W_V, W_C\}\right) = \text{RowSoftmax}\left(ZW_Q\left(ZW_K\right)^\top\right) ZW_V,$$

where $\Pi_{\text{norm}}$ is applied row-wise. Our final output is $g_{\text{tf-scalar}}(X; W^{1:L+1}, w) = w^\top g^{(L)}_{\text{tf-head}}\left(X; W^{1:L+1}\right)$ `[CLS]` for $\|w\| \le B_w$.

In order to construct a cover, we will first bound the distance between the function $g$ with different weight parameters $W^{1:L+1}$ and $\widehat{W}^{1:L+1}$. This bound will depend on the closeness of the parameters which will allow us to construct a cover of the network in an iterative fashion by constructing covers of each layer.

### Lipschitzness of the network

To bound the Lipschitzness of the network, we will first bound the distance between $f$ with different weights and inputs.

**Lemma A.13** (Instantiation of Lemma 4.7)**.** *For any $W_K, \widehat{W}_K, W_V, \widehat{W}_V, W_Q, \widehat{W}_Q \in \mathbb{R}^{d\times k}$, for all $Z \in \mathbb{R}^{T\times d}$ such that $\left\|Z^\top\right\|_{2,\infty} \le 1$,*

$$\left\|\left(f\left(Z; \{W_Q, W_K, W_V, \cdot\}\right) - f\left(Z; \{\widehat{W}_Q, \widehat{W}_K, \widehat{W}_V, \cdot\}\right)\right)^\top\right\|_{2,\infty}$$
$$\le 2\left\|W_V\right\|_2 \left\|\left(W_Q W_K^\top - \widehat{W}_Q \widehat{W}_K^\top\right) Z^\top\right\|_{2,\infty} + \left\|\left(W_V - \widehat{W}_V\right)^\top Z^\top\right\|_{2,\infty}$$

*Proof.* Consider a fixed row $\tau$ of the output of the functions,

$$\begin{aligned}
&\left\| f\left(Z; \{W_Q, W_K, W_V, \cdot\}\right)[\tau] - f\left(Z; \{\widehat{W}_Q, \widehat{W}_K, \widehat{W}_V, \cdot\}\right)[\tau] \right\| \\
&= \left\| W_V^\top Z^\top \mathsf{softmax}\left(Z W_K W_Q^\top z_\tau\right) - \widehat{W}_V^\top Z^\top \mathsf{softmax}\left(Z \widehat{W}_K \widehat{W}_Q^\top z_\tau\right) \right\|
\end{aligned}$$

By triangle inequality:

$$\begin{aligned}
&\leq \left\| W_V^\top Z^\top \left(\mathsf{softmax}\left(Z W_K W_Q^\top z_\tau\right) - \mathsf{softmax}\left(Z \widehat{W}_K \widehat{W}_Q^\top z_\tau\right)\right) \right\| \\
&\quad + \left\| (W_V - \widehat{W}_V)^\top Z^\top \mathsf{softmax}\left(Z \widehat{W}_K \widehat{W}_Q^\top z_\tau\right) \right\|
\end{aligned}$$

Using $\|Pv\| \leq \|P\|_{2,\infty} \|v\|_1$:

$$\begin{aligned}
&\leq \left\| W_V^\top Z^\top \right\|_{2,\infty} \left\| \mathsf{softmax}\left(Z W_K W_Q^\top z_\tau\right) - \mathsf{softmax}\left(Z \widehat{W}_K \widehat{W}_Q^\top z_\tau\right) \right\|_1 \\
&\quad + \left\| (W_V - \widehat{W}_V)^\top Z^\top \right\|_{2,\infty} \left\| \mathsf{softmax}\left(Z \widehat{W}_K \widehat{W}_Q^\top z_\tau\right) \right\|_1
\end{aligned}$$

By Corollary A.7, $\left\|Z^\top\right\|_{2,\infty} \leq 1$, $\|PQ\|_{2,\infty} \leq \|P\|_2 \|Q\|_{2,\infty}$, and $\|P^\top\|_2 = \|P\|_2$:

$$\begin{aligned}
&\leq 2 \|W_V\|_2 \left\| Z W_K W_Q^\top z_\tau - Z \widehat{W}_K \widehat{W}_Q^\top z_\tau \right\|_\infty + \left\| (W_V - \widehat{W}_V)^\top Z^\top \right\|_{2,\infty} \\
&\leq 2 \|W_V\|_2 \left\| \left(W_Q W_K^\top - \widehat{W}_Q \widehat{W}_K^\top\right) Z^\top \right\|_{2,\infty} + \left\| (W_V - \widehat{W}_V)^\top Z^\top \right\|_{2,\infty}.
\end{aligned}$$

□

**Lemma A.14.** *For any $W_K, W_V, W_Q \in \mathbb{R}^{d \times k}$, for all $Z, \widehat{Z} \in \mathbb{R}^{T \times d}$ such that $\left\|Z^\top\right\|_{2,\infty} \leq 1$, $\|\widehat{Z}^\top\|_{2,\infty} \leq 1$,*

$$\begin{aligned}
&\left\| \left( f\left(Z; \{W_Q, W_K, W_V, \cdot\}\right) - f\left(\widehat{Z}; \{W_Q, W_K, W_V, \cdot\}\right) \right)^\top \right\|_{2,\infty} \\
&\leq \|W_V\|_2 \left(1 + 4 \left\| W_K W_Q^\top \right\|_2\right) \left\| (Z - \widehat{Z})^\top \right\|_{2,\infty}.
\end{aligned}$$

*Proof.* Consider a fixed row $\tau$ of the output of the functions,

$$\left\| f\left(Z; \{W_Q, W_K, W_V, \cdot\}\right)[\tau] - f\left(\widehat{Z}; \{W_Q, W_K, W_V, \cdot\}\right)[\tau] \right\|$$
$$= \left\| W_V^\top Z^\top \mathsf{softmax}\left(ZW_K W_Q^\top z_\tau\right) - W_V^\top \widehat{Z}^\top \mathsf{softmax}\left(\widehat{Z}W_K W_Q^\top \widehat{z}_\tau\right) \right\|$$

By triangle inequality:

$$\leq \left\| W_V^\top \left(Z - \widehat{Z}\right)^\top \mathsf{softmax}\left(ZW_K W_Q^\top z_\tau\right) \right\| + \left\| W_V^\top \widehat{Z}^\top \left(\mathsf{softmax}\left(ZW_K W_Q^\top z_\tau\right) - \mathsf{softmax}\left(\widehat{Z}W_K W_Q^\top \widehat{z}_\tau\right)\right) \right\|$$

Using $\|Pv\| \leq \|P\|_{2,\infty}\|v\|_1$:

$$\leq \left\| W_V^\top \left(Z - \widehat{Z}\right) \right\|_{2,\infty} \left\| \mathsf{softmax}\left(ZW_K W_Q^\top z_\tau\right) \right\|_1$$
$$+ \left\| W_V^\top \widehat{Z}^\top \right\|_{2,\infty} \left\| \mathsf{softmax}\left(ZW_K W_Q^\top z_\tau\right) - \mathsf{softmax}\left(\widehat{Z}W_K W_Q^\top \widehat{z}_\tau\right) \right\|_1$$

By Corollary A.7, $\left\| \widehat{Z}^\top \right\|_{2,\infty} \leq 1$ and $\|PQ\|_{2,\infty} \leq \|P\|_2 \|Q\|_{2,\infty}$:

$$\leq \|W_V\|_2 \left\| (Z - \widehat{Z})^\top \right\|_{2,\infty} + 2\|W_V\|_2 \left\| ZW_K W_Q^\top z_\tau - \widehat{Z}W_K W_Q^\top \widehat{z}_\tau \right\|_\infty$$

By triangle inequality:

$$\leq \|W_V\|_2 \left\| (Z - \widehat{Z})^\top \right\|_{2,\infty} + 2\|W_V\|_2 \left( \left\| (Z - \widehat{Z}) W_K W_Q^\top z_\tau \right\|_\infty + \left\| \widehat{Z}W_K W_Q^\top (z_\tau - \widehat{z}_\tau) \right\|_\infty \right)$$

Since $\left\| \widehat{Z}^\top \right\|_{2,\infty} \leq 1$ and $\|Pv\|_\infty \leq \|P^\top\|_{2,\infty}\|v\|$:

$$\leq \|W_V\|_2 \left(1 + 4\left\| W_K W_Q^\top \right\|_2\right) \left\| (Z - \widehat{Z})^\top \right\|_{2,\infty}.$$

□

With the above lemmas, we are ready to prove the effect of change of weights on $g$.

**Lemma A.15.** *For any $W_1^{i+1}$, $\widehat{W}_1^{i+1}$ satisfying the norm constraints,*

$$\begin{aligned}
&\left\|\left(g_{\text{tf-block}}^{(i+1)}(X; W^{1:i+1}) - g_{\text{tf-block}}^{(i+1)}(X; \widehat{W}^{1:i+1})\right)^\top\right\|_{2,\infty} \\
&\leq \left\|\left(W_C^{(i)} - \widehat{W}_C^{(i)}\right)^\top \sigma\left(\Pi_{\text{norm}}\left(f\left(\left(X; \widehat{W}^{1:i}\right); \widehat{W}^{(i)}\right)\right)\right)^\top\right\|_{2,\infty} \\
&\quad + L_\sigma B_C^{(i)} B_V^{(i)}\left(1 + 4B_{QK}^{(i)}\right)\left\|\left(g_{\text{tf-block}}^{(i)}\left(X; W^{1:i}\right) - g_{\text{tf-block}}^{(i)}\left(X; \widehat{W}^{1:i}\right)\right)^\top\right\|_{2,\infty} \\
&\quad + 2L_\sigma B_C^{(i)} B_V^{(i)}\left\|\left(W_Q^{(i)} W_K^{(i)\top} - \widehat{W}_Q^{(i)} \widehat{W}_K^{(i)\top}\right) g_{\text{tf-block}}^{(i)}\left(X; \widehat{W}^{1:i}\right)^\top\right\|_{2,\infty} \\
&\quad + L_\sigma B_C^{(i)}\left\|(W_V - \widehat{W}_V)^\top g_{\text{tf-block}}^{(i)}\left(X; \widehat{W}^{1:i}\right)^\top\right\|_{2,\infty}.
\end{aligned}$$

*Proof.* Unrolling one layer, we have

$$\begin{aligned}
&\left\|\left(g_{\text{tf-head}}^{(i+1)}\left(X; W^{1:i+1}\right) - g_{\text{tf-head}}^{(i+1)}\left(X; \widehat{W}^{1:i+1}\right)\right)^\top\right\|_{2,\infty} \\
&= \left\|\left(\Pi_{\text{norm}}\left(\sigma\left(\Pi_{\text{norm}}\left(f\left(g_{\text{tf-head}}^{(i)}\left(X; W^{1:i}\right); W^{(i)}\right)\right)\right) W_C^{(i)}\right)\right.\right. \\
&\qquad \left.\left. - \Pi_{\text{norm}}\left(\sigma\left(\Pi_{\text{norm}}\left(f\left(g_{\text{tf-head}}^{(i)}\left(X; \widehat{W}^{1:i}\right); \widehat{W}^{(i)}\right)\right)\right) \widehat{W}_C^{(i)}\right)\right)^\top\right\|_{2,\infty}
\end{aligned}$$

Using Lemma A.9 for each row:

$$\leq \left\|W_C^{(i)\top}\sigma\left(\Pi_{\text{norm}}\left(f\left(g_{\text{tf-head}}^{(i)}\left(X; W^{1:i}\right); W^{(i)}\right)\right)\right)^\top - \widehat{W}_C^{(i)\top}\sigma\left(\Pi_{\text{norm}}\left(f\left(g_{\text{tf-head}}^{(i)}\left(X; \widehat{W}^{1:i}\right); \widehat{W}^{(i)}\right)\right)\right)\right\|_{2,\infty}$$

By triangle inequality for each row:

$$\begin{aligned}
&\leq \underbrace{\left\|W_C^{(i)\top}\left(\sigma\left(\Pi_{\text{norm}}\left(f\left(g_{\text{tf-head}}^{(i)}\left(X; W^{1:i}\right); W^{(i)}\right)\right)\right) - \sigma\left(\Pi_{\text{norm}}\left(f\left(g_{\text{tf-head}}^{(i)}\left(X; \widehat{W}^{1:i}\right); \widehat{W}^{(i)}\right)\right)\right)\right)^\top\right\|_{2,\infty}}_{(A)} \\
&\quad + \left\|\left(W_C^{(i)} - \widehat{W}_C^{(i)}\right)^\top \sigma\left(\Pi_{\text{norm}}\left(f\left(g_{\text{tf-head}}^{(i)}\left(X; \widehat{W}^{1:i}\right); \widehat{W}^{(i)}\right)\right)\right)^\top\right\|_{2,\infty}.
\end{aligned}$$

Let us focus on term $(A)$.

Bounding the norm per row:

$$(A) \leq \left\| W_C^{(i)} \right\|_2 \left\| \sigma\left(\Pi_{\text{norm}}\left(f\left(g_{\text{tf-head}}^{(i)}\left(X; W^{1:i}\right); W^{(i)}\right)\right)\right)^\top - \sigma\left(\Pi_{\text{norm}}\left(f\left(g_{\text{tf-head}}^{(i)}\left(X; \widehat{W}^{1:i}\right); \widehat{W}^{(i)}\right)\right)\right)^\top \right\|_{2,\infty}$$

Since $\sigma$ is $L_\sigma$-Lipschitz and $\left\| W_C^{(i)} \right\|_2 \leq B_C^{(i)}$, for each row:

$$\leq L_\sigma B_C^{(i)} \left\| \Pi_{\text{norm}}\left(f\left(g_{\text{tf-head}}^{(i)}\left(X; W^{1:i}\right); W^{(i)}\right)\right)^\top - \Pi_{\text{norm}}\left(f\left(g\left(X; \widehat{W}^{1:i}\right); \widehat{W}^{(i)}\right)\right)^\top \right\|_{2,\infty}$$

Using Lemma A.9 for each row:

$$\leq L_\sigma B_C^{(i)} \left\| f\left(g_{\text{tf-head}}^{(i)}\left(X; W^{1:i}\right); W^{(i)}\right)^\top - f\left(g_{\text{tf-head}}^{(i)}\left(X; \widehat{W}^{1:i}\right); \widehat{W}^{(i)}\right)^\top \right\|_{2,\infty}$$

By triangle inequality:

$$\begin{aligned}
&\leq L_\sigma B_C^{(i)} \left\| f\left(g_{\text{tf-head}}^{(i)}\left(X; W^{1:i}\right); W^{(i)}\right)^\top - f\left(g_{\text{tf-head}}^{(i)}\left(X; \widehat{W}^{1:i}\right); W^{(i)}\right)^\top \right\|_{2,\infty} \\
&\quad + L_\sigma B_C^{(i)} \left\| f\left(g_{\text{tf-head}}^{(i)}\left(X; \widehat{W}^{1:i}\right); W^{(i)}\right)^\top - f\left(g_{\text{tf-head}}^{(i)}\left(X; \widehat{W}^{1:i}\right); \widehat{W}^{(i)}\right)^\top \right\|_{2,\infty}
\end{aligned}$$

By Lemma A.13 and A.14 and norm bounds on the matrices:

$$\begin{aligned}
&\leq L_\sigma B_C^{(i)} B_V^{(i)} \left(1 + 4B_{QK}^{(i)}\right) \left\| g_{\text{tf-head}}^{(i)}\left(X; W^{1:i}\right)^\top - g\left(X; \widehat{W}^{1:i}\right)^\top \right\|_{2,\infty} \\
&\quad + 2L_\sigma B_C^{(i)} B_V^{(i)} \left\| \left( W_Q^{(i)} W_K^{(i)\top} - \widehat{W}_Q^{(i)} \widehat{W}_K^{(i)\top} \right) g_{\text{tf-head}}^{(i)}\left(X; \widehat{W}^{1:i}\right)^\top \right\|_{2,\infty} \\
&\quad + L_\sigma B_C^{(i)} \left\| \left(W_V - \widehat{W}_V\right)^\top g_{\text{tf-head}}^{(i)}\left(X; \widehat{W}^{1:i}\right)^\top \right\|_{2,\infty}.
\end{aligned}$$

Combining the above gives us the desired result. □

Lastly, we take account of the last linear weight and observe that,

**Lemma A.16.** *For any $W^{1:L+1}$, $\widehat{W}^{1:L+1}$ and $w, \widehat{w}$,*

$$\begin{aligned}
&\left| g_{\text{tf-scalar}}\left(X; W^{1:L+1}, w\right) - g_{\text{tf-scalar}}\left(X; \widehat{W}^{1:L+1}, \widehat{w}\right) \right| \\
&\leq \|w\| \left\| g_{\text{tf-block}}^{(L+1)}\left(X; W^{1:L+1}\right)_{\texttt{[CLS]}} - g_{\text{tf-block}}^{(L+1)}\left(X; \widehat{W}^{1:L+1}\right)_{\texttt{[CLS]}} \right\| + \left| (w - \widehat{w})^\top g_{\text{tf-block}}^{(L+1)}\left(X; \widehat{W}^{1:L+1}\right)_{\texttt{[CLS]}} \right|.
\end{aligned}$$

*Proof.* Observe that,

$$\left|g_{\text{tf-scalar}}\left(X; W^{1:L+1}, w\right) - g_{\text{tf-scalar}}\left(X; \widehat{W}^{1:L+1}, \widehat{w}\right)\right|$$
$$= \left|w^\top g^{(L+1)}_{\text{tf-block}}\left(X; W^{1:L+1}\right)_{\texttt{[CLS]}} - \widehat{w}^\top g^{(L+1)}_{\text{tf-block}}\left(X; \widehat{W}^{1:L+1}\right)_{\texttt{[CLS]}}\right|$$

By triangle inequality:

$$\leq \left|w^\top \left(g^{(L+1)}_{\text{tf-block}}\left(X; W^{1:L+1}\right)_{\texttt{[CLS]}} - g^{(L+1)}_{\text{tf-block}}\left(X; \widehat{W}^{1:L+1}\right)_{\texttt{[CLS]}}\right)\right| + \left|(w-\widehat{w})^\top g^{(L+1)}_{\text{tf-block}}\left(X; \widehat{W}^{1:L+1}\right)_{\texttt{[CLS]}}\right|$$

Bounding the inner product by norms:

$$\leq \|w\| \left\|g^{(L+1)}_{\text{tf-block}}\left(X; W^{1:L+1}\right)_{\texttt{[CLS]}} - g^{(L+1)}_{\text{tf-block}}\left(X; \widehat{W}^{1:L+1}\right)_{\texttt{[CLS]}}\right\| + \left|(w-\widehat{w})^\top g^{(L+1)}_{\text{tf-block}}\left(X; \widehat{W}^{1:L+1}\right)_{\texttt{[CLS]}}\right|.$$

□

### Constructing the cover

The cover construction follows the standard recipe of composing covers per layer (as in Bartlett et al. (2017)).

**Theorem A.17.** *Let $\mathcal{F}^{(L)}_{\text{tf-scalar}}$ represent the class of functions of L-layer Transformer blocks satisfying the norm bounds (specified before) followed by linear layer on the* `[CLS]` *token. Then, for all $X^{(i)}$*

$$\log \mathcal{N}_\infty(\mathcal{F}^{(L)}_{\text{tf-scalar}}; \varepsilon; X^{(1)}, \ldots, X^{(m)}, \|\cdot\|_2) \lesssim$$
$$\frac{\log(dmT)}{\varepsilon^2} \times \left(B_w^{\frac{2}{3}} + \sum_{i=1}^{L} \alpha_i^{\frac{2}{3}} \left(B_C^{2,1(i)\frac{2}{3}} + d^{\frac{2}{3}}\left(2L_\sigma B_C^{(i)} B_V^{(i)} B_{QK}^{2,1\,(i)}\right)^{\frac{2}{3}} + k^{\frac{2}{3}}\left(L_\sigma B_C^{(i)} B_V^{2,1(i)}\right)^{\frac{2}{3}}\right)\right)^3$$

*where* $\alpha_i = \prod_{j<i} L_\sigma B_C^{(j)} B_V^{(j)}(1 + 4B_{QK}^{(j)})$.

*Proof.* Our goal is to show that for every $\varepsilon > 0$, and collection of inputs $X^{(1)}, \ldots, X^{(m)}$, there is a cover $\mathcal{C}$ of vectors in $\mathbb{R}^{(m)}$ such that for all $W^{1:L+1}$ and $w$ satisfying the norm bounds, there is some $v \in \mathcal{C}$ such that $\max_i |g_{\text{tf-scalar}}(X^{(i)}; W^{1:L+1}, w) - v| \leq \varepsilon$.

In each layer of the transformer, $W_Q^{(i)}$ and $W_K^{(i)}$ always appear together in the form $W_K^{(i)} W_Q^{(i)\top}$. Therefore, we will overload notation and define $W_{QK}^{(i)} : W_K^{(i)} W_Q^{(i)\top}$. Our cover $\mathcal{C}$ will be proper, consisting of vectors of the form $(g_{\text{tf-scalar}}(X^{(i)}; \widehat{W}^{1:L+1}, \widehat{w}))_{i\in[m]}$. We will build the cover iteratively by finding finite collections of matrices $\widehat{\mathcal{W}}^{1:i}$ for each layer.

First observe that for any collection of $Z^{(1)}, \dots, Z^{(m)} \in \mathbb{R}^{T\times d_1}$, and any $W, \widehat{W} \in \mathbb{R}^{d_1\times d_2}$,

$$\max_{i\in[m]} \left\| W^\top Z^{(i)\top} - \widehat{W}^\top Z^{(i)\top} \right\|_{2,\infty} = \max_{i\in[m],t\in[T]} \left\| W^\top z_t^{(i)} - \widehat{W}^\top z_t^{(i)} \right\|.$$

This crucially allows us to aggregate over the samples and context length. In particular, we can apply Lemma 4.10 with the input vectors $(z_t^{(i)})_{i\in[m],t\in[T]}$; a total of $mT$ input vectors. Specifically, for any $\varepsilon$ and $\mathcal{W}(d_1, d_2, \alpha) := \{W \in \mathbb{R}^{d_1\times d_2} \mid \|W\|_{2,1} \le \alpha\}$ with fixed $Z^{(i)}$ satisfying $\left\|Z^{(i)\top}\right\|_{2,\infty} \le 1$, Lemma 4.10 gives us such a cover.

First let us build a cover for one Transformer layer with inputs $Z^{(1)}, \dots, Z^{(m)}$. We will begin with creating an $\varepsilon_V$-cover $\widehat{\mathcal{W}}_V$ for the function class of linear transformations given by $\mathcal{W}_V : \{W \in \mathbb{R}^{d\times k}, \|W\|_{2,1} \le \alpha, \|W\|_2 \le s\}$ and $\varepsilon_{QK}$-cover $\widehat{\mathcal{W}}_{QK}$ for $\mathcal{W}_{QK} := \{W \in \mathbb{R}^{d\times d}, \left\|W^\top\right\|_{2,1} \le \beta, \|W\|_2 \le r\}$ and inputs $Z^{(1)}, \dots, Z^{(m)}$. For each pair of $\widehat{W}_V \in \widehat{\mathcal{W}}_V$ and $\widehat{W}_{QK} \in \widehat{\mathcal{W}}_{QK}$, we construct an $\varepsilon_C$-cover $\widehat{\mathcal{W}}_C(\widehat{W}_V, \widehat{W}_{QK})$ for $\mathcal{W}_C : \{W \in \mathbb{R}^{k\times d}, \|W\|_{2,1} \le \gamma, \|W\|_2 \le c\}$ and inputs $\left\{\sigma\left(\Pi_{\text{norm}}\left(f\left(Z^{(i)}; \widehat{W}_V, \widehat{W}_{QK}\right)\right)\right)\right\}_{i=1}^m$. Our final cover is

$$\widehat{\mathcal{W}} := \left\{(\widehat{W}_V, \widehat{W}_{QK}, \widehat{W}_C) : \widehat{W}_V \in \widehat{\mathcal{W}}_V, \widehat{W}_V \in \widehat{\mathcal{W}}_V, \widehat{W}_C \in \widehat{\mathcal{W}}_C(\widehat{W}_V, \widehat{W}_{QK})\right\}.$$

Using Lemma A.15, we can show that $\widehat{\mathcal{W}}$ is an $\varepsilon$-cover for $g(\cdot; \{W_V, W_{QK}, W_C\})$ and inputs $Z^{(1)}, \dots, Z^{(m)}$ where

$$\varepsilon = \varepsilon_C + 2L_\sigma cs\varepsilon_{QK} + L_\sigma c\varepsilon_V.$$

Using Lemma 4.10, the size of the cover is

$$|\widehat{\mathcal{W}}| \le |\widehat{\mathcal{W}}_V||\widehat{\mathcal{W}}_{QK}| \max_{\substack{\widehat{W}_V \in \widehat{\mathcal{W}}_V \\ \widehat{W}_{QK} \in \widehat{\mathcal{W}}_{QK}}} \left|\widehat{\mathcal{W}}_C(\widehat{W}_V, \widehat{W}_{QK})\right|$$

$$\implies \log|\widehat{\mathcal{W}}| \lesssim \left(\frac{\alpha^2}{\varepsilon_V^2} + \frac{\beta^2}{\varepsilon_{QK}^2} + \frac{\gamma^2}{\varepsilon_C^2}\right)\log(dmT).$$

We are now ready to inductively construct a cover for the deeper network. Suppose we have a $\varepsilon^{(i)}$-cover $\widehat{\mathcal{W}}^{1:i}$ for $g(\cdot; W^{1:i})$ on $X^{(1)}, \cdots, X^{(m)}$. We show how to construct an $\varepsilon^{(i+1)}$-cover for $g(\cdot; W^{1:i+1})$. For every element $\widehat{W}^{1:i} \in \widehat{\mathcal{W}}^{1:i}$ we construct a $\left(\varepsilon_C^{(i)} + 2L_\sigma B_C^{(i)} B_V^{(i)} \varepsilon_{QK}^{(i)} + L_\sigma B_C^{(i)} \varepsilon_V^{(i)}\right)$-cover $\widehat{\mathcal{W}}_i(\widehat{W}^{1:i})$ for the transformer layer (as above) on inputs $\left\{g(X^{(j)}; \widehat{W}^{1:i})\right\}_{j=1}^m$. Consider the cover

$$\widehat{\mathcal{W}}^{1:i+1} := \left\{(\widehat{W}^{1:i}, \widehat{W}^{(i)}) : \widehat{W}^{1:i} \in \widehat{\mathcal{W}}^{1:i}, \widehat{W}^{(i)} \in \widehat{\mathcal{W}}_i(\widehat{W}^{1:i})\right\}.$$

By Lemma A.15, this gives,

$$\varepsilon^{(i+1)} = L_\sigma B_C^{(i)} B_V^{(i)} (1 + 4B_{QK}^{(i)})\varepsilon^{(i)} + \varepsilon_C^{(i)} + 2L_\sigma B_C^{(i)} B_V^{(i)} \varepsilon_{QK}^{(i)} + L_\sigma B_C^{(i)} \varepsilon_V^{(i)}.$$

The size of the cover is

$$|\widehat{\mathcal{W}}^{1:i+1}| \le |\widehat{\mathcal{W}}^{1:i}| \max_{\widehat{W}^{1:i} \in \widehat{\mathcal{W}}^{1:i}} \left|\widehat{\mathcal{W}}_i(\widehat{W}^{1:i})\right|.$$

Inductively applying this, we get

$$\begin{aligned}
\varepsilon^{(L+1)} &= \sum_{i=1}^{L} \left(\prod_{j<i} L_\sigma B_C^{(j)} B_V^{(j)} (1 + 4B_{QK}^{(j)})\right) \left(\varepsilon_C^{(i)} + 2L_\sigma B_C^{(i)} B_V^{(i)} \varepsilon_{QK}^{(i)} + L_\sigma B_C^{(i)} \varepsilon_V^{(i)}\right) \\
&= \sum_{i=1}^{L} \alpha_i \left(\varepsilon_C^{(i)} + 2L_\sigma B_C^{(i)} B_V^{(i)} \varepsilon_{QK}^{(i)} + L_\sigma B_C^{(i)} \varepsilon_V^{(i)}\right)
\end{aligned}$$

where $\alpha_i = \prod_{j<i} L_\sigma B_C^{(j)} B_V^{(j)} (1 + 4B_{QK}^{(j)})$.

The size of the cover is

$$\log\left(|\widehat{\mathcal{W}}^{1:L+1}|\right) \le \sum_{i=1}^{L} \left( \frac{B_V^{2,1(i)^2}}{\varepsilon_V^{(i)^2}} + \frac{B_{QK}^{2,1\,(i)^2}}{\varepsilon_{QK}^{(i)\,2}} + \frac{B_C^{2,1(i)^2}}{\varepsilon_C^{(i)^2}} \right) \log(dmT).$$

Notice that the layer-norm maintains the norm bound on the inputs. Lastly, we need to cover the linear layer on the `[CLS]` token and compose it with the cover of $g^{1:L}$ (as before). Using Lemma A.10 and A.16, we can get the final $\varepsilon$-cover $\mathcal{C}$ with

$$\varepsilon = B_w \sum_{i=1}^{L} \alpha_i \left( \varepsilon_C^{(i)} + 2L_\sigma B_C^{(i)} B_V^{(i)} \varepsilon_{QK}^{(i)} + L_\sigma B_C^{(i)} \varepsilon_V^{(i)} \right) + \varepsilon_w$$

and size

$$\log|\mathcal{C}| \lesssim \frac{B_w^2 \log(m)}{\varepsilon_w^2} + \sum_{i=1}^{L} \left( \frac{B_V^{2,1(i)^2}}{\varepsilon_V^{(i)^2}} + \frac{B_{QK}^{2,1\,(i)^2}}{\varepsilon_{QK}^{(i)\,2}} + \frac{B_C^{2,1(i)^2}}{\varepsilon_C^{(i)^2}} \right) \log(dmT).$$

Using Lemma A.8, the size of the cover for fixed $\varepsilon$ gives us the desired result. □

## A.2 Sparse function representation via bounded-norm Transformers

### A.2.1 Setup

Reductions from Boolean functions to Transformers. In order to establish our function approximation results, we must first define a canonical mapping between length-$T$ Boolean strings $b \in \{0,1\}^T$ and Transformer inputs $X \in \mathbb{R}^{T\times d}$. The key point (which has also been considered since the inception of the Transformer (Vaswani et al., 2017), and continues to be a crucial consideration in practice (Dosovitskiy et al., 2020)) is that the network's permutation-equivariant

symmetry needs to be broken by assigning different embeddings to different indices of $b$. There are several possible natural choices here, which are all of practical interest:

- *Deterministic positional embeddings.* Fix positional embedding matrices $P \in \mathbb{R}^{T\times d}, E \in \mathbb{R}^{\{0,1\}\times d}$, and a special direction $v_{\texttt{[CLS]}} \in \mathbb{R}^d$, such that the $T+3$ vectors $\{P_{t,:}\}_{t=1}^{T} \cup \{E_{j,:}\}_{j\in 0,1} \cup \{v_{\texttt{[CLS]}}\}$ are an approximately orthonormal basis for $\mathbb{R}^d$ (see below). The input to the Transformer is then $X = E_b + P$, where $E_b \in \mathbb{R}^{T\times d}$ such that $[E_b]_{t,:} = E_{b_t,:}$ for each $t \in [T]$. In the $f_{\textsf{tf-scalar}}$ formulation, we choose the auxiliary input $x_{\texttt{[CLS]}}$ to be the constant vector $v_{\texttt{[CLS]}}$. This closely matches applications of Transformers in NLP (Vaswani et al., 2017).

- *Trainable positional embeddings.* Like the above, but $P$ is a trainable parameter; we still require approximate orthogonality of $\{E_{j,:}\}_{j\in 0,1} \cup \{v_{\texttt{[CLS]}}\}$. It is also possible to consider the case where $E$ and $v_{\texttt{[CLS]}}$ are trainable (matching the way token embeddings are trained in practice). This becomes important in the regime of large vocabulary sizes that require embeddings to capture shared information between tokens; however, this is not necessary for our constructions, as we limit our consideration to binary tokens. This simplifies our constructions and improves statistical rates; additionally, it is a popular and well-studied alternative (Vaswani et al., 2017; Devlin et al., 2018; Radford et al., 2018, 2019; Brown et al., 2020).

- *Bag of vectors.* Fix a matrix $V \in \mathbb{R}^{T\times d}$ with approximately orthogonal rows (like the deterministic $P$), but choose the Transformer input

$$X := V\mathrm{diag}(b).$$

  This construction replaces positional embeddings with positional "indicator vectors" which

can be swapped between any of the Transformer's input positions. It has the advantage of being symmetric with respect to permutation of the Transformer's input positions: it turns out that

$$f_{\text{tf-scalar}}(V\text{diag}(b)) = f_{\text{tf-scalar}}(V\Pi\text{diag}(b)),$$

for any $T \times T$ permutation matrix $\Pi$. It is also the most natural construction when considering the composition of sparse Boolean functions across multiple layers: a layer can output combinations of the basis rows $v_i$ for further function composition, like Boolean gates.

Approximately orthonormal basis. Each of the Boolean function approximation constructions will rely on a basis set of vectors, which will be used as positional embeddings (or the variable indices in the bag-of-vectors construction). We will fix a set of approximately orthonormal vectors $\{v_i : \|v_i\| = 1\}_{i=1}^{T'}$ in $\mathbb{R}^d$: for each $i \neq j$, we have $|v_i^\top v_j| \leq \Delta$. When $\Delta = 0$, the maximal $T'$ for which such a set exists is $d$; for $\Delta \in (0, \frac{1}{2})$, the Johnson-Lindenstrauss lemma (Johnson et al., 1986) implies that the maximal set of is of size $\exp(\Theta(d\Delta^2))$. For given choices of $d$, $\Delta$ and a maximal $\{v_1, \ldots, v_{T'}\}$, our construction is valid for contexts of length $T \leq T'$. For the special vectors $e_0, e_1, v_{\texttt{[CLS]}}$, we will assume that these are exactly orthogonal to the $v_i$ and each other, so that the $v_i$ must be a basis in dimension at least $d - 3$. This is for clarity only– it reduces the number of error terms to propagate through the analysis.

Self-attention block. In each construction (which specifies an input $X \in \mathbb{R}^{T \times d}$, we will specify the parameters $W_Q, W_K, W_V, W_C, w = e_1$ of a scalar-output Transformer $f_{\text{tf-scalar}}$, which takes an input $X \in \mathbb{R}^{(T+1) \times d}$; the auxiliary token input will be the constant vector $x_{\texttt{[CLS]}} := v_{\texttt{[CLS]}} \in \mathbb{R}^d$. The internal activation function $\sigma$ is chosen to be the identity. Summarizing, the

functional form of $f_{\text{tf-scalar}} \in \mathcal{F}_{\text{tf-scalar}}$ in these constructions is

$$f_{\text{tf-scalar}}(X; W_Q, W_K, W_V, W_C, e_1) = \mathsf{softmax}\left(v_{[\text{CLS}]}^\top W_Q W_K^\top X^\top\right) X W_V W_C e_1.$$

In the intermediate lemmas, it will also be useful to consider the corresponding attention head output

$$f_{\text{tf-head}}(X; W_Q, W_K, W_V, W_C) = \mathsf{softmax}\left(v_{[\text{CLS}]}^\top W_Q W_K^\top X^\top\right) X W_V W_C,$$

and its projections $f_{\text{tf-head}} \circ \Pi_{d_{\text{proj}}}$ onto the first $d_{\text{proj}}$ coordinates.

Feedforward networks. We establish some notation for feedforward networks. An $L$-layer feedforward network, with activation function $\sigma : \mathbb{R} \to \mathbb{R}$ and dimensions $d_1, \ldots, d_{L+1}$, is parameterized by weight matrices $W_i \in \mathbb{R}^{d_{i+1} \times d_i}$, and maps $x \in \mathbb{R}^{d_1}$ to $y \in \mathbb{R}^{d_{L+1}}$, by the iterative equations

$$y_1^\top := \sigma(x^\top W_1),$$

$$y_{i+1}^\top := \sigma(y_i^\top W_i), \qquad i = 1, \ldots, L-1,$$

$$f_{\mathsf{mlp}}(x; W_1, \ldots, W_L)^\top = y^\top := y_L^\top W_i.$$

When $d_{L+1} = 1$, we will use the notation $w$ instead of $W_L$. It will be convenient to incorporate *bias* weights by introducing an extra input coordinate $W_i \in \mathbb{R}^{(d_{i+1}+1) \times d_i}$, and augmenting the linear function accordingly:

$$y_i^\top W_i \mapsto [y_i^\top \; 1] W_i.$$

Self-attention composed with a feedforward network. The full definition of the Transformer layer composes a self-attention layer ($f_{\text{tf-layer}} : \mathbb{R}^{T \times d} \to \mathbb{R}^{T \times d}$) with a position-wise feedforward network ($f_{\mathsf{mlp}} : \mathbb{R}^d \to \mathbb{R}^d$). We will use this combination of modules to establish our

function approximation results: $f_{\text{tf-layer}}$ acts as a sparse bottleneck, while $f_{\text{mlp}}$ approximates an arbitrary function of the selected coordinates. For our single-layer constructions, it is most convenient to establish notation for a scalar-output Transformer with a feedforward network. To this end, define $\mathcal{F}_{\text{tf+mlp}}$ to be the function class with the same $\mathsf{Score}$, $\mathsf{Norm}$, $\varphi_{\text{in}}$ functions as in $\mathcal{F}_{\text{tf-scalar}}$ (thus, the same parameters $W_Q$, $W_K$, $W_V$), with identity activation function, but a feedforward neural network replacing the linear $\varphi_{\text{out}}$ and $w$. Concretely, with $L = 3$ and the ReLU activation function $(\cdot)_+$, $\mathcal{F}_{\text{tf+mlp}}$ contains functions of the form

$$f_{\text{tf+mlp}}(X;\theta) = \left((y^\top W_1)_+ W_2\right)_+ w,$$

$$y = \mathsf{softmax}\left(v_{[\mathsf{CLS}]}^\top W_Q W_K^\top X^\top\right) X W_V W_C w,$$

with parameters $\theta := (W_Q, W_K, W_V, W_C, W_1, W_2, w)$.

Multiple self-attention heads. The final component we will need for the function approximation setup is multi-headed self-attention. We will extend the definition of the single-headed $f_{\text{tf-head}}$ to

$$f_{\text{tf-heads}}\left(X; \left\{W_Q^{[b]}, W_K^{[b]}, W_V^{[b]}, W_C^{[b]}\right\}_{b=1}^{H}\right) := \sum_{b=1}^{H} f_{\text{tf-head}}\left(X; W_Q^{[b]}, W_K^{[b]}, W_V^{[b]}, W_C^{[b]}\right),$$

and substitute this definition into $f_{\text{tf+mlp}}$ when discussing a multi-headed construction.

Classes and properties of Boolean functions. We will call a Boolean function $f : \{0,1\}^T \to \mathcal{Y}$ *$\mathcal{I}$-sparse* if it only depends on a fixed subset $\mathcal{I} \subseteq [T]$ of its inputs:

$$b_i = b'_i \quad \forall i \in \mathcal{I} \implies f(b) = f(b').$$

Overloading notation, if $\mathcal{I} = s$, we will also call $f$ $s$-sparse. We will call an $\mathcal{I}$-sparse Boolean function $f$ *symmetric* if its value is invariant under permutation of the indices in $\mathcal{I}$:

$$|\{i \in \mathcal{I} : b_i = 1\}| = |\{i \in \mathcal{I} : b'_i = 1\}| \implies f(b) = f(b').$$

Further, we will call an $\mathcal{I}$-sparse real-valued symmetric Boolean function $f : \{0,1\}^T \to \mathcal{Y}$ *monotone* if $f(b)$ is monotonically increasing in $r := |\{i \in \mathcal{I} : b_i = 1\}|$. If, for some $\gamma > 0$, it holds that $f(r+1) \geq f(r) + \gamma$ for each $r = 0, \ldots, s-1$, we call $f$ $\gamma$-*strictly monotone*. A vector-valued $\mathcal{I}$-sparse Boolean function $f : \{0,1\}^T \to \mathbb{R}^{d_f}$ is $\gamma$-*injective* if

$$\left\| f(b) - f(b') \right\|_\infty \geq \gamma$$

for each $b, b'$ that differ at some position $i \in \mathcal{I}$; $f$ is called $B$-bounded if $\|f(b)\|_\infty \leq B$ for all $b \in \{0,1\}^T$.

Uniform approximation. For some $\varepsilon \geq 0$ and a function $f : \{0,1\}^T \to \mathbb{R}^d$, we say that $\widehat{f} \in \mathcal{F}$ $\varepsilon$-uniformly approximates $f$ under the mapping $b \mapsto X(b)$ if

$$\left\| \widehat{f}(X(b)) - f(b) \right\|_\infty \leq \varepsilon, \qquad \forall b \in \{0,1\}^T.$$

### A.2.2 Results

We give an overview of the function approximation results under each input mapping $b \mapsto X(b)$, as a multi-part proposition:

**Proposition A.18** (Sparse variable creation with Transformers)**.** *The function classes* $\mathcal{F}_{\mathsf{tf\text{-}scalar}}, \mathcal{F}_{\mathsf{tf+mlp}}$ *contain the following classes of sparse Boolean functions:*

- *Deterministic positional embeddings:* For any $\mathcal{I}$, $\mathcal{F}_{\mathsf{tf\text{-}scalar}}$ can approximate a particular monotone symmetric $\mathcal{I}$-sparse $f$, with Transformer weight norm bounds from the real-valued construction in Lemma A.19. $\mathcal{F}_{\mathsf{tf+mlp}}$ with 1 head can exactly represent any symmetric $s$-sparse $f$, with the same bounds on Transformer weight norms, and feedforward network weight norms scaling as $O(\text{poly}(s))$. $\mathcal{F}_{\mathsf{tf+mlp}}$ with $s$ heads can exactly represent any $s$-sparse $f$, with Transformer weight norm bounds from the vector-valued construction in Lemma A.19, and feedforward network weight norms scaling as $O(\text{poly}(s) \cdot 2^s)$.

- *Trainable positional embeddings:* For any $\mathcal{I}$, $\mathcal{F}_{\mathsf{tf\text{-}scalar}}$ can approximate a particular monotone symmetric $\mathcal{I}$-sparse $f$, with positional embedding and Transformer weight norm bounds from the real-valued construction in Lemma A.20. $\mathcal{F}_{\mathsf{tf+mlp}}$ with 1 head can exactly represent any symmetric $s$-sparse $f$, with the same bounds on $P$ and Transformer weight norms, and feedforward network weight norms scaling as $O(\text{poly}(s))$. $\mathcal{F}_{\mathsf{tf+mlp}}$ with $s$ heads can exactly represent any sparse $f$, with $P$ and Transformer weight norm bounds from the vector-valued construction in Lemma A.20, and feedforward network weight norms scaling as $O(\text{poly}(s) \cdot 2^s)$.

- *Bag of vectors:* For any $\mathcal{I}$, $\mathcal{F}_{\mathsf{tf\text{-}scalar}}$ can approximate a particular monotone symmetric $\mathcal{I}$-sparse $f$, with Transformer weight norms from Lemma A.21. $\mathcal{F}_{\mathsf{tf+mlp}}$ with 1 head can represent any symmetric $s$-sparse $f$, with the same Transformer weight norm bounds, and feedforward network weight norms scaling as $O(\text{poly}(s))$. $\mathcal{F}_{\mathsf{tf+mlp}}$ with 1 head can also exactly represent any $s$-sparse $f$, with the same bounds on Transformer weight norms, and feedforward network weight norms scaling as $O(\text{poly}(s) \cdot 2^s)$.

The formal statements are obtained by $(\gamma/4)$-uniformly approximating a $\gamma$-strictly monotone or $\gamma$-injective function with self-attention alone (Lemmas A.19, A.20, A.21), then applying a robust universal function representation construction (Lemmas A.22, A.23) appropriately. They are

organized as follows:

**Lemma A.19** (Deterministic $P$, no MLP)**.** *Suppose $X(b) = P + E_b$ with deterministic $P$. Let $\mathcal{I} \subseteq [T]$ such that $|\mathcal{I}| = s \leq d, k$, and $\Delta < 1/s$. Then, for all $0 < \gamma \leq 1$, there exists a 1-bounded, $(2/s)$-strictly monotone symmetric $\mathcal{I}$-sparse Boolean function $g_{\mathcal{I}} : \{0,1\}^T \to \mathbb{R}$ and Transformer head parameters such that $f_{\text{tf-scalar}}(X(b); W_Q, W_K, W_V, W_C, w = e_1)$ $(\gamma/4)$-uniformly approximates $g_{\mathcal{I}}$. The norms satisfy*

$$\left\|W_Q\right\|_F \leq \frac{\log\left(\frac{8T}{\gamma}\right)}{1 - s\Delta}, \qquad \|W_K\|_F \leq s, \qquad \|W_V\|_F \leq 2, \qquad \|W_C\|_F \leq 1.$$

*Also, there exists a 1-bounded, 2-injective $\mathcal{I}$-sparse Boolean function $g'_{\mathcal{I}} : \{0,1\}^T \to \mathbb{R}^s$ and $s$-headed Transformer parameters such that $f_{\text{tf-head}}\left(X(b); \left\{W_Q^{[b]}, W_K^{[b]}, W_V^{[b]}, W_C^{[b]}\right\}_{b=1}^{s}\right) \circ \Pi_s$ uniformly approximates $g'_{\mathcal{I}}$. The norms of each head satisfy*

$$\left\|W_Q^{[b]}\right\|_F \leq \frac{\log\left(\frac{8T}{\gamma}\right)}{1 - s\Delta}, \qquad \left\|W_K^{[b]}\right\|_F \leq 1, \qquad \left\|W_V^{[b]}\right\|_F \leq 2, \qquad \left\|W_C^{[b]}\right\|_F \leq 1.$$

**Lemma A.20** (Trainable $P$, no MLP)**.** *Suppose $X(b) = P + E_b$ with trainable $P$. Let $\mathcal{I} \subseteq [T]$ such that $|\mathcal{I}| = s \leq d, k$. Then, for any $0 < \gamma \leq 1$, and with the same $g_{\mathcal{I}}$ as in Lemma A.19, there exists $P$ and Transformer head parameters such that $f_{\text{tf-scalar}}(X(b); W_Q, W_K, W_V, W_C, w = e_1)$ $(\gamma/4)$-uniformly approximates $g_{\mathcal{I}}$. The norms satisfy*

$$\left\|P^\top\right\|_{2,1} \leq s, \qquad \left\|W_Q\right\|_F \leq \log\left(\frac{8T}{\gamma}\right), \qquad \|W_K\|_F \leq 1, \qquad \|W_V\|_F \leq 2, \qquad \|W_C\|_F \leq 1.$$

*Also, for the same $g'_{\mathcal{I}}$ as in Lemma A.19, there exists $P$ and $s$-headed Transformer parameters such that $f_{\text{tf-head}}\left(X(b); \left\{W_Q^{[b]}, W_K^{[b]}, W_V^{[b]}, W_C^{[b]}\right\}_{b=1}^{s}\right) \circ \Pi_s$ uniformly approximates $g'_{\mathcal{I}}$. The norms of each*

*head satisfy*

$$\left\|P^\top\right\|_{2,1} \le s, \qquad \left\|W_Q^{[b]}\right\|_F \le \log\left(\frac{8T}{\gamma}\right), \qquad \left\|W_K^{[b]}\right\|_F \le 1, \qquad \left\|W_V^{[b]}\right\|_F \le 2, \qquad \left\|W_C^{[b]}\right\|_F \le 1.$$

**Lemma A.21** (Bag of vectors, no MLP). *Suppose $X(b) = V + \mathrm{diag}(b)$. Let $\mathcal{I} \subseteq [T]$ such that $|\mathcal{I}| = s \le d, k$, and $\Delta < 1/s$. Then, for all $s\Delta < \gamma < 1$, there exists an $s$-bounded, $(1/s)$-strictly monotone symmetric $\mathcal{I}$-sparse Boolean function $g_{\mathcal{I}} : \{0,1\}^T \to \mathbb{R}$ and Transformer head parameters such that $f_{\text{tf-scalar}}(X(b); W_Q, W_K, W_V, W_C, w = e_1)$ $(\gamma/4)$-uniformly approximates $g_{\mathcal{I}}$. The norms satisfy*

$$\left\|W_Q\right\|_F \le \frac{\log\left(\frac{8Ts(1+\Delta)}{\gamma - s\Delta}\right)}{1 - s\Delta}, \qquad \|W_K\|_F \le s+1, \qquad \|W_V\|_F \le 2s, \qquad \|W_C\|_F \le s.$$

*Also, there exists a 1-bounded, $(1/s)$-injective $\mathcal{I}$-sparse Boolean function $g'_{\mathcal{I}} : \{0,1\}^T \to \mathbb{R}^s$ and Transformer head parameters such that $f_{\text{tf-head}}(X(b); W_Q, W_K, W_V, W_C) \circ \Pi_s$ uniformly approximates $g'_{\mathcal{I}}$. The norms satisfy the same bounds as above.*

**Lemma A.22** (Monotone to symmetric functions via MLP). *Let $f : \{0,1\}^T \to \mathbb{R}$ be any real-valued symmetric $s$-sparse Boolean function with index set $\mathcal{I}$. Let $W_Q, W_K, W_V, W_C$ be the parameters of a function*

$$f_{\text{tf-head}}(X; W_Q, W_K, W_V, W_C) := \mathsf{softmax}\left(v_{\texttt{[CLS]}}^\top W_Q W_K^\top X^\top\right) X W_V W_C,$$

*and let $\Pi_1 : \mathbb{R}^d \to \mathbb{R}$ be the projection onto the first coordinate. Suppose that under some mapping $b \mapsto X(b)$, $f_{\text{tf-head}} \circ \Pi_s$ $(\gamma/4)$-uniformly approximates a $B$-bounded $\gamma$-strictly monotone symmetric $\mathcal{I}$-sparse Boolean function $g : \{0,1\}^T \to \mathbb{R}$, for some $\gamma$. Then, there exists a function $f_{\text{tf+mlp}} \in \mathcal{F}_{\text{tf+mlp}}$ with the same weights $W_Q, W_K, W_V, W_C$, and 3-layer feedforward network weights $W_1, W_2, w$,*

*such that*

$$f_{\mathsf{tf+mlp}}(X(b)) = f(b), \qquad \forall b \in \{0,1\}^T,$$

*with dimensions* $(d_2, d_3) = (4(s+1), 2(s+1))$ *and weight norms satisfying*

$$\|W_1\|_\infty \le \frac{8\max(1,B)}{\gamma}, \qquad \|W_2\|_\infty \le \frac{8s}{\gamma}, \qquad \|w\|_\infty \le \max_{b\in\{0,1\}^T} |f(b)|.$$

**Lemma A.23** (Injective to arbitrary functions via MLP)**.** *Let* $f : \{0,1\}^T \to \mathbb{R}$ *be any real-valued s-sparse Boolean function with index set* $\mathcal{I}$ *such that* $|\mathcal{I}| = s \le d$*. Let* $W_Q, W_K, W_V, W_C$ *be the parameters of a function*

$$f_{\mathsf{tf\text{-}head}}(X; W_Q, W_K, W_V, W_C) := \mathsf{softmax}\left(v_{\mathsf{[CLS]}}^\top W_Q W_K^\top X^\top\right) X W_V W_C,$$

*and let* $\Pi_s : \mathbb{R}^d \to \mathbb{R}^s$ *be the projection onto the first s coordinates. Suppose that under some mapping* $b \mapsto X(b)$*,* $f_{\mathsf{tf\text{-}head}} \circ \Pi_s$ $(\gamma/4)$*-uniformly approximates a* $\gamma$*-injective function* $g : \{0,1\}^T \to \mathbb{R}^s$ *satisfying* $\|g(b)\|_\infty \le B$*. Then, there exists a function* $f_{\mathsf{tf+mlp}} \in \mathcal{F}_{\mathsf{tf+mlp}}$ *with the same weights* $W_Q, W_K, W_V, W_C$*, and 3-layer feedforward network weights* $W_1, W_2, w$*, such that*

$$f_{\mathsf{tf+mlp}}(X(b)) = f(b), \qquad \forall b \in \{0,1\}^T,$$

*with dimensions* $(d_2, d_3) = (4s2^s, 2\cdot 2^s)$ *and weight norms satisfying*

$$\|W_1\|_\infty \le \frac{8\max(1,B)}{\gamma}, \qquad \|W_2\|_\infty \le \frac{8s}{\gamma}, \qquad \|w\|_\infty \le \max_{b\in\{0,1\}^T} |f(b)|.$$

### A.2.3 Useful lemmas

We will use a construction which approximates a "hard selection" of $s$ indices using the softmax mixture; for this, we will need to quantify the approximation error when the inputs to the softmax function are bounded.

**Lemma A.24** (Softmax truncation). *Let $z \in (\mathbb{R} \cup \{-\infty\})^T$ such that $z_t \geq R$ for each $1 \leq t \leq s$, and $z_t \leq 0$ for each $s+1 \leq t \leq T$. Define $z' \in (\mathbb{R} \cup \{-\infty\})^T$ so that $z'_t = z_t$ for $1 \leq t \leq s$, and $z_t = -\infty$ for $s+1 \leq t \leq T$. Then, letting $e^{-\infty} = 0$ in the definition of $\mathsf{softmax}(\cdot)$, we have*

$$\left\|\mathsf{softmax}(z') - \mathsf{softmax}(z)\right\|_1 \leq 2\frac{T-s}{s\exp(R)} < \frac{2T}{\exp(R)}.$$

*Proof.* We have

$$\left\|\mathsf{softmax}(z') - \mathsf{softmax}(z)\right\|_1 = \sum_{t=1}^{s} \exp(z_t)\left(\frac{1}{1^\top \exp(z')} - \frac{1}{1^\top \exp(z)}\right) + \sum_{t=s+1}^{T} \frac{\exp(z_t)}{1^\top \exp(z)}.$$

The first summation is equal to

$$1 - \frac{1^\top \exp(z')}{1^\top \exp(z)} \leq \frac{T-s}{s\exp(R)},$$

while the same upper bound holds for the second summation, since each term is at most $\frac{1}{s\exp(R)}$.

□

Our results on approximating arbitrary sparse Boolean functions will depend on a generic construction for robustly approximating an arbitrary function $f : \mathbb{R}^d \to \mathbb{R}$ with a feedforward neural network. For simplicity of presentation, we use a standard* 3-layer ReLU network construction, which *exactly* represents a piecewise constant function in specified regions.

*For example, this follows from the discussion in Chapter 4 of (Nielsen, 2015).

**Lemma A.25** (Exact function representation with a 3-layer ReLU net)**.** *Let $f : \mathbb{R}^{d_f} \to \mathbb{R}$, and let $x_1, \ldots, x_n \in \mathbb{R}^{d_f}$ such that $\|x_i\|_\infty \le B$ for each $i \in [n]$, $\left\|x_i - x_j\right\|_\infty \ge 4\delta$ for each $i \ne j \in [n]$. Then, there is a 3-layer feedforward network with ReLU activations, with parameters $W_1 \in \mathbb{R}^{(d_f+1)\times d_2}$, $W_2 \in \mathbb{R}^{(d_2+1)\times d_3}$, $w \in \mathbb{R}^{d_3}$*[*]*, such that*

$$f_{\mathsf{mlp}}(x_i + z) = f(x_i)$$

*for all $i \in [n]$ and $\|z\|_\infty \le \delta$, where $\mathsf{ReLU}(x) := x_+ = \max(0, x)$ is applied entrywise, with*

$$d_2 = 4nd_f, \quad d_3 = 2n,$$

$$\|W_1\|_\infty \le \frac{\max(1, B)}{\delta}, \quad \|W_2\|_\infty \le \frac{d_f}{\delta}, \quad \|w\|_\infty \le \max_{i\in[n]} |f(x_i)|.$$

*Proof.* First, we construct a one-dimensional "bump" function basis, and propagate the Lipschitz constants. A threshold function with a linear "ramp" of width $\delta$ can be obtained from a linear combination of 2 ReLU functions:

$$\nu_\delta(x) := (x/\delta + 1)_+ - (x/\delta)_+ = \begin{cases} 0 & x \le -\delta \\ x/\delta + 1 & -\delta \le x \le 0 \\ 1 & x \ge 0 \end{cases}.$$

Next, we construct the bump function

$$\psi_\delta(x) := \nu_\delta(x) - \nu_\delta(2\delta - x).$$

By this construction, we have $\psi_\delta(x) = 1$ for $0 \le x \le 2\delta$ and $\psi_\delta(x) = 0$ for $x \le -\delta$ and $x \ge 3\delta$,

[*]Here, $W_1$, $W_2$ have bias terms; $w$ does not.

interpolating linearly on $[-\delta, 0]$ and $[2\delta, 3\delta]$. Next, define

$$\psi_\delta(x; x_0) := \psi_\delta(x - x_0 + \delta)$$

$$= \left(\frac{x - x_0}{\delta} + 2\right)_+ - \left(\frac{x - x_0}{\delta} + 1\right)_+ - \left(\frac{x_0 - x}{\delta} + 2\right)_+ + \left(\frac{x_0 - x}{\delta} + 1\right)_+$$

so that $\psi_\delta(x; x_0) = 1$ for $|x - x_0| \le \delta$, $\psi_\delta(x; x_0) = 0$ for $|x - x_0| \ge 2\delta$.

We construct the first layer $W_1 \in \mathbb{R}^{(d+1)\times(4nd)}$ using these bump functions: indexing the $4nd$ dimension by $(h \in [4], i \in [n], j \in [d])$, we construct

$$[W_1]_{:,(1,i,:)} := \begin{bmatrix} \frac{1}{\delta} I \\ -\frac{x_i}{\delta} + 2 \cdot 1^\top \end{bmatrix}, \qquad [W_1]_{:,(2,i,:)} := \begin{bmatrix} \frac{1}{\delta} I \\ -\frac{x_i}{\delta} + 1^\top \end{bmatrix},$$

$$[W_1]_{:,(3,i,:)} := \begin{bmatrix} -\frac{1}{\delta} I \\ \frac{x_i}{\delta} + 2 \cdot 1^\top \end{bmatrix}, \qquad [W_1]_{:,(4,i,:)} := \begin{bmatrix} -\frac{1}{\delta} I \\ \frac{x_i}{\delta} + 1^\top \end{bmatrix},$$

so that

$$\left([x\ 1]^\top \left[[W_1]_{j,(1,i,:)}\ [W_1]_{j,(2,i,:)}\ [W_1]_{j,(3,i,:)}\ [W_1]_{j,(4,i,:)}\right]\right)_+ [1\ -1\ -1\ 1]^\top = \psi_\delta(x; [x_i]_j).$$

The second layer is used to construct $n$ activations which are indicators of whether $x$ is in the neighborhood of each $x_i$. For each $x_i$, we will simply average the $d_f$ one-dimensional indicators for each coordinate, and implement a threshold function $\nu_{\delta/d_f}(1-x)$. We choose $W_2 \in \mathbb{R}^{(4nd_f+1)\times(2n)}$, with the $4nd_f + 1$ dimension indexed by $(h, i, j)$ and an extra bias dimension $\perp$, and the $2n$ dimension indexed by $(h' \in \{1, 2\}, i' \in [n])$ so that

$$[W_2]_{(h,i,:),(h',i')} := [1\ -1\ -1\ 1]_h \cdot 1[i = i'] \cdot \frac{1}{\delta} \cdot 1,$$

$$[W_2]_{\perp,(1,i')} := 1 - \frac{d_f}{\delta}, \qquad [W_2]_{\perp,(2,i')} := -\frac{d_f}{\delta}.$$

Finally, the third (output) layer $w \in \mathbb{R}^{2n}$, with dimensions indexed by $(h \in \{1,2\}, i \in [n])$, multiplies the indicators of each $x_i$ by the desired $f(x_i)$:

$$w_{(1,i)} := f(x_i), \quad w_{(2,i)} := -f(x_i).$$

For any $x_0 \in \mathbb{R}^{d_f}$, let $B_\delta(x_0)$ be the set of $x$ such that $\|x - x_0\|_\infty \leq \delta$. By this construction, for each $x \in B_\delta(x_i)$, we have $f(x) = x_i$, as required. □

Note that we use 3-layer ReLU networks for function approximation in order to minimize the introduction of unnecessary notation. Some minor remarks:

- It would be routine to replace this construction with any architecture which can represent an arbitrary function *approximately* (Hornik et al., 1989; Cybenko, 1989); this includes the 2-layer feedforward networks (and nonlinear activations other than the ReLU) which are typically used by Transformers in practice.
- It is possible to embed this construction in $f_{\mathsf{tf+mlp}}$ with a 2-layer ReLU network, by using $(W_C, W_1, W_2)$ and introducing a nonlinearity after $W_C$, without changing the results.
- When $d_f = 1$, $W_2$ is unnecessary (one can represent $f$ directly using the bump function basis).

### A.2.4 Beyond Boolean domains

These representational results are stated with a Boolean domain $\{0,1\}^T$ for clarity and simplicity of presentation; this is not essential or fundamental to these constructions. Generalizations to the following input domains are straightforward:

- Discrete categorical tokens $\{1, \ldots, M\}^T$. Use $M$ orthogonal token embeddings, instead of only 2. The "sparse function approximation" construction uses $sM^s$ instead of $s2^s$ parameters. The smaller (poly$(s)$-parameter) representation of symmetric Boolean functions has no clear analogue.

- Continuous inputs on a bounded domain (e.g. $[0,1]^T$). The (fixed or trainable) positional embeddings can still select the $s$ sparse indices. Replace the discrete ReLU network construction with any continuous universal function approximation construction. The "bag-of-vectors" formulation has no clear analogue.

The capacity results from Section 4.4 hold for any input domain, as long as the embeddings are bounded in the $\|\cdot\|_{2,\infty}$ norm. Note that Transformers are predominantly used with discrete inputs.

### A.2.5 Proofs

Throughout the constructions in each case, we will refer to standard coordinate bases in several spaces:

- $E_0, E_1 \in \mathbb{R}^d$ denote the embeddings of the $0, 1$ tokens $E_{0,:}, E_{1,:}$.

- $e_1^{(k)}, \ldots, e_k^{(k)}$ denotes the standard basis in $\mathbb{R}^k$.

- $e_i^{(d)}$ denotes the standard basis in $\mathbb{R}^d$.

- $e_1^{(T)}, \ldots, e_T^{(T)}, e_{\texttt{[CLS]}}^{(T)}$ denotes the standard basis in $\mathbb{R}^{T+1}$ with the special `[CLS]` index.

- Recall that the $v_i$ form a $\Delta$-approximate orthonormal basis for $\mathbb{R}^d$, $v_{\texttt{[CLS]}}, e_0, e_1$ are exactly orthogonal to each of them as well as each other, and $d$ is chosen such that these conditions can be met.

Let $n(i)$ be a unique bijection between $\mathcal{I}$ and $[s]$. Let $v_{\mathcal{I}} := \sum_{i \in \mathcal{I}} v_i$.

Approximate vector equality. We will use $u \approx_\varepsilon v$ to denote that two vectors $u, v \in \mathbb{R}^{d_u}$ satisfy $\|u - v\|_\infty \le \varepsilon$.

*Proof of Lemma A.19.* We construct attention heads such that the softmax mixture always selects the indices in $\mathcal{I}$.

**Single head, deterministic** $P$**.** We seek to approximate the 1-bounded, $(2/s)$-strictly monotone function

$$\frac{1}{s}\sum_{i\in\mathcal{I}} \chi_i,$$

where $\chi_i = +1$ if $b_i = 0$ and $-1$ if $b_i = 1$. Set

$$W_Q := R v_{\texttt{[CLS]}} e_1^{(k)\top}, \quad W_K := v_{\mathcal{I}} e_1^{(k)\top}, \quad W_V := (E_0 - E_1) e_1^{(k)\top}, \quad W_C := e_1^{(k)} e_1^{(d)\top},$$

where $R$ will be chosen later. Then, by approximate orthogonality,

$$v_{\texttt{[CLS]}}^\top W_Q W_K^\top X^\top = v_{\texttt{[CLS]}}^\top W_Q W_K^\top (P + E_b)^\top = v_{\texttt{[CLS]}}^\top W_Q W_K^\top P^\top \approx_{Rs\Delta} R \sum_{i\in\mathcal{I}} e_i^{(T)\top}.$$

By Lemma A.24,

$$\left\| \mathsf{softmax}\left( v_{\texttt{[CLS]}}^\top W_Q W_K^\top X^\top \right) - \frac{1}{s}\sum_{i\in\mathcal{I}} e_i^{(T)\top} \right\|_1 \le \frac{2T}{\exp(R - 2Rs\Delta)}.$$

Finally, we have

$$X W_V W_C = E_b W_V W_C = \left( \sum_{i\in[T]} \chi_i e_i^{(T)} \right) e_1^{(d)\top},$$

so that by Hölder's inequality,

$$f_{\text{tf-head}}(X) \circ \Pi_1 = \mathsf{softmax}\left( v_{\texttt{[CLS]}}^\top W_Q W_K^\top X^\top \right) X W_V W_C e_1^{(d)} \approx_{\frac{2T}{\exp(R-2Rs\Delta)}} \frac{1}{s}\sum_{i\in\mathcal{I}} \chi_i.$$

To get $(\gamma/4)$-uniform approximation, we choose

$$R = \frac{\log\left(\frac{8T}{\gamma}\right)}{1 - s\Delta}.$$

**Multiple heads, deterministic** $P$**.** For $h = 1, \ldots, s$, and the same $R$ as above:

$$W_Q^{[h]} := Rv_{\texttt{[CLS]}} e_1^{(k)\top}, \quad W_K^{[h]} := v_{n-1(h)} e_1^{(k)\top}, \quad W_V^{[h]} := (E_0 - E_1) e_2^{(k)\top}, \quad W_C^{[h]} := e_1^{(k)} e_h^{(d)\top}.$$

This is the same construction as above, but each head only selects one of the coordinates in $\mathcal{I}$. Thus, by the same analysis,

$$f_{\textsf{tf-head}}(X) \circ \Pi_s \approx_{\frac{2T}{\exp(R - 2Rs\Delta)}} \sum_{i \in \mathcal{I}} \chi_i e_{n(i)}^{(d)}.$$

This function is clearly 1-bounded and 2-injective. □

*Proof of Lemma A.20.* The constructions closely follow Lemma A.19, but are simpler.

**Single head, trainable** $P$**.** For each $i \in \mathcal{I}$, set the trainable positional embeddings to be

$$P_{i,:} := \begin{cases} v_1 & i \in \mathcal{I} \\ 0 & \text{otherwise} \end{cases}.$$

Set

$$W_Q := Rv_{\texttt{[CLS]}} e_1^{(k)\top}, \quad W_K := v_1 e_1^{(k)\top}, \quad W_V := (E_0 - E_1) e_1^{(k)\top}, \quad W_C := e_1^{(k)} e_1^{(d)\top}.$$

Now, we have (with equality)

$$v_{\texttt{[CLS]}}^\top W_Q W_K^\top X^\top = R \sum_{i \in \mathcal{I}} e_i^{(T)\top},$$

so that Lemma A.24 gives

$$\left\| \mathsf{softmax}\left(v_{\texttt{[CLS]}}^\top W_Q W_K^\top X^\top\right) - \frac{1}{s}\sum_{i\in\mathcal{I}} e_i^{(T)\top} \right\|_1 \le \frac{2T}{\exp(R)}.$$

Like before, we have

$$f_{\mathsf{tf\text{-}head}}(X)\circ\Pi_1 = \mathsf{softmax}\left(v_{\texttt{[CLS]}}^\top W_Q W_K^\top X^\top\right) X W_V W_C e_1^{(d)} \approx_{\frac{2T}{\exp(R)}} \frac{1}{s}\sum_{i\in\mathcal{I}} \chi_i.$$

To get $(\gamma/4)$-uniform approximation, we choose

$$R = \log\left(\frac{8T}{\gamma}\right).$$

**Multiple heads, trainable** $P$**.** For each $i \in \mathcal{I}$, set the trainable positional embeddings to be

$$P_{i,:} := \begin{cases} e_{n(i)}^{(d)} & i \in \mathcal{I} \\ 0 & \text{otherwise} \end{cases}.$$

For $h = 1, \ldots, s$, and the same $R$ as above:

$$W_Q^{[h]} := R v_{\texttt{[CLS]}} e_1^{(k)\top}, \quad W_K^{[h]} := e_h^{(d)} e_1^{(k)\top}, \quad W_V^{[h]} := (E_0 - E_1) e_1^{(k)\top}, \quad W_C^{[h]} := e_1^{(k)} e_h^{(d)\top}.$$

This is the same construction as above, but each head only selects one of the coordinates in $\mathcal{I}$. Thus, by the same analysis,

$$f_{\mathsf{tf\text{-}head}}(X)\circ\Pi_s \approx_{\frac{2T}{\exp(R)}} \sum_{i\in\mathcal{I}} \chi_i e_{n(i)}^{(d)}.$$

□

*Proof of Lemma A.21.* This input mapping does not use position embeddings, and does not need

multiple heads to implement arbitrary (non-symmetric) functions. The constructed monotone and injective functions are slightly different, but the proof strategy is very similar. The key difference is that the softmax mixture is uniform only on the positions $i \in \mathcal{I}$ where $b_i = 1$.

**Bag of vectors, scalar output.** The function we will approximate is defined as follows:

$$g_{\mathcal{I}}(r) := \frac{r-s}{r+1}, \quad \text{where } r = \sum_{i \in \mathcal{I}} b_i, \quad s = |\mathcal{I}|.$$

Note that this function is $(1/s)$-strictly monotone, and has absolute value bounded by $s$. Set

$$W_Q := R v_{\texttt{[CLS]}} e_1^{(k)\top}, \quad W_K := (v_{\mathcal{I}} + v_{\texttt{[CLS]}}) e_1^{(k)\top},$$

$$W_V := \sum_{i \in \mathcal{I}} v_i e_{n(i)}^{(k)\top} - v_{\texttt{[CLS]}} \left( \sum_{i \in \mathcal{I}} e_{n(i)}^{(k)} \right)^\top, \quad W_C := \sum_{i \in \mathcal{I}} e_{n(i)}^{(k)} e_1^{(d)\top},$$

where $R$ will be chosen later. Then, by approximate orthogonality,

$$v_{\texttt{[CLS]}}^\top W_Q W_K^\top X^\top \approx_{Rs\Delta} R \left( v_{\texttt{[CLS]}} + \sum_{i \in \mathcal{I}} b_i e_i^{(T)\top} \right),$$

so that by Lemma A.24,

$$\left\| \mathsf{softmax}\left( v_{\texttt{[CLS]}}^\top W_Q W_K^\top X^\top \right) - \frac{1}{r+1} \left( e_{\texttt{[CLS]}}^{(T)\top} + \sum_{i \in \mathcal{I}} b_i e_i^{(T)\top} \right) \right\|_1 \leq \frac{2T}{\exp(R - 2Rs\Delta)}.$$

Finally, we have

$$X W_V W_C e_1^{(d)} = -s e_{\texttt{[CLS]}}^{(T)} + \sum_{i \in [T]} b_i \cdot v_i^\top v_{\mathcal{I}} \cdot e_i^{(T)} \approx_{s\Delta} -s e_{\texttt{[CLS]}}^{(T)} + \sum_{i \in \mathcal{I}} b_i e_i^{(T)},$$

so that

$$
\begin{aligned}
&|f_{\text{tf-head}}(X) \circ \Pi_1 - g_{\mathcal{I}}(r)| \leq \\
&\qquad \left\| \mathsf{softmax}\left(v_{\texttt{[CLS]}}^\top W_Q W_K^\top X^\top\right) - \frac{1}{r+1}\left(e_{\texttt{[CLS]}}^{(T)\top} + \sum_{i\in\mathcal{I}} b_i e_i^{(T)\top}\right)\right\|_1 \left(\left\|XW_V W_C e_1^{(d)}\right\|_\infty + s\Delta\right) \\
&\qquad\qquad + \left\|\mathsf{softmax}\left(v_{\texttt{[CLS]}}^\top W_Q W_K^\top X^\top\right)\right\|_1 (s\Delta) \\
&\leq \frac{2Ts(1+\Delta)}{\exp(R - 2Rs\Delta)} + s\Delta.
\end{aligned}
$$

To get $(\gamma/4)$-uniform approximation, we choose

$$
R = \frac{\log\left(\frac{8Ts(1+\Delta)}{\gamma - s\Delta}\right)}{1 - s\Delta}.
$$

**Bag of vectors, $s$-dimensional output.** We use the same construction as above, except

$$
W_C := \sum_{i\in\mathcal{I}} e_{n(i)}^{(k)} e_{n(i)}^{(d)\top}.
$$

This will allow us to approximate the function

$$
g'_{\mathcal{I}}(b) = \frac{1}{r+1}\sum_{i\in\mathcal{I}} (b_i - 1) e_{n(i)}^{(d)},
$$

which is $(1/s)$-injective and has absolute value is bounded by 1. Then, for each $i \in \mathcal{I}$, we have

$$
XW_V W_C e_i^{(d)} = -e_{\texttt{[CLS]}}^{(T)} + v_i^\top v_{\mathcal{I}} \cdot e_i^{(T)} \approx_{s\Delta} -e_{\texttt{[CLS]}}^{(T)} + b_i e_i^{(T)}.
$$

Repeating the above analysis for each coordinate, we have

$$f_{\text{tf-head}}(X) \circ \Pi_s \approx_\varepsilon g'_{\mathcal{I}}(r),$$

where a slightly tighter bound

$$\varepsilon = \frac{2T(1+s\Delta)}{\exp(R - 2Rs\Delta)} + s\Delta$$

comes from the fact that $\left\| XW_V W_C e_i^{(d)} \right\|_\infty$ is now bounded by 1 instead of $s$. The previous choice of $R$ suffices for $(\gamma/4)$-uniform approximation. □

*Proof of Lemma A.22.* This follows by instantiating Lemma A.25 with $\delta = \gamma/8, d_f = 1, n = s+1$. Notice that a $(\gamma/4)$-uniform approximation of a $\gamma$-strictly monotone function satisfies the conditions needed for Lemma A.25. □

*Proof of Lemma A.23.* This follows by instantiating Lemma A.25 with $\delta = \gamma/8, d_f = s, n = 2^s$. Notice that a $(\gamma/4)$-uniform approximation of a $\gamma$-injective function satisfies the conditions needed for Lemma A.25. □

## A.3 Details for experiments

### A.3.1 Empirical scaling laws (Figure 4.2)

In this section, we provide details for the sample complexity scaling law experiments, which are the main empirical verification of the log $T$ dependence of the sample complexity arising from the analysis.

Data. Synthetic supervised learning tasks corresponding to learning a 3-sparse conjunction were generated by the protocol described in the main paper, parameterized by sample size $m$ and context

$T$: in each trial, one of the $\binom{T}{3}$ subsets of indices was selected uniformly at random*, under i.i.d. Bernoulli inputs $x_i \sim \text{Bern}(p)$. Here, $p = (1/2)^{1/3}$ was chosen so that the classes are balanced $\Pr[y = 0] = \Pr[y = 1]$. $m$ samples were drawn this distribution to form a training set (rejecting training sets which were compatible with multiple hypotheses), and $10^4$ samples were drawn from the same distribution as a holdout validation set. The grid of problem instances was constructed as follows: $T \in \{100, 150, 200, \ldots, 750, 800\} \cup \{900, 1000, 1100\}$, $m \in \{50, 60, 70, \ldots, 200\}$.

Training. A 1-layer Transformer network (with a scalar output at a special trainable token $x_{\texttt{[CLS]}}$ at an extra index `[CLS]`) was trained with Adam (Kingma & Ba, 2014) and full-batch gradients of the cross entropy loss for binary classification. 40 independent trials were performed (re-randomizing the dataset generation); each trial was restarted (with fresh random initialization and dropout masks) 5 times before being labeled as a failure. Cross-validation was performed on a holdout sample of size $10^4$ every 10 iterations. At the end of 1000 training iterations, the trial was counted as a success if the maximum validation accuracy throughout training was greater than 0.99. (In 100% of runs, the training loss was driven to $< 10^{-4}$, with 100% training accuracy, within 1000 iterations.)

Architecture. Like (Chen et al., 2021a), our experimental setup is based on a popular PyTorch implementation (`https://github.com/karpathy/minGPT`), with some optimizations for faster 1-layer training and inference. This implementation includes widely-used architectural details (GeLU activations; dropout) which were not discussed in the theoretical analysis; refer to the referenced repository for details. All hyperparameter settings left undiscussed are taken from the default settings in this codebase.

*Note that due to the permutation-symmetry of the Transformer architecture (as long as the position embeddings are initialized with a permutation-symmetric distribution), it is equivalent to select $\mathcal{I} = [s]$. Also, by symmetry of the initial token embeddings, AND and OR are interchangeable in these experiments and results.

Hyperparameters. A fixed architecture was used ($d = 64, k = 4$, 16 parallel heads), with trainable positional embeddings initialized with Gaussian entries $\mathcal{N}(0, \sigma^2), \sigma = 0.02$, 3 input token embeddings (corresponding to 0, 1, `[CLS]`), and 2 output embeddings (corresponding to the possible labels 0, 1). For regularization mechanisms, typical choices were used: 0.1 for {attention, embedding, output} dropout; $10^{-4}$ weight decay. The Adam optimizer was instantiated with typical parameters $\eta = 10^{-3}, \beta_1 = 0.9, \beta_2 = 0.999$.

Infrastructure and computational costs. All experiments were performed on an internal cluster, with NVIDIA Tesla P100, NVIDIA Tesla P40, and NVIDIA RTX A6000 GPUs. Each training run took at most 10 minutes (with most computation time spent on cross-validation), for a total of $\sim$ 150 GPU-hours.

### A.3.2 Other figures

Example training curves. Figure 4.3 *(left)* shows the best training curves (in terms of highest validation accuracy) out of 5 restarts for all 40 replicates, in the settings $T = 300, m = 200$ and $T = 300, m = 50$. As the sample size $m$ decreases, the trained models overfit with higher probability; when they overfit, they differ significantly in validation accuracy.

Attention weights. In Figure 4.3, the normalized attention weights at the `[CLS]`position are shown for a Transformer model trained for 500 iterations with $T = 50, m = 300$ (achieving 100% validation accuracy on $10^4$ holdout samples), for 100 validation samples (which each induce different attention weights, shown in the scatter plot). One key difference (for simplicity of visualization) is that this network only has one attention head, with embedding dimensions $d = k = 64$.

Parity. These experiments use the same architecture as the main AND experiments, except the. In the loss curves presented in Figure 4.4, gradient-based training is done on streaming online losses

(so that there is no training/validation split), with batch size 2048.

## A.4 Additional related work

In this section, we discuss some additional related work.

Domains beyond language. We provide a few references on the successful application of Transformers to domains beyond natural language processing; this frontier is continually and rapidly evolving.

- With minimal changes compared to the analogous natural language settings, Transformer sequence models have been applied to theorem proving (Polu & Sutskever, 2020) and program synthesis (Chen et al., 2021b).
- Beyond sequence modeling: In self-supervised learning of image representations, the Vision Transformer (ViT) (Dosovitskiy et al., 2020) has sometimes outperformed older convolution-based architectures, particularly when pretrained on massive datasets. Further architectural variants have been proposed for vision and other continuous modalities (Tolstikhin et al., 2021; Lee-Thorp et al., 2021; Jaegle et al., 2021b,a; d'Ascoli et al., 2021).
- Natural sciences: A state-of-the-art pipeline for protein structure prediction (Jumper et al., 2021) features a self-attention-based component.
- Reinforcement learning: Transformer architectures have shown promise for planning (Janner et al., 2021) and offline RL (Chen et al., 2021a).

Expressive power of Transformers. Several works establish results on the representational power of self-attention architectures in regimes where the statistical guarantees are necessarily weak

or vacuous (i.e. there are too many functions in the class). Dehghani et al. (2018); Yun et al. (2019); Bhattamishra et al. (2020a,b) establish universal function approximation and Turing-completeness, which have been known for previous architectures (Siegelmann & Sontag, 1995). Our work is a significantly finer-grained analysis, in which we establish a hierarchy of function classes (indexed by sparsity $s$) representable by these architectures, with tight (in terms of $T$) statistical guarantees. Hron et al. (2020); Yang (2020) analyze properties of the kernels induced by Transformers at the infinite-width limit. Quoting the discussion following Theorem 4.1 of (Wei et al., 2021), who show statistically meaningful approximations of TM using Transformers: *"The correct norm-based Rademacher complexity bound to use for Transformers is unclear."* The connection between Transformers and circuits also appears in (Elhage et al., 2021; Olsson et al., 2022), with a different technical approach (interpreting the classes of computations performed by attention weights and heads). Likhosherstov et al. (2021) analyze the sparsity patterns representable by a self-attention head, with results superficially similar to ours: when the embedding dimension is at least logarithmic in the context length, all sparse matrices can be approximately realized by an attention head. However, their analysis is not about the capacity of the function class: it quantifies over the input $X$, and holds the parameters $(W_Q, W_K, \ldots)$ to be constant (rather than vice versa). This finding serves as an interesting complement to our result: even though the attention mixture weights can take on exponentially many sparsity patterns for distinct inputs, the generalization error scales as $\log(T)$.

Interpreting attention mixtures. A line of empirical work ("BERTology") has made progress on understanding and interpreting state-of-the-art Transformer language models by examining the activations of their attention mechanisms (Clark et al., 2019; Tenney et al., 2019; Rogers et al., 2020). In some cases, these works have found instances in which Transformers seem to have learned features that are reminiscent of (sparse) hand-crafted features used in natural language processing, without explicit supervision. Our analysis formalizes the intuition that self-attention heads

can represent sparse interactions within the context in a statistically meaningful way.

Other theoretical work on self-attention. Kerg et al. (2020) analyze self-attention and its benefits for learning long-term dependencies by establishing gradient norms bounds and showing how attention helps address the problem of gradient vanishing in recurrent networks. In contrast to our results that analyze the statistical and representational properties of attention-based architectures, this work focuses on the computational aspects of gradient-based methods on recurrent networks with self-attention.

Other attention-based architectures. Our analysis is amenable to computationally efficient variants of the Transformer which use parsimonious (e.g. low-rank) approximations of the softmax kernel, like the Performer (Choromanski et al., 2020). Building upon the success of modern attention-based architectures, a large body of work (e.g. Goyal et al. (2020, 2021), and the works cited within) has sought to design architectures which induce model sparsity and modularity. Our analysis is relevant to any architecture that uses a softmax (or similar) bottleneck for statistical capacity, and could inform design principles for norm-based capacity control of these architectures.

Synthetic experiments with Transformers. Power et al. (2021) train small Transformer networks on synthetic algebraic tasks, and discover an abrupt phase transition from overfitting to correct generalization similar to ours. Tay et al. (2020) propose some synthetic tasks for benchmarking the ability of Transformer variants to capture long-range dependences. Chen et al. (2021a) present a synthetic demonstration of extrapolation (inferring a maximum-reward path from random walk observations) when using Transformers for offline reinforcement learning. Lu et al. (2021) probe the transfer learning capabilities of pretrained Transformers, and consider some simple Boolean tasks. Our experimental protocol of learning sparse Boolean functions provides a simple

and fundamental setting for elucidating computational and statistical properties of sequence modeling architectures.

# B

# Hidden Progress

## B.1 Additional background, preliminaries, and related work

### B.1.1 Parities: orthogonality and computational hardness

For each integer $n \geq 1$ and nonempty subset of indices $S \subseteq [n]$, define the $(n, S)$-*parity function* $\chi_S(x) = \prod_{i \in S} x_i$, i.e. the parity of the bits at the indices in $S$. We define the $(n, S)$-*parity distribution* $\mathcal{D}_S$ over examples $(x, y)$ as follows: the features $x \sim \text{Unif}(\{\pm 1\}^n)$ are uniform random bits, with

labels $y \in \{\pm 1\}$ given by the parity function $y = \chi_S(x)$. For $0 \leq k \leq n$, the corresponding $(n, k)$-*parity learning problem* is the task of identifying an unknown size-$k$ set $S$ (chosen at random), using samples from $\mathcal{D}_S$. With knowledge of $k$ but not $S$, a learner must use the labels to distinguish between $\binom{n}{k}$ possible "relevant feature sets"; thus, the statistical limit for this problem is $\log \binom{n}{k} = \Theta(k \log n)$ samples.

This work leverages the parity problem as a "computationally hard case" for identifying the features $S$ which are relevant to the label. Observe that for any $S' \subseteq [n]$, it holds that

$$\mathop{\mathbb{E}}_{x \sim \mathrm{Unif}(\{\pm 1\}^n)} \left[\chi_S(x)\chi_{S'}(x)\right] = \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}_S} \left[\chi_{S'}(x)\, y\right] = \begin{cases} 1 & S' = S \\ 0 & \text{otherwise} \end{cases}. \tag{B.1}$$

That is, a learner who guesses indices $S'$ cannot use correlations as feedback to reveal which (or how many) indices in $S'$ are correct, unless $S'$ is *exactly* the correct subset. In this sense, for the $(n, k)$-parity problem, the $\binom{n}{k} - 1$ wrong answers are indistinguishable from each other. Thus, the structure of this problem forces this form of learner (but not necessarily all learning algorithms) to perform exhaustive search over subsets.

Property B.1 (a.k.a. the orthogonality of parities under the correlation inner product) implies that any function $f : \{\pm 1\}^n \to \mathbb{R}$ has a unique Fourier expansion (see, e.g. (O'Donnell, 2014)):

$$f(x) = \sum_{S \subseteq [n]} \widehat{f}(S)\chi_S(x), \qquad \widehat{f}_S = \mathop{\mathbb{E}}_{x \sim \mathrm{Unif}(\{\pm 1\}^n)} \left[f(x)\chi_S(x)\right]. \tag{B.2}$$

In the statistical query (SQ) model (Kearns, 1998), Property (B.1) implies computational lower bounds. In this model, a learner, rather than having access to examples drawn from the distribution, can query an oracle, which responds with noisy estimates of the query over the distribution. Namely, each iteration the learner outputs a query $q_i : \{\pm 1\}^n \to [-1, 1]$, and the oracle returns

some value $v_i$ satisfying $|v_i - \mathbb{E}_{(x,y)\sim\mathcal{D}}[q_i(x,y)]| \leq \tau$, for some tolerance parameter $\tau > 0$. It can be shown that Equation B.1 implies that each query will have a non-trivial correlation only with a small fraction (namely, $1/\tau^2$) of the possible parities, which then implies that an SQ algorithm which solves the $(n,k)$-parity problem using $T$ queries of tolerance $\tau$ must satisfy $T/\tau^2 \geq \Omega(n^k)$. This constitutes a lower bound on the number of queries (and/or on the tolerance), which indicates that essentially, SQ algorithms cannot do much better than exhaustive search (going over all the possible choices of size-$k$ subsets).

It should be mentioned that the $(n,k)$-parity problem can be solved efficiently by a learning algorithm that has access to examples (i.e., an algorithm that does not operate in the SQ framework). Specifically, this problem can be solved by the Gaussian elimination algorithm. Moreover, it has been shown that the (Stochastic) Gradient Descent algorithm, discussed in the next section, can also be utilized for solving parities, given accurate enough estimates of the gradient and a very particular choice of neural network architecture Abbe & Sandon (2020). That said, when the accuracy of the gradients is not sufficient, GD suffers from the same SQ lower bound mentioned above (i.e., GD is essentially an SQ algorithm Abbe et al. (2021)).

Learning sparse *noisy* parities, even at a very small noise level (i.e., $o(1)$ or $n^{-\delta}$) is believed to be computationally hard. This was first explicitly conjectured by Alekhnovich (2003), and has been the basis for several cryptographic schemes (e.g., (Ishai et al., 2008; Applebaum et al., 2009, 2010)). For noiseless sparse parities, Kol et al. (2017) show time-space hardness in the setting where $k = \omega(1)$. We present some experiments with noisy parities in Appendix B.3.6, finding that our empirical results (and theoretical analysis) are robust to $\Theta(1)$ noise.

### B.1.2 Neural networks and standard training

Next, we establish notation for the standard neural network training pipeline. Our main results are presented in the online learning setting, with a stream of i.i.d. batches of examples. At each iteration

$t = 1, \ldots, T$, a learning algorithm receives a batch of $B$ examples $\{(x_{t,i}, y_{t,i})\}_{i=1}^{B}$ drawn i.i.d. from $\mathcal{D}_S$, then outputs a classifier $\widehat{y}_t : \{\pm 1\}^n \to \{\pm 1\}$. If $\mathbb{E}_{(x,y)\sim\mathcal{D}_S}[1[\widehat{y}_t(x) \neq y]] \leq \varepsilon$ (i.e. $\widehat{y}_t$ agrees with the correct parity on at least a $(1-\varepsilon)$ fraction of inputs), the learner is said to have solved the parity problem with error $\varepsilon$ in $t$ iterations ($tB$ samples); the smallest $t$ for which this is true is the *convergence time* $t_c$. A learner may also output an initial classifier $\widehat{y}_0$ before observing any data.

This formulation permits *improper* function classes (i.e. other than parities over subsets $S'$) for the parity learning problem. In particular, we will focus on hypothesis classes of continuous functions $f : \{\pm 1\}^n \times \Theta \to \mathbb{R}$, which map to classifiers $\widehat{y}(x) = \operatorname{sign}(f(x;\theta))$*. When $\Theta$ is a vector space over $\mathbb{R}$, a ubiquitous online learning algorithm is gradient descent (GD). For a choice of loss function $\ell : \{\pm 1\} \times \mathbb{R} \to \mathbb{R}$, initialization $\theta_0$, learning rate schedule $\{\eta_t\}_{t=1}^{T} \subseteq \mathbb{R}$ and weight-decay schedule $\{\lambda_t\}_{t=1}^{T} \subseteq \mathbb{R}$, GD defines iterative updates

$$\theta_{t+1} \leftarrow (1-\lambda_t)\theta_t - \eta_t \nabla_\theta \left( \frac{1}{B} \sum_{i=1}^{B} \ell(y_{t,i}, f(x_{t,i};\theta_t)) \right), \tag{B.3}$$

where $f$ (the *architecture*) and $\ell$ are assumed to be such that this gradient (more generally, subgradient) is well-defined. In this context, *online* and *stochastic* gradient descent (OGD/SGD) are equivalent names for the update rule (B.3).

A fundamental object of study in deep learning is the multi-layer perceptron (MLP). In this setting, a 2-layer MLP with *width* $h$ and activation function $\sigma : \mathbb{R} \to \mathbb{R}$, parameterized by $W \in \mathbb{R}^{r\times n}, b \in \mathbb{R}^r, u \in \mathbb{R}^r$, specifies the function

$$f(x; W, b, u) = u^\top \sigma(Wx + b),$$

where $\sigma(\cdot)$ is applied entrywise. It is standard to use GD to jointly update the network's parameters.

*When $f(x;\theta) = 0$ in practice (e.g. with sign initialization), we break the tie arbitrarily. We ensure in the theoretical analysis that this does not happen.

Our results include positive results about "single neurons": MLPs with width $r = 1$. We note that for our theoretical analysis, when training MLPs with GD, we allow for different learning rate and weight decay schedule for the different layers.

Finally, we will analyze randomized learning algorithms, such as GD with random initialization $\theta_0$, whose iterates $\theta_t$ (and thus classifiers $\widehat{y}$) are random variables even when the samples are not. A learning algorithm has *permutation symmetry* if, for all sequences of data $\{(x_{t,i}, y_{t,i})\}$, the classifiers $\widehat{y}_t \circ \pi$ resulting from feeding $\{(\pi(x_{t,i}), y_{t,i})\}$ to the learner have identical distributions as $\pi$ ranges over all permutations of indices. The neural architectures and initializations (and thus, SGD) considered in this work are permutation-symmetric; for this reason, it is convenient for notation to choose $S = [k]$ as the canonical $(n, k)$-parity learning problem, without loss of generality.

### B.1.3 Additional related work

Feature learning using GD on neural networks. A line of recent work has focused on understanding the feature learning ability of gradient descent dynamics on neural networks. These analyses go beyond the Neural Tangent Kernel (NTK) regime, where they show a separation between learning with fixed features versus GD on neural networks, for these problems. Several of these works assume structure (often "sparse") in the input data which is useful for the prediction task, and helps avoid computational hardness. In contrast, our work focuses on studying hard problems at their computational limit. Here we discuss the most relevant works in detail:

A line of work (Diakonikolas et al. (2020); Yehudai & Ohad (2020); Frei et al. (2020)) focuses on learning a single non-linearity $y = \sigma(w^\top x)$ (typically $\sigma(\cdot)$ is the ReLU or sigmoid) using gradient-based methods. These works obtain polynomial-time convergence guarantees when the distribution satisfies a *spread* condition. These results do not extend to the Boolean hypercube.

Daniely & Malach (2020) also study the problem of learning sparse parities using neural networks. One key difference from our work is that they assume a modified version of the problem,

where the input distribution is not uniform over the hypercube, but instead leaks information about the label. In particular, the distribution ensures that the relevant parity bits always have the same value. Shi et al. (2021) generalize this setting by considering a setting where labels are generated based on certain class specific patterns and the data itself is generated using these patterns with some extra background patterns. This also embeds information in the data itself regarding the label, unlike our setting, where the labels are uncorrelated with the input features. Under these structural assumption, the papers study how GD on a two-layer network can learn useful features in polynomial time. Both these analysis also exploits the first gradient step to find useful features. Shi et al. (2021) additionally require a second step to refine the features.

Ba et al. (2022) show how the first gradient step is important for feature leaning. In particular, they show that first update is essentially rank-1 and aligns with the linear component of the underlying function. The functions we consider (parity) do not have a linear component.

Abbe et al. (2022b) define a notion of initial alignment between the network at initialization and the target function and show that it is essential to get polynomial time learnability with noisy gradients on a fully connected network. Our MLP results also exploit the correlation between the gradient and the label to ensure that the gradient update gives us signal.

Frei et al. (2022a) also study learnability of a parity-like function with $k = 2$ under noisy labels. The paper analyzes early stopping GD for learning the underlying labeling function. Our setup is quite different from theirs and can handle $k > 2$.

In concurrent work, Damian et al. (2022) consider the problem of learning polynomials which depend on few relevant directions using gradient descent on a two-layer network. They assume that the distributional Hessian of the ground truth function spans exactly the subspace of the relevant direction. Using this, they show that gradient descent can learn the relevant subspace with sample complexity scaling as $d^2$ and not $d^p$ where $p$ is the degree of the underlying polynomial as long as the number of relevant directions is much less than $d$. Their proof technique is similar to our two-layer

MLP result which also exploits correlation in the first gradient step. However, for our setting, the distributional Hessian has rank 0 and does not satisfy their assumptions.

Statistical mechanics of machine learning. An extensive body of work originating in the statistical physics community has studied phase transitions in the learning curves of neural networks (Gardner & Derrida, 1989; Watkin et al., 1993; Engel & Van den Broeck, 2001). These works typically focus on student-teacher learning in the "thermodynamic limit", in which the number of training examples is $\alpha$ times larger than the input dimension and both are taken to infinity. One of the classic toy architectures analyzed in this literature is the *parity machine* (Mitchison & Durbin, 1989; Hansel et al., 1992; Opper, 1994; Kabashima, 1994; Simonetti & Caticha, 1996). In our work, we introduce PolyNets, a variant of parity machines in which the output is real-valued rather than $\pm 1$; and we theoretically analyze disjoint-Polynets, which are the real-output analogue of the oft-considered parity machines with tree architecture. While much of the statistical mechanics of ML literature focuses on an idealized training limit in which the weights reach a Gibbs distribution equilibrium, there is a strand of the literature that aims to characterize the trajectory of SGD training in the high-dimensional limit with constant-sized sets of ordinary differential equations (Saad & Solla, 1995b,a; Goldt et al., 2019). These papers discuss cases, including problems that share aspects with 2-sparse parities Refinetti et al. (2021), where the network gets stuck in (and then escapes from) a plateau of suboptimal generalization error. Recently, Arous et al. (2021) studied (for rank-one parameter estimation problems) the relative amount of time spent by SGD in an initial high-error "search" phase versus a final "descent" phase, which is reminiscent of the framing of Theorem 5.7. However, to our knowledge prior work has not shown $k$-sparse parities can be learned with a number of iterations that nearly matches known lower bounds, nor has it specifically studied phase transitions in $k$-sparse parity learning during gradient descent.

Learning parities with the NTK. Another relevant line of work studies learning the parity problem using the neural tangent kernel (NTK) (Jacot et al., 2018). Namely, in some settings, when the network's weights stay close to their initialization throughout the training, SGD converges to a solution that is given by a linear function over the initial features of the NTK. As shown in Theorem 5.6, learning parities over a fixed set of features requires the size of the model to be $\Omega(n^k)$. In contrast, the model size (number of hidden neurons) considered in this paper does not depend at all on the input dimension $n$. Nevertheless, the NTK analysis does give better sample complexity guarantees than the ones presented in this work, with a somewhat more natural version of SGD. For example, the work of Ji & Telgarsky (2019) demonstrates learning 2-sparse parities using NTK analysis with a sample complexity of $O(n^2)$, which matches the sample complexity lower bound for learning this problem with NTK (see Wei et al. (2019a)). Concurrent work by Telgarsky (2022) shows that this sample complexity can be improved to $O(n)$ once the optimization leaves the NTK regime. However, this analysis is given for networks of size $O(n^n)$, much larger than the networks considered in this paper. We refer the reader to Table 1 in Telgarsky (2022) for a complete comparison of the sample-complexity, run-time and model-size bounds achieved by different works studying 2-sparse parities.

## B.2 Proofs

### B.2.1 Global convergence for SGD on MLPs

For some even number $r$, consider a ReLU MLP of size $r$:

$$f(x;\theta) = \sum_{i=1}^{r} u_i \sigma(w_i^\top x + b_i)$$

Where $\sigma$ is the ReLU activation $\sigma(x) = \max\{x, 0\}$, $w_1, \ldots, w_r \in \mathbb{R}^{r\times n}$, $b \in \mathbb{R}^r$, $u \in \mathbb{R}^r$ and we denote the set of parameters by $\theta = \{w_1, \ldots, w_r, b, u\}$. We denote by $u_i^{(t)}, w_i^{(t)}, b^{(t)}$ and $\theta_t$ the value of the relevant parameters at iteration $t$ of gradient-descent. For brevity, we sometimes denote $w_i = w_i^{(0)}, b_i = b_i^{(0)}, u_i = u_i^{(0)}$. W.l.o.g., we assume that $S = [k]$, and so $y = \chi_{[k]}(x) = \prod_{i=1}^k x_i$. Indeed, since the weights' initialization we consider is permutation symmetric, this does not limit the generality of the results. We take $\ell$ to be the **hinge loss**: $\ell(y, \widehat{y}) = \max\{1 - y\widehat{y}, 0\}$. We use the following unbiased initialization:

- For all $1 \le i \le r/2$, randomly initialize

$$w_i^{(0)} \sim \text{Unif}\left(\{\pm 1\}^n\right), u_i^{(0)} \sim \text{Unif}\left(\{\pm 1\}\right), b_i^{(0)} \sim \text{Unif}(\{-1 + 1/k, \ldots, 1 - 1/k\})$$

- For all $r/2 < i \le r$, initialize

$$w_i^{(0)} = w_{i-r/2}^{(0)}, b_i^{(0)} = b_{i-r/2}^{(0)}, u_i^{(0)} = -u_{i-r/2}^{(0)}$$

We start by computing the population gradient at initialization. Using the fact that $\ell'(0, y) = -y$ we get the following:

$$\mathbb{E}\left[\nabla_{w_{i,j}} \ell(f(x; \theta_0), y)\right] = \mathbb{E}\left[-y \nabla_{w_{i,j}} f(x; \theta_0)\right] \tag{B.4}$$

$$= \mathbb{E}\left[-y u_i 1\left\{w_i \cdot x + b_i > 0\right\} x_j\right].$$

For $j \in [k]$ we have:

$$\mathbb{E}\left[\nabla_{w_{i,j}} \ell(f(x; \theta_0), y)\right] = -u_i \, \mathbb{E}\left[\left(\prod_{j' \in [k]\setminus\{j\}} x_{j'}\right) 1\left\{w_i \cdot x + b_i > 0\right\}\right]$$

For $j \notin [k]$ we have:

$$\mathbb{E}\left[\nabla_{w_{i,j}} \ell(f(x;\theta_0),y)\right] = -u_i \,\mathbb{E}\left[\left(\prod_{j' \in [k]\cup\{j\}} x_{j'}\right) 1\{w_i \cdot x + b_i > 0\}\right]$$

Finally, we have:

$$\mathbb{E}\left[\nabla_{b_i} \ell(f(x;\theta_0),y)\right] = -u_i \,\mathbb{E}\left[\left(\prod_{j' \in [k]} x_{j'}\right) 1\{w_i \cdot x + b_i > 0\}\right]$$

Denote

$$g_{i,j} = \mathbb{E}\left[\nabla_{w_{i,j}} \ell(f(x;\theta_0),y)\right],\ \gamma_i = \mathbb{E}\left[\nabla_{b_i} \ell(f(x;\theta_0),y)\right]$$

For some function $f$ and some subset $S \subseteq [n]$, denote $\widehat{f}(S) = \mathbb{E}[f(x)\chi_S(x)]$.

Denote $\mathrm{LTF}_{w,b}(x) = 1\{w \cdot x + b > 0\}$ and let $\mathrm{Maj}(x) = \mathrm{sign}\left(\sum_{i=1}^n x_i\right)$ and observe that, if $|b| < 1$,

$$\mathrm{LTF}_{w,b}(x) = \frac{1}{2} + \frac{1}{2}\mathrm{Maj}(w \odot x)$$

where $w \odot x = (w_1 x_1, \ldots, w_n x_n) \in \{\pm 1\}$. Since $\{\chi_S\}_{S\subseteq[n]}$ is a Fourier Basis, we can write $\mathrm{Maj} = \sum_{S\subseteq[n]} \widehat{\mathrm{Maj}}(S)\chi_S$ and therefore:

$$\begin{aligned}\mathrm{LTF}_{w,b}(x) &= \frac{1}{2} + \frac{1}{2}\mathrm{Maj}(w \odot x) = \frac{1}{2} + \frac{1}{2}\sum_{S\subseteq[n]} \widehat{\mathrm{Maj}}(S)\chi_S(w \odot x) \\ &= \frac{1}{2} + \frac{1}{2}\sum_{S\subseteq[n]} \widehat{\mathrm{Maj}}(S)\chi_S(w)\chi_S(x)\end{aligned}$$

So, for every $S \subseteq [n]$ with $|S| \geq 1$ we have $\widehat{\mathrm{LTF}}_{w,b}(S) = \frac{1}{2}\widehat{\mathrm{Maj}}(S)\chi_S(w)$ and so $|\widehat{\mathrm{LTF}}_{w,b}(S)| = \frac{1}{2}|\widehat{\mathrm{Maj}}(S)|$.

**Lemma B.1** (O'Donnell (2014), Section 5.3). *Fix some $k$, and assume that $n \geq 2k^2$. Then, for every*

$S \subseteq [n]$ *s.t.* $|S| = k$ *it holds that:*

- *If* $k$ *is even:* $\widehat{\mathrm{Maj}}(S) = 0$.
- *If* $k$ *is odd:*

$$c_1^2 k^{-2/3} \leq \binom{n}{k} \widehat{\mathrm{Maj}}(S)^2 \leq c_2^2 k^{-2/3}$$

*for some universal constants* $c_1, c_2$. *More precisely,*

$$\widehat{\mathrm{Maj}}(S) = (-1)^{\frac{k-1}{2}} \frac{\binom{\frac{n-1}{2}}{\frac{k-1}{2}}}{\binom{n-1}{k-1}} \cdot 2^{-(n-1)} \binom{n-1}{\frac{n-1}{2}}$$

Observe that for all $S \subseteq [n]$ s.t. $|S| = k$ it holds that $\widehat{\mathrm{Maj}}(S) = \widehat{\mathrm{Maj}}([k])$ and denote $\xi_k := \widehat{\mathrm{Maj}}([k])$.

Therefore, by the previous lemma we get that for every even $k$ the following holds:

$$\frac{\xi_{k+1}^2}{\xi_{k-1}^2} \leq \frac{c_2^2 (k-1)^{2/3} \binom{n}{k-1}}{c_1^2 (k+1)^{2/3} \binom{n}{k+1}} \leq \frac{c_2}{c_1} \frac{(k+1)k}{(n-k)(n-k+1)} \leq \frac{8c_2}{c_1} \frac{k^2}{n^2} \tag{B.5}$$

Also, observe that

**Lemma B.2** (Fourier gap for majority)**.** *Fix some* $k$ *and assume that* $n \geq 4k$. *Then, Majority has a* $\gamma_k$*-Fourier gap at* $S$ *of size* $k$ *with* $\gamma_k = 0.03(n-1)^{-\frac{k-1}{2}}$.

*Proof.* First we establish a simple relationship between $|\xi_{k-1}|$ and $|\xi_{k+1}|$.

$$\begin{aligned}
|\xi_{k-1}| &= \frac{\binom{\frac{n-1}{2}}{\frac{k}{2}-1}}{\binom{n-1}{k-2}} 2^{-(n-1)} \binom{n-1}{\frac{n-1}{2}} \\
&= \frac{n-k}{k-1} \cdot \frac{\binom{\frac{n-1}{2}}{\frac{k}{2}}}{\binom{n-1}{k}} \cdot 2^{-(n-1)} \binom{n-1}{\frac{n-1}{2}} \\
&= \frac{n-k}{k-1} \cdot |\xi_{k+1}|.
\end{aligned}$$

Here, the first equation follows from Lemma B.1, and the second equation follows by simple algebra using the following equality: $\binom{m}{r} = \frac{m-r+1}{r}\binom{m}{r-1}$.

Now, we can bound the difference,

$$\begin{aligned}
|\xi_{k-1}| - |\xi_{k+1}| &= \frac{n-2k+1}{k-1} \cdot |\xi_{k+1}| \\
&= \frac{n-2k+1}{k-1} \cdot \frac{\binom{\frac{n-1}{2}}{\frac{k}{2}}}{\binom{n-1}{k}} \cdot 2^{-(n-1)} \binom{n-1}{\frac{n-1}{2}} \\
&\geq \frac{n-2k+1}{k-1} \cdot \left(\frac{n-1}{k}\right)^{-k/2} \cdot e^{-k} \cdot \frac{2\sqrt{2\pi}}{e^2\sqrt{n-1}} \\
&\geq 0.03(n-1)^{-\frac{k-1}{2}}.
\end{aligned}$$

Here, the first equality holds from above, the second by Lemma B.1, the third inequality holds from standard approximations of the binomial coefficients, and the last inequality follows from the following inequalities: $n - 2k + 1 \geq (n-1)/2$ (by assumption on $n$) and $\frac{\sqrt{2\pi}k^{k/2}}{(k-1)e^{k+2}} > 0.03$ (by standard calculus). This gives us the desired result.

□

**Lemma B.3.** *Assume that $k$ is even and that $n \geq 2(k+1)^2$. Then, the following hold:*

1. *If $j \in \{1, \ldots, k\}$ then:*

$$g_{i,j} = -\frac{1}{2} u_i \xi_{k-1} \cdot \chi_{[k]\setminus\{j\}}(w_i)$$

2. *If $j \in \{k+1, \ldots, n\}$ then:*

$$g_{i,j} = -\frac{1}{2} u_i \xi_{k+1} \cdot \chi_{[k]\cup\{j\}}(w_i)$$

3. $\gamma_i = 0$.

*Proof.* If $j \in [k]$ then:

$$g_{i,j} = -u_i \mathbb{E}\left[\chi_{[k]\setminus\{j\}}(x)\mathrm{LTF}_{w_i,b_i}(x)\right] = -u_i\widehat{\mathrm{LTF}}_{w_i,b_i}([k]\setminus\{j\})$$
$$= -\frac{1}{2}u_i\widehat{\mathrm{Maj}}([k]\setminus\{j\})\chi_{[k]\setminus\{j\}}(w_i) = -\frac{1}{2}u_i\xi_{k-1}\cdot\chi_{[k]\setminus\{j\}}(w_i)$$

Similarly, if $j \notin [k]$ we have:

$$g_{i,j} = -u_i \mathbb{E}\left[\chi_{[k]\cup\{j\}}(x)\mathrm{LTF}_{w_i,b_i}(x)\right] = -u_i\widehat{\mathrm{LTF}}_{w_i,b_i}([k]\cup\{j\})$$
$$= -\frac{1}{2}u_i\widehat{\mathrm{Maj}}([k]\cup\{j\})\chi_{[k]\cup\{j\}}(w_i) = -\frac{1}{2}u_i\xi_{k+1}\cdot\chi_{[k]\cup\{j\}}(w_i)$$

Finally, we have:

$$\gamma_i = -u_i\widehat{\mathrm{LTF}}_{w_i,b_i}([k]) = -u_i\widehat{\mathrm{Maj}}([k]) = 0$$

□

**Lemma B.4.** *Let $\tau > 0$ be some tolerance parameter, fix $\varepsilon \in (0,1)$ and let $\eta = \frac{1}{k|\xi_{k-1}|}$. Assume that $k$ is an even number. Fix some $w_1,\ldots,w_k \in \{\pm 1\}^n$, $b_1,\ldots,b_r \in (-1,1)$ and $u_1,\ldots,u_k \in \{\pm 1\}$. Let $\widehat{w}_i = -\eta\widehat{g}_i$ and $\widehat{b}_i = b_i - \eta\widehat{\gamma}_i$ s.t. $\|\widehat{g}_i - g_i\|_\infty \le \tau$ and $\|\widehat{\gamma}_i - \gamma_i\| \le \tau$. Assume the following holds:*

- *For all $i,j \in [k]$ it holds that $w_{i,j} = u_i \cdot \operatorname{sign}\xi_{k-1}$.*
- *$b_i = -\frac{1}{2} + \frac{i+1}{k}$*

*Then, if $\tau \le \frac{|\xi_{k-1}|}{16k\sqrt{2n\log(2k/\varepsilon)}}$ and $n \ge \frac{2^{11}k^4c_2^2\log(2k/\varepsilon)}{c_1^2}$ there exists some $\widehat{u} \in \mathbb{R}^k$ with $\|\widehat{u}\|_\infty \le 8k$ s.t. $f(x) = \sum_{i=1}^k \widehat{u}_i\sigma(\widehat{w}_i\cdot x + \widehat{b}_i)$ satisfies*

$$\mathop{\mathbb{E}}_x[\ell(f(x),\chi_{[k]}(x))] \le 16\varepsilon k^2 n$$

*Additionally, for all $i$ and all $x$ it holds that $|\sigma(\widehat{w}_i\cdot x + \widehat{b}_i)| \le n+1$.*

*Proof.* We start with the following claim.

**Claim 1**: For all $i$ and for all $j \in [k]$ it holds that $\left|\widehat{w}_{i,j} - \frac{1}{2k}\right| \leq \frac{\tau}{|\xi_{k-1}|}$.

**Proof**: First, observe that by the assumption it holds that for all $i, j \in [k]$,

$$u_i \cdot \chi_{[k]\setminus\{j\}}(w_i) = u_i \cdot \prod_{j' \in [k]\setminus\{j\}}^{k} w_{i,j} = u_i \cdot (u_i \cdot \operatorname{sign} \xi_{k-1})^{k-1} = \operatorname{sign} \xi_{k-1}$$

Now, from Lemma B.3 we have:

$$g_{i,j} = -\frac{1}{2} u_i \xi_{k-1} \cdot \chi_{[k]\setminus\{j\}}(w_i) = -\frac{1}{2}|\xi_{k-1}|$$

and so

$$\left|w_{i,j} - \frac{1}{2k}\right| = \left|-\eta\widehat{g}_{i,j} - \frac{1}{2k}\right| = \frac{1}{k}\left|-k\eta\widehat{g}_{i,j} + \frac{g_{i,j}}{|\xi_{k-1}|}\right| = \frac{1}{k|\xi_{k-1}|}\left|g_{i,j} - \widehat{g}_{i,j}\right| \leq \frac{\tau}{|\xi_{k-1}|}$$

**Claim 2**: For all $i$ and for all $j > k$ it holds that $|\widehat{w}_{i,j}| \leq \frac{|\xi_{k+1}|+2\tau}{2k|\xi_{k-1}|}$

**Proof**: Using Lemma B.3 we have:

$$|\widehat{w}_{i,j}| = \eta|\widehat{g}_{i,j}| \leq \eta(|g_{i,j}| + |g_{i,j} - \widehat{g}_{i,j}|) \leq \eta\left(\frac{|\xi_{k+1}|}{2} + \tau\right) = \frac{|\xi_{k+1}| + 2\tau}{2k|\xi_{k-1}|}$$

**Claim 3**: For all $i$ it holds that $|\widehat{b}_i - b_i| \leq \frac{\tau}{k|\xi_{k-1}|}$

**Proof**: Using Lemma B.3 we have:

$$|\widehat{b}_i - b_i| = \eta|\widehat{g}_{i,0}| = \eta|g_{i,j} - \widehat{g}_{i,j}| \leq \eta\tau = \frac{\tau}{k|\xi_{k-1}|}$$

**Claim 4**: Fix $\vartheta > 0$. Let $h_i$ be a function s.t. $h_i(x) = \sigma(\frac{1}{2k}\sum_{j=1}^{k} x_j + b_i)$ and $\widehat{h}_i$ a function s.t. $\widehat{h}_i(x) = \sigma(\widehat{w}_{i,j} \cdot x + \widehat{b}_i)$. Then, if $\tau \leq \frac{\vartheta}{2}\frac{k|\xi_{k-1}|}{\sqrt{2n\log(2k/\varepsilon)}}$ and $n \geq \frac{32c_2^2\log(2k/\varepsilon)}{c_1^2\vartheta^2}$, the following holds:

1. $\mathbb{P}_{x\sim\{\pm 1\}^n}\left[|h_i(x)-\widehat{h}_i(x)|\geq \delta\right]\leq \frac{\varepsilon}{k}$
2. $|\widehat{h}_i(x)|\leq n+1$

**Proof**: Let $x\sim\{\pm 1\}^n$, and denote $\tilde{x}_j=\widehat{w}_{i,j}x_j$. Denote $\Delta=\frac{|\xi_{k+1}|+2\tau}{2k|\xi_{k-1}|}$. So, for $j>k$, $\tilde{x}_j$ is a random variable satisfying $|\tilde{x}_j|\leq\Delta$. Furthermore, it holds that $\mathbb{E}\left[\sum_{j>k}\tilde{x}_j\right]=0$. Therefore, from Hoeffding's inequality:

$$\mathbb{P}\left[\left|\sum_{j>k}\tilde{x}_j\right|\geq\Delta\sqrt{\frac{n\log(2k/\varepsilon)}{2}}\right]\leq 2\exp\left(-\frac{2\left(\Delta\sqrt{\frac{n\log(2k/\varepsilon)}{2}}\right)^2}{n\Delta^2}\right)\leq\frac{\varepsilon}{k}$$

Now, fix some $x$ s.t. $\left|\sum_{j>k}\tilde{x}_j\right|\leq\Delta\sqrt{\frac{n\log(2k/\varepsilon)}{2}}$. In this case we have:

$$\begin{aligned}
|h_i(x)-\widehat{h}_i(x)| &\leq \left|\frac{1}{2k}\sum_{j=1}^{k}x_j+b_i-\widehat{w}_i\cdot x+\widehat{b}_i\right| \\
&\leq \sum_{j=1}^{k}\left|\frac{1}{2k}x_j-\widehat{w}_{i,j}x_j\right|+\left|\sum_{j>k}\widehat{w}_{i,j}x_j\right|+\left|b_i-\widehat{b}_i\right| \\
&\leq \frac{k\tau}{|\xi_{k-1}|}+\Delta\sqrt{\frac{n\log(2k/\varepsilon)}{2}}+\frac{\tau}{k|\xi_{k-1}|} \\
&= \frac{\tau(k^2+1)}{k|\xi_{k-1}|}+\frac{|\xi_{k+1}|+2\tau}{2k|\xi_{k-1}|}\cdot\sqrt{\frac{n\log(2k/\varepsilon)}{2}} \\
&= \frac{\tau\left(\sqrt{2}(k^2+1)+\sqrt{n\log(2k/\varepsilon)}\right)}{\sqrt{2}k|\xi_{k-1}|}+\frac{|\xi_{k+1}|\sqrt{n\log(2k/\varepsilon)}}{2\sqrt{2}k|\xi_{k-1}|} \\
&\leq \frac{\tau\sqrt{2n\log(2k/\varepsilon)}}{k|\xi_{k-1}|}+\frac{2\sqrt{2}c_2\sqrt{\log(2k/\varepsilon)}}{c_1\sqrt{n}}
\end{aligned}$$

where in the last inequality we use Eq. (B.5). So, choosing $\tau\leq\frac{\delta}{2}\frac{k|\xi_{k-1}|}{\sqrt{2n\log(2k/\varepsilon)}}$ and $n\geq\frac{32c_2^2\log(2k/\varepsilon)}{c_1^2\delta^2}$ gives the required.

**Claim 5:** Let $h_1,\ldots,h_k$ be the functions defined in the previous claim. Then, there exists weights

$u^*$ with $\|u^*\|_\infty \leq 8k$ s.t. for $f^*(x) = \sum_{i=1}^{k} u_i^* h_i(x)$ it holds that $f^*(x) = 2\chi_{[k]}(x)$ for all $x \in \{\pm 1\}^n$.

**Proof:** For $i \leq k-2$ define $u_i^* = 8k(-1)^{i+1}$ and $u_{k-1}^* = 6k$, $u_k^* = -2k$.

**Proof of Lemma B.4:** Choose $\widehat{u} = u^*$. Using Claim 4 and the union bound, w.p. $1-\varepsilon$ over $x \sim \{\pm 1\}^n$ it holds that for all $i \in [k]$, $|h_i(x) - \widehat{h}_i(x)| \leq \delta$. Therefore, w.p. $\geq 1-\varepsilon$

$$|f(x) - f^*(x)| = \left|\sum_{i=1}^{k} u_i^* (h_i(x) - \widehat{h}_i(x))\right| \leq \sum_{i=1}^{k} |u_i^*| \left|h_i(x) - \widehat{h}_i(x)\right| \leq 8k^2\delta$$

so, choosing $\delta = \frac{1}{8k^2}$ we get that, w.p. at least $1-\varepsilon$ over the choice of $x$ it holds that $f(x)\chi_{[k]}(x) \geq 1$. Additionally, for every $x$ it holds that

$$|f(x)| \leq \sum_{i=1}^{k} |u_i^*| \left|\widehat{h}_i(x)\right| \leq 8k^2(n+1) \leq 16k^2 n$$

Therefore, we get:

$$\begin{aligned}\mathop{\mathbb{E}}_{x}\left[\ell(f(x), \chi_{[k]}(x))\right] &\leq \varepsilon \mathop{\mathbb{E}}_{x}\left[\ell(f(x), \chi_{[k]}(x)) | f(x)\chi_{[k]}(x) < 1\right] \\ &\leq \varepsilon \mathop{\mathbb{E}}_{x}\left[|f(x)| | f(x)\chi_{[k]}(x) < 1\right] \leq 16\varepsilon k^2 n\end{aligned}$$

□

**Lemma B.5.** *Assume we randomly initialize an MLP using the unbiased initialization defined previously. Consider the following update:*

$$w_i^{(1)} = (1-\lambda_0) w_i^{(1)} - \eta_0 \widehat{g}_i,\ b_i^{(1)} = b_i^{(1)} - \eta_0 \widehat{\gamma}_i$$

*where*

$$\widehat{g}_i = \frac{1}{B}\sum_{l=1}^{B} \nabla_{w_i}\ell(f(x_{0,l};\theta_0), y_{0,l}),\ \widehat{\gamma}_i = \frac{1}{B}\sum_{l=1}^{B} \nabla_{b_i}\ell(f(x_{0,l};\theta_0), y_{0,l})$$

*Let $k$ be even number. Then, for every $\varepsilon, \delta \in (0, 1/2)$, denoting $\tau = \frac{|\xi_{k-1}|}{16k\sqrt{2n\log(2k/\varepsilon)}}$, if $\eta = \frac{1}{k|\xi_{k-1}|}$, $\lambda_0 = 1, r \geq k \cdot 2^k \log(k/\delta), B \geq \frac{2}{\tau^2}\log(4nr/\delta)$ and $n \geq \frac{2^{11}k^4c_2^2\log(2k/\varepsilon)}{c_1^2}$, w.p. at least $1-2\delta$ over the initialization and the sample, there exists $\widehat{u} \in \mathbb{R}^r$ with $\|\widehat{u}\|_\infty \leq 8k$ and $\|\widehat{u}\|_2 \leq 8k\sqrt{k}$ s.t. $f(x) = \sum_{i=1}^{r} \widehat{u}_i \sigma\left(w_i^{(1)} \cdot x + b_i^{(1)}\right)$ satisfies*

$$\mathop{\mathbb{E}}_{x}[\ell(f(x), \chi_{[n]}(x))] \leq 16\varepsilon k^2 n$$

*Additionally, it holds that $\|\sigma(W^{(1)} \cdot x + b^{(1)})\|_\infty \leq n+1$.*

*Proof.* Note that by the choice of initialization it holds that $f(x; W^{(0)}) = 0$, and by the assumption on the loss function $\ell'(f(x; W^{(0)}), y) = -y$. Therefore, we get that $\mathbb{E}\left[\nabla_{w_i}\ell(f(x; W^{(0)}), y)\right] = g_i$ and $\mathbb{E}\left[\nabla_{b_i}\ell(f(x; W^{(0)}), y)\right] = \gamma_i$.

**Claim:** with probability at least $1-\delta$,

$$\text{for all } i,j: \ \left\|\widehat{g}_i - \mathbb{E}\left[\nabla_{w_i}\ell(f(x; W^{(0)}), y)\right]\right\|_\infty \leq \tau \text{ and } \left|\widehat{\gamma}_i - \mathbb{E}\left[\nabla_{b_i}\ell(f(x; W^{(0)}), y)\right]\right| \leq \tau \tag{B.6}$$

**Proof:** Fix some $i, j$ and note that by Hoeffding's inequality,

$$\Pr\left[|\widehat{g}_{i,j} - \mathbb{E}\widehat{g}_{i,j}| \geq \tau\right] \leq 2\exp\left(-B\tau^2/2\right) \leq \frac{\delta}{nr+r}$$

and similarly we get $\Pr\left[|\widehat{\gamma}_i - \mathbb{E}\widehat{\gamma}_i| \geq \tau\right] \leq \frac{\delta}{nr+r}$. The required follows from the union bound.

Now, assume that Eq. (B.6) holds. For some random $w_i \sim \{\pm 1\}^n$, the probability that $w_{i,j} = w_{i,j'}$ for all $j, j' \in [k]$ is $2^{-k+1}$. Additionally, for some fixed $i' \in [k]$, the probability that $b_i = -\frac{1}{2} + \frac{i'}{k}$ is $\frac{1}{2k}$. Therefore, for some fixed $i \in [r/2]$ and $i' \in [k]$, with probability $\frac{1}{k\cdot 2^{k-1}}$, $b_i = b_{i+r/2} =$

$-\frac{1}{2}+\frac{i'}{k}$ and either $w_{i,j}=u_i \operatorname{sign} \xi_{k-1}$ or $w_{i+r/2,j}=u_{i+r/2,j} \operatorname{sign} \xi_{k-1}$. Taking $r \geq k \cdot 2^k \log(k/\delta)$, we get that the probability that there is no $i \in [r/2]$ that satisfies the above condition (for fixed $i'$) is:

$$\left(1-\frac{1}{k2^{k-1}}\right)^{r/2} \leq \exp\left(-\frac{r}{2k2^{k-1}}\right) \leq \frac{\delta}{k}$$

Using the union bound, with probability at least $1-\delta$, there exists a set of $k$ neurons satisfying the conditions of Lemma B.4, and therefore the required follows from the Lemma. □

We use the following well-known result on convergence of SGD (see for example Shalev-Shwartz & Ben-David (2014)):

**Theorem B.6.** *Let $M, \rho > 0$. Fix $T$ and let $\eta = \frac{M}{\rho\sqrt{T}}$ Let $F$ be a convex function and $u^* \in \arg\min_{\|u\|_2 \leq M} f(u)$. Let $u^{(0)} = 0$ and for every $t$, let $v_t$ be some random variable s.t. $\mathbb{E}\left[v_t | u^{(t)}\right] = \nabla_{u^{(t)}} F(u^{(t)})$ and let $u^{(t+1)} = u^{(t)} - \eta v^{(t)}$. Assume that $\|v_t\|_2 \leq \rho$ w.p. 1. Then,*

$$\frac{1}{T}\sum_{t=1}^{T} F(u^{(t)}) \leq F(u^*) + \frac{M\rho}{\sqrt{T}}$$

We prove the following theorem:

*Theorem 5.5* (SGD on MLPs learns sparse parities; full statement). Let $k$ be an even number. Assume we randomly initialize an MLP using the unbiased initialization defined previously. Fix $\varepsilon \in (0, 1/2)$ and let $T \geq \frac{2^9 k^3 r n^2}{\varepsilon^2}$, $r \geq k \cdot 2^k \log(8k/\varepsilon)$, $B \geq c_1^{-1} 2^8 k^{7/6} n \binom{n}{k-1} \log(128k^3 n/\varepsilon) \log(32nr/\varepsilon)$, $n \geq \frac{2^{11} k^4 c_2^2 \log(128k^3 n/\varepsilon)}{c_1^2}$. Choose the following learning rate and weight decay schedule:

- At the first step, use $\eta_0 = \frac{1}{k|\xi_{k-1}|}$, $\lambda_0 = 1$ for all weights.
- After the first step, use $\eta_t = 0$ for the first layers weights and biases and $\eta_t = \frac{4k^{1.5}}{n\sqrt{r(T-1)}}$ for the second layer weights, with $\lambda_t = 0$ for both layers.
- Bias terms are never regularized.

Then, the following holds, with expectation over the randomness of the initialization and the sampling of the batches:

$$\mathbb{E}\left[\min_{t\in[T]} \ell(f(x;\theta_t),y)\right] \le \varepsilon$$

*Proof.* Let $F(u) = \mathbb{E}_x\left[\ell(u^\top \sigma(W^{(1)}x + b^{(1)}),y)\right]$ and notice that $F$ is a convex function. For every $t$, denote

$$v_t = \frac{1}{B}\sum_{l=1}^{B} \nabla_{u^{(t)}} \ell(f(x_{t,l};\theta_t),y) = \frac{1}{B}\sum_{l=1}^{B} \nabla_{u^{(t)}} \ell\left(\left(u^{(t)}\right)^\top \sigma(W^{(1)}x_{l,t} + b^{(1)}), y_{l,t}\right)$$

where we use the fact that we don't update the weights of the first layer after the first step. From the above we get $\mathbb{E}[v_t|u^{(t)}] = \nabla_{u^{(t)}} F(u^{(t)})$.

Now, we will show that w.h.p. there exists $u^*$ with good loss. Let $\varepsilon' = \frac{\varepsilon}{64k^2n}$, $\delta' = \frac{\varepsilon}{8}$. Denote $\tau = \frac{|\xi_{k-1}|}{16k\sqrt{2n\log(128k^3n/\varepsilon)}} = \frac{|\xi_{k-1}|}{16k\sqrt{2n\log(2k/\varepsilon')}}$. Observe that $r \ge k\cdot 2^k\log(k/\delta')$, and using the fact that $|\xi_{k-1}| \ge c_1(k-1)^{-1/3}\binom{n}{k-1}^{-1}$ we get

$$B \ge \frac{2^8k^2\cdot n\log(128k^3n/\varepsilon)}{|\xi_{k-1}|}\log(32nr/\delta) = \frac{2}{\tau^2}\log(4nr/\delta')$$

and additionally $n \ge \frac{2^{11}k^4c_2^2\log(2k/\varepsilon')}{c_1^2}$.

From the above, applying Lemma B.5 with $\varepsilon', \delta'$ we get that w.p. $1-\varepsilon/4$ there exists $u^* \in \mathbb{R}^r$ with $\|u^*\|_2 \le 8k\sqrt{k}$ s.t. $F(u^*) \le \varepsilon/4$ and for all $i$ and all $x$ it holds that $\|\sigma(W^{(1)}\cdot x + b^{(1)})\|_\infty \le n+1$. Using this, we get:

$$\|v_t\|_2 \le \frac{1}{B}\sum_{l=1}^{B} \|\sigma(W^{(1)}x_{l,t} + b^{(1)}), y_{l,t}\|_2 \le \sqrt{r}(n+1)$$

So, we can apply Theorem B.6 with $M = 8k\sqrt{k}$ and $\rho = 2\sqrt{rn}$ and get that, w.p. $1-\varepsilon/4$ over the

initialization and the first step, it holds that

$$\mathop{\mathbb{E}}_{\text{steps 2...T}}\left[\min_{t\in\{2,\ldots,T\}}\ell(f(x;\theta_t),y)\right] \le \mathbb{E}\left[\frac{1}{T-1}\sum_{t=2}^{T}\ell(f(x;\theta_t),y)\right]$$
$$= \mathbb{E}\left[\frac{1}{T-1}\sum_{t=2}^{T}F(u^{(t)})\right] \le \varepsilon/4 + \frac{16k^{1.5}\sqrt{rn}}{\sqrt{T-1}} \le \varepsilon/2.$$

Now, that after the first step we have $u^{(1)} = 0$ and therefore $\ell(f(x;\theta_1),y) = 1$, and so we always have $\min_{t\in[T]}\ell(f(x;\theta_t),y) \le 1$. Therefore, taking expecation over all steps we get:

$$\mathop{\mathbb{E}}_{\text{steps 1...T}}\left[\min_{t\in[T]}\ell(f(x;\theta_t),y)\right] \le \varepsilon/2 + \varepsilon/2 = \varepsilon.$$

The simplified form $B = \Omega(n^k\log(n/\varepsilon))$ in the main paper comes from the fact that $\binom{n}{k-1} \le n^{k-1}/(k-1)!$. This $1/(k-1)!$ factor dominates the other poly$(k)$ factors. □

### B.2.2 Recoverability of the parity indices from Fourier gaps

Given a network architecture where some neuron has a $\gamma$-Fourier gap with respect to the target subset $S$, we quantify how the indices in $S$ can be determined by observing an estimate of the population gradient for a general activation function $\sigma$ and $w_t$:

**Proposition B.7** (Fourier gap implies feature recoverability)**.** *For an activation function $\sigma:\mathbb{R}\to\mathbb{R}$, let $f(x;w) = \sigma(w^\top x)$ be the corresponding* 1*-neuron predictor. Let $\mathcal{D}_S$ be an $(n,k)$-sparse parity distribution. Let $g(w)$ be an estimate[*] for the neuron's population gradient of the correlation loss $\ell$:*

$$||g(w) - \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}_S}[\nabla_w\ell(y,f(x;w))]\,||_\infty < \gamma/2.$$

[*]For $O(1)$-bounded stochastic gradient estimators, $O\left(\frac{\log n}{\gamma^2}\right)$ samples suffice to obtain such an estimate.

*Then, for every $w$ such that $\sigma'(w^\top x)$ has a $\gamma$-Fourier gap at $S$, the $k$ indices at which $g(w)$ has the largest absolute values are exactly the indices in $S$.*

*Proof.* Let $h(x) := \sigma'(w^\top x)$. We compute the population gradient, we we call $\bar{g}(w)$:

$$\begin{aligned}
[\bar{g}(w)]_i = \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}_S}[\nabla_{w_i}\ell(y,f(x;w))] &= -\mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}_S}\left[\sigma'(w^\top x)yx_i\right] \\
&= \begin{cases} \mathbb{E}\left[\sigma'(w^\top x)\prod_{j\in S\setminus\{i\}} x_i\right] & i \in S \\ \mathbb{E}\left[\sigma'(w^\top x)\prod_{j\in S\cup\{i\}} x_i\right] & i \notin S \end{cases} \\
&= \begin{cases} \widehat{h}(S\setminus\{i\}) & i \in S \\ \widehat{h}(S\cup\{i\}) \le \widehat{h}(S\cup\{i\}) - \gamma & i \notin S \end{cases},
\end{aligned}$$

where the inequality in the final $i \notin S$ case is due to the Fourier gap property. Then, it holds that for all $i \in S$ we have $|g_i| > \gamma/2$ and for all $i \notin S$ we have $|g_i| < \gamma/2$. Thus, the largest entries of the estimate $g(w)$ occur at the indices in $S$, as claimed. □

### B.2.3 Global convergence for disjoint-PolyNets

In this section we will develop theory for disjoint-PolyNets trained with correlation loss, as illustrated in Figure B.1. Section B.2.3 will consider optimization with gradient flow, and section B.2.4 will consider optimization with SGD at any batch size $B \ge 1$.

For any $n \ge 1$ and $1 \le k \le n$ such that $n' := n/k$ is an integer, let $P_1, \dots, P_{n'}$ denote (without loss of generality) the partition $P_i := \{n'(i-1)+1, \dots, n'\cdot i\}$. Then, the $(n,k)$-*disjoint-PolyNet* is the neural architecture, with trainable parameters are $\{w_i \in \mathbb{R}^{n'}\}_{i=1}^k$, which outputs

$$f(x; w_{1:k}) := \prod_{i=1}^{k} \langle w_i, x_{P_i}\rangle.$$

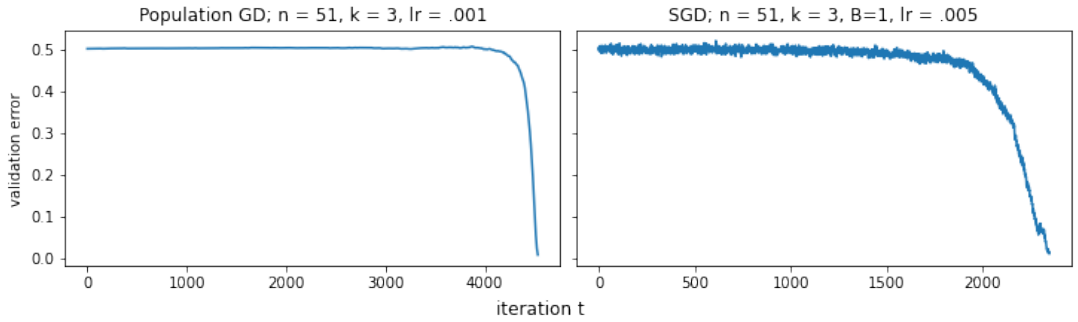



**Figure B.1:** Example training curves for the disjoint-PolyNet trained with correlation loss, which is the setting of Appendix B.2.3. The left plot shows validation error under population gradient descent with small step size, approximating the setting of Section B.2.3, and the right plot shows a run of SGD with batch size 1, as in Section B.2.4, though with a constant learning rate schedule. Initialization is i.i.d. standard Gaussian.

### Gradient flow analysis

For $i$, $\mathbb{E}\left[\nabla_{w_i}\ell(f(x; w_{1:k}), y)\right] = 0$, so the irrelevant weights remain fixed at initialization. For each $i \in [k]$, let $v_i^{(t)}$ be the relevant weight in the $k$th partition.

In gradient flow, the relevant weights evolve according to the following differential equations:

$$\forall i \in [n]: \quad \dot{v}_i = \prod_{j \neq i} v_j$$

**Lemma B.8.** *Suppose disjoint-PolyNet for $k > 2$ is initialized such that $\prod_i v_i(0) > 0$, and optimized with gradient flow. Let $\bar{v}_a := \frac{1}{k}\sum_{i=1}^k (v_i(0))^2$, and $\bar{v}_g := \left(\prod_{i=1}^k (v_i(0))^2\right)^{1/k}$. For any $b \geq 0$ and $i \in [k]$, let $T_i(b) := \arg\sup_{t \geq 0}(|v_i(t)| \leq b)$. Then*

$$T_i(b) \geq \frac{1}{k-2}\left(\bar{v}_a^{1-k/2} - (\bar{v}_a + b^2 - v_i(0)^2)^{1-k/2}\right).$$

*Let $T_i(\infty) := \arg\sup_{t \geq 0}(|v_i(t)| < \infty)$. Then*

$$T_i(\infty) - T_i(b) \leq \frac{1}{k-2}(\bar{v}_g + b^2 - v_i(0)^2)^{1-k/2}.$$

*Proof.* First, observe that the product of the relevant weights is non-decreasing during gradient flow.

$$\frac{d\left(\prod_{i=1}^{k} v_i(t)\right)}{dt} = \sum_{i=1}^{k} \frac{\partial\left(\prod_{i=1}^{k} v_i\right)}{\partial v_i} \cdot \dot{v}_i = \sum_{i=1}^{k} \left(\prod_{j\neq i} v_i\right)^2 > 0$$

Thus, $\prod_i v_i(0) > 0$ implies that $\prod_i v_i(t) > 0$ for all $t$.

Observe that

$$\frac{dv_i^2}{dt} = 2v_i\dot{v}_i = 2\prod_{j=1}^{k} v_j = 2\prod_{j=1}^{k} |v_j|.$$

This implies that for all $i, l \in [t]$,

$$\frac{dv_i^2}{dt} = \frac{dv_l^2}{dt}. \tag{B.7}$$

In other words, the squares of the relevant weights each follow the same trajectory, shifted according to their initializations. Let $q(t) := (v_i(t))^2 - (v_i(0))^2$, for any $i$. This quantity evolves as follows:

$$\dot{q} = 2\prod_{i=1}^{k} |v_i| = 2\left(\prod_{i=1}^{k} (q(t) + (v_i(0))^2)\right)^{1/2}$$

Since $q(t)$ is strictly increasing, its inverse $q^{-1}$ is well-defined, and we can use the inverse function theorem to characterize $q^{-1}$ for all $t \geq 0$:

$$q^{-1}(c) = \int_0^c \frac{1}{2}\left(\prod_{i=1}^{k} (\gamma + (v_i(0))^2)\right)^{-1/2} d\gamma.$$

We can upper- and lower-bound the integrand by applying Maclaurin's inequality (see page 52 in Hardy et al. (1952)):

$$\left(\gamma + \bar{v}_g\right)^k \leq \prod_{i=1}^{k} (\gamma + (v_i(0))^2) \leq \left(\gamma + \bar{v}_a\right)^k.$$

The amount of time it takes for $q$ to reach a value of $c$ is thus:

$$\int_0^c \frac{1}{2}\left(\prod_{i=1}^{k}(\gamma+(v_i(0))^2)\right)^{-1/2} d\gamma \geq \int_0^c \frac{1}{2}(\gamma+\bar{v}_a)^{-k/2}d\gamma = \frac{1}{k-2}\left(\bar{v}_a^{1-k/2}-(c+\bar{v}_a)^{1-k/2}\right).$$

Hence, for any $b \geq 0$, for each $i$, $|v_i(t)| \leq b$ as long as

$$t \leq \frac{1}{k-2}\left(\bar{v}_a^{1-k/2}-(\bar{v}_a+b^2-v_i(0)^2)^{1-k/2}\right).$$

Meanwhile, the amount of time after $q^{-1}(c)$ it takes for $q$ to explode to infinity is

$$\int_c^\infty \frac{1}{2}\left(\prod_{i=1}^{k}(\gamma+(v_i(0))^2)\right)^{-1/2} d\gamma \leq \int_c^\infty \frac{1}{2}(\gamma+\bar{v}_g)^{-k/2}d\gamma = \frac{1}{k-2}(c+\bar{v}_g)^{1-k/2}.$$

Substituting $c = b^2 - v_i(0)^2$, we obtain that the amount of time it takes for $|v_i|$ to grow from $b$ to $\infty$ is

$$\frac{1}{k-2}(\bar{v}_g+b^2-v_i(0)^2)^{1-k/2}.$$

We can upper- and lower-bound $\dot{q}$ by applying Maclaurin's inequality (see page 52 in Hardy et al. (1952)):

$$2\left(q(t)+\bar{v}_g\right)^{k/2} \leq \dot{q} \leq 2\left(q(t)+\bar{v}_a\right)^{k/2}.$$

When $k = 2$, solving the LHS and RHS differential inequalities yields:

$$\bar{v}_g(e^{2t}-1) \;\leq\; q(t) \;\leq\; \bar{v}_a(e^{2t}-1).$$

When $k > 2$, we obtain:

$$(\bar{v}_g^{-(k/2-1)}-(k-2)t)^{-\frac{1}{k/2-1}}-\bar{v}_g \;\leq\; q(t) \;\leq\; (\bar{v}_a^{-(k/2-1)}-(k-2)t)^{-\frac{1}{k/2-1}}-\bar{v}_a \tag{B.8}$$

From the lower bound on $q(t)$, we can infer that the relevant weights all explode to infinity by the following time:

$$t = \frac{1}{(k-2)} \bar{v}_g^{-(k/2-1)} = \frac{1}{(k-2)\left(\prod_i v_i(0)\right)^{1-2/k}}$$

From the upper bound, we can infer that for any $c > 0$, it is the case that $q(t) \leq c$ so long as

$$t \leq \frac{1}{k-2}\left(\bar{v}_a^{-(k/2-1)} - (\bar{v}_a + c)^{-(k/2-1)}\right).$$

Hence, for each $i$, $|v_i(t)| \leq b$ for all

$$t \leq \frac{1}{k-2}\left(\bar{v}_a^{-(k/2-1)} - (\bar{v}_a + b^2 - (v_i(0))^2)^{-(k/2-1)}\right).$$

□

Now we analyze the relationship between the relevant weights and the accuracy of the disjoint-PolyNet.

For $y \in \mathbb{R}$, let

$$\text{sign}(y) = \begin{cases} 1 & \text{if } y > 0 \\ 0 & \text{if } y = 0 \\ -1 & \text{if } y < 0 \end{cases}$$

Then define the error of $f$ with parameters $w_{1:k}$ as

$$\mathsf{err}(w_{1:k}) := \Pr_{x \sim \{\pm 1\}^n}[\text{sign}(f(x; w_{1:k})) \neq \chi_S(x)]$$

**Lemma B.9.** *Let $w$ be any setting of the weights of a disjoint-PolyNet such that $\prod_i v_i > 0$. For ease of*

*notation, let* $u_i := w_{i,2:n'}$ *be the irrelevant portion of* $w_i$*. There is a constant* $c$ *such that*

$$\frac{1}{2} - \frac{1}{2}\prod_{i=1}^{k}\left(\operatorname{erf}\left(\frac{|v_i|}{\|u_i\|_2\sqrt{2}}\right) + \frac{c\|u_i\|_\infty}{\|u_i\|_2^3}\right) \le \operatorname{err}(w_{1:k}) \le 2\sum_{i=1}^{k}\exp\left(-\frac{|v_i|^2}{\|u_i\|_2^2}\right)$$

*where* erf *is the Gauss error function* $\operatorname{erf}(y) := \frac{2}{\sqrt{\pi}}\int_0^y e^{-\tau^2}\,d\tau$*.*

*Proof.* Let $z_i = x_{(i-1)n'+1}$ be the $i$th relevant coordinate of $x$, and let $z_i^- = x_{(i-1)n'+2:in'}$ be the irrelevant coordinates in $P_i$.

Then we have that the error of $f$ with parameters $w_{1:k}$ is

$$\begin{aligned}
\operatorname{err}(w_{1:k}) &= \Pr_x\left[\operatorname{sign}\left(\prod_{i=1}^{k} w_i^\top x_{P_i}\right) \neq \chi_S(x)\right] \\
&= \Pr_x\left[\operatorname{sign}\left(\prod_{i=1}^{k} w_i^\top x_{P_i}\right) = -\chi_S(x)\right] + \Pr_x\left[\prod_{i=1}^{k} w_i^\top x_{P_i} = 0\right] \\
&= \Pr_x\left[\operatorname{sign}\left(\prod_{i=1}^{k} w_i^\top x_{P_i}\right) = -\chi_S(x)\right] + \Pr_x\left[\prod_{i=1}^{k} w_i^\top x_{P_i} = 0\right] \\
&= \Pr_x\left[\#\{i \in [k] : \operatorname{sign}(v_i z_i + u_i^\top z_i^-) = -z_i\} \text{ is odd}\right] + \Pr_x\left[\prod_{i=1}^{k} w_i^\top x_{P_i} = 0\right] \\
&= \Pr_{z_1^-,\ldots,z_k^-}\left[\#\{i \in [k] : u_i^\top z_i^- > v_i\} \text{ is odd}\right]/2 \\
&\quad + \Pr_{z_1^-,\ldots,z_k^-}\left[\#\{i \in [k] : u_i^\top z_i^- < -v_i\} \text{ is odd}\right]/2 + \Pr_x\left[\prod_{i=1}^{k} w_i^\top x_{P_i} = 0\right] && \text{(B.9)} \\
&= \Pr_{z_1^-,\ldots,z_k^-}\left[\#\{i \in [k] : u_i^\top z_i^- > v_i\} \text{ is odd}\right] + \Pr_x\left[\prod_{i=1}^{k} w_i^\top x_{P_i} = 0\right] && \text{(B.10)} \\
&= \Pr_{z_1^-,\ldots,z_k^-}\left[\#\{i \in [k] : u_i^\top z_i^- > |v_i|\} \text{ is odd}\right] + \Pr_x\left[\prod_{i=1}^{k} w_i^\top x_{P_i} = 0\right] && \text{(B.11)}
\end{aligned}$$

Line B.9 follows because $u_i^\top z_i^-$ and $z_i$ are independent. In line B.10 we use that $u_i^\top z_i^-$ is symmetric about 0. Finally, line B.11 uses the assumption that $\prod_{i=1}^{k} v_i > 0$.

We can bound $\Pr_x\left[\prod_{i=1}^k w_i^\top x_{P_i} = 0\right]$ using Hoeffding's inequality:

$$0 \le \Pr_x\left[\prod_{i=1}^k w_i^\top x_{P_i} = 0\right] \le \sum_{i=1}^k \Pr_{z_i}[u_i^\top z_i^- = v_i] \le \sum_{i=1}^k \Pr_{z_i}[u_i^\top z_i^- \ge |v_i|] \le \sum_{i=1}^k \exp\left(-\frac{|v_i|^2}{\|u_i\|_2^2}\right).$$

The indicator random variables $\mathbb{1}[u_i^\top z_i^- > |v_i|]$ are independent of each other, so the first term in line B.11 can be characterized using the distribution of the parity of a sum of independent Bernoulli random variables. Let $X_i \sim \mathrm{Ber}(p_i)$ for $i \in [k]$, and let $X = \sum_{i=1}^k X_i$. The generating function for $X$ is $f(z) = \prod_{i=1}^k ((1-p_i) + p_i z)$. The parity of $X$ then satisfies

$$\Pr[X \text{ is odd}] = \frac{f(1)}{2} - \frac{f(-1)}{2} = \frac{1}{2} - \frac{1}{2}\prod_{i=1}^k (1-2p_i).$$

First we will prove the upper bound on $\mathsf{err}$. First, observe that

$$\frac{1}{2} - \frac{1}{2}\prod_{i=1}^k (1-2p_i) \le \sum_{i=1}^k p_i.$$

Thus,

$$\begin{aligned}\Pr_{z_1^-,\ldots,z_k^-}\left[\#\{i \in [k] : u_i^\top z_i^- > |v_i|\} \text{ is odd}\right] &\le \sum_{i=1}^k \Pr_{z_i}[u_i^\top z_i^- > |v_i|] \\ &\le \sum_{i=1}^k \exp\left(-\frac{|v_i|^2}{\|u_i\|_2^2}\right)\end{aligned}$$

by Hoeffding's inequality, and we are done.

Now we will prove the lower bound on err. We have

$$\Pr_{z_1^-,\ldots,z_k^-}\left[\#\{i\in[k] : u_i^\top z_i^- > |v_i|\} \text{ is odd}\right] = \frac{1}{2} - \frac{1}{2}\prod_{i=1}^{k}(1-2\Pr_{z_i}[u_i^\top z_i^- > |v_i|])$$
$$= \frac{1}{2} - \frac{1}{2}\prod_{i=1}^{k}\Pr_{z_i}[|u_i^\top z_i^-| \le |v_i|]. \tag{B.12}$$

We can bound this expression using the Berry-Esseen theorem (Berry, 1941; Esseen, 1942). Let $\beta \sim N(0, \|u_i\|_2^2)$. Then, the Berry-Esseen theorem states that there is a constant $c$ (which in practice can be .56) such that for any $i \in [k]$,

$$\left|\Pr_{z_i}[|u_i^\top z_i^-| \le |v_i|] - \Pr[|\beta| \le |v_i|]\right| \le \frac{c\|u_i\|_\infty}{\|u_i\|_2^3}$$

and we can use the characterization

$$\Pr[|\beta| \le |v_i|] = \operatorname{erf}\left(\frac{|v_i|}{\|u_i\|_2\sqrt{2}}\right).$$

Plugging this into equation B.12, we obtain the lower bound on err. □

First, we'll apply Lemma B.8 and Lemma B.9 to the situation where the $w_i$'s have $\pm 1$ initialization. This generalizes Theorem 5.7.

**Corollary B.10.** *Suppose all the weights in a disjoint-Polynet are initialized randomly in* $\pm 1$ *and* $k \ge 3$.

*Let* $T(\alpha) := \arg\sup_{t\ge 0}(\mathsf{err}(w_{1:k}(t)) \ge \alpha$. *Then, for* $\gamma \in (0, 1/2)$, *if* $\prod_i v_i(0) > 0$ *(which happens w.p. 1/2),*

$$\frac{T(1/2-\gamma)}{T(0)} = 1 - O((n')^{1-k/2}\cdot\gamma^{2/k-1}).$$

*Thus, even for* $\gamma$ *arbitrarily close to* 0, *when the input is sufficiently long, the network spends almost all*

*of training with error above* $1/2 - \gamma$.

*Proof.* For $\pm 1$ initialization,

$$\frac{\|u_i\|_\infty}{\|u_i\|_2^3} = (n')^{-3/2}$$

so by Lemma B.9,

$$\begin{aligned}\mathsf{err}(w_{1:k}(t)) &\geq \frac{1}{2} - \frac{1}{2}\prod_{i=1}^{k}\left(\mathrm{erf}\left(\frac{|v_i(t)|}{\sqrt{2n'}}\right) + c(n')^{-3/2}\right) \\ &\geq \frac{1}{2} - \frac{1}{2}\prod_{i=1}^{k}\left(\sqrt{\frac{2}{\pi n'}}\cdot|v_i(t)| + c(n')^{-3/2}\right)\end{aligned}$$

for some $c > 0$.

By Lemma B.8, $|v_i(t)| \leq b$ for all $i$ whenever

$$t \leq \frac{1}{k-2}\left(1 - b^{2-k}\right).$$

Setting

$$b = \sqrt{\pi n'/2}\left(\gamma^{1/k} - c(n')^{-3/2}\right),$$

we obtain that $T(1/2-\gamma) = \frac{1}{k-2}(1 - O((n')^{1-k/2}\cdot\gamma^{2/k-1}))$.

Also by Lemma B.8, using the language of that lemma statement, for all $i$

$$T_i(\infty) - T_i(b) \leq \frac{1}{k-2}\cdot b^{2-k} = \frac{1}{k-2}\cdot O((n')^{1-k/2}\cdot\gamma^{2/k-1}).$$

Once all the relevant weights have exploded to infinity, the error of the network will have zero error, so the result follows. □

Now let us apply Lemma B.8 and Lemma B.9 to the situation where the $w_i$'s have standard normal initialization. Again we find that with high probability, the phase of learning with near-trivial

accuracy is much longer than the subsequent period until perfect accuracy, as illustrated by the left-hand plot in Figure B.1. This culminates in the full theorem statement regarding phase transitions in the loss:

*Theorem 5.7* (Loss plateau for gradient flow on disjoint-PolyNets; full statement). Suppose all the weights in a disjoint-PolyNet are initialized $\sim N(0,1)$, and $k \geq 3$. Then, conditioned on $\prod_i v_i(0) > 0$, with probability $1 - 1/\mathrm{poly}(n')$ over the randomness of the initialization, for $\gamma \in (0, 1/2)$,

$$\frac{T(1/2-\gamma)}{T(0)} = 1 - \tilde{O}((n')^{1-k/2} \cdot \gamma^{2/k-1}).$$

where $T(\cdot)$ is defined as in Corollary B.10.

*Proof.* By a standard application of the generic Chernoff bound (Wainwright, 2019), for each $i \in [k], j \in [n'-1]$, we have

$$\Pr[|u_{i,j}| > \tau] \leq 2e^{-\tau^2/2} \quad \text{for all } \tau \geq 0.$$

Applying the union bound, we obtain

$$\Pr[\exists i, j \text{ s.t. } |u_{i,j}| > \tau] \leq 2n'ke^{-\tau^2/2} \quad \text{for all } \tau \geq 0.$$

This implies that w.p. $\geq 1 - \varepsilon/2$,

$$\forall i: \quad \|u_i\|_\infty \leq \sqrt{2\log(4n'k/\varepsilon)}. \tag{B.13}$$

For each $i$, $\|u_i\|_2^2$ follows a chi-squared distribution with $n'-1$ degrees of freedom. By Laurent & Massart (2000), for any $\tau \geq 0$,

$$\|u_i\|_2^2 \in [(n'-1) - 2\sqrt{(n'-1)\tau}, (n'-1) + 2\sqrt{(n'-1)\tau} + 2\tau] \quad \text{w.p. } \geq 1 - 2e^{-\tau}.$$

Hence, w.p. $\geq 1 - \varepsilon/2$,

$$\forall i: \quad \|u_i\|_2 = \sqrt{n'} + O(\sqrt{\log(k/\varepsilon)}) \tag{B.14}$$

With probability $1 - \varepsilon$ both $\|u_i\|_\infty$ and $\|u_i\|_2$ are bounded as above, in which case we obtain that for $\varepsilon = 1/\mathrm{poly}(n', k)$, and for some $c_1, c_2 > 0$,

$$\frac{\|u_i\|_\infty}{\|u_i\|_2^3} \leq \frac{c_1\sqrt{\log(n'k)}}{\left(\sqrt{n'} + c_2\sqrt{\log(n'k)}\right)^3} \leq \tilde{O}\left((n')^{-3/2}\right).$$

Plugging this into Lemma B.9 gives us

$$\begin{aligned}
\mathsf{err}(w_{1:k}(t)) &= \frac{1}{2} - \frac{1}{2}\prod_{i=1}^{k}\left(\mathrm{erf}\left(\frac{|v_i(t)|}{\|u_i\|_2\sqrt{2}}\right) + O\left(\frac{\|u_i\|_\infty}{\|u_i\|_2^3}\right)\right) \\
&\geq \frac{1}{2} - \frac{1}{2}\prod_{i=1}^{k}\left(\tilde{O}\left(\frac{|v_i(t)|}{\sqrt{n'}}\right) + \tilde{O}\left((n')^{-3/2}\right)\right).
\end{aligned}$$

Thus, we can choose $b = \tilde{\Omega}(\gamma^{1/k}\sqrt{n'})$ such that if $v_i(t) \leq b$ for all $i$, then $\mathsf{err}(w_{1:k}(t)) \geq \frac{1}{2} - \gamma$.

By Lemma B.8, for all $i$,

$$T_i(b) \geq \frac{1}{k-2}\left(\bar{v}_a^{1-k/2} - (\bar{v}_a + b^2 - v_i(0)^2)^{1-k/2}\right) \geq \frac{1}{k-2}\left(\bar{v}_a^{1-k/2} - (b^2 - O(\log(k)))^{1-k/2}\right).$$

By the Chernoff bound, $\max_i |v_i(0)| \leq \sqrt{2\log(4k/\delta)}$ w.p. $\geq 1 - \delta/2$. And, by the same bound we used for $\|u_i\|_2^2$, we have that $\bar{v}_a = 1 + O(\sqrt{\log(1/\delta)/k})$ w.p. $1 - \delta/2$. Thus, for $\delta = 1/\mathrm{poly}(k)$, we have that with probability $1 - \delta$,

$$T_i(b) \geq \frac{1}{k-2}\left(\tilde{\Omega}(1) - (b^2 - O(\log(k)))^{1-k/2}\right) = \frac{1}{k-2}\cdot\tilde{\Omega}(1). \tag{B.15}$$

Also by Lemma B.8,

$$T_i(\infty) - T_i(b) \le \frac{1}{k-2}(\bar{v}_g + b^2 - v_i(0)^2)^{1-k/2} \le \frac{1}{k-2}(b^2 - v_i(0)^2)^{1-k/2}.$$

With probability $1 - 1/\mathrm{poly(k)}$,

$$T_i(\infty) - T_i(b) \le \frac{1}{k-2}(b^2 - O(\log(k)))^{1-k/2} = \frac{1}{k-2} \cdot \tilde{O}\left((n')^{1-k/2} \cdot \gamma^{2/k-1}\right). \tag{B.16}$$

Combining Equation B.15 and Equation B.16, we obtain the desired statement.

□

### B.2.4 Global convergence and phase transition for gradient flow on disjoint-PolyNets

In this section we will analyze the training of a disjoint-PolyNet using SGD with online (i.i.d.) batches. We will show a convergence result for the 0-1 error of the learned classifier.

*Theorem 5.8* (SGD on disjoint-PolyNets learns disjoint parities; full statement). Assume we randomly initialize the disjoint-PolyNet with weights drawn uniformly from $\{\pm 1\}$. Fix $\varepsilon \in (0, 1/2)$ and run SGD at any batch size $B \ge 1$ for $T \ge 6\log(2nT/\delta)\log(2k/\varepsilon)(3n' - 2)^{2k-1}$ iterations. There exists an adaptive learning rate schedule, such that, with probability $1/2$ over the randomness of the initialization and $1 - \delta$ over the sampling of SGD, the following holds:

$$\mathrm{err}\left(w_{1:k}^{(T+1)}\right) \le \varepsilon.$$

*Proof.* For simplicity of presentation, we will assume $B = 1$. Let the sample at iteration $t$ be $(x^{(t)}, y^{(t)})$ where $x^{(t)} \sim \mathsf{Unif}(\{\pm 1\}^n)$ and $y^{(t)} = \chi_S^{(t)}(x^{(t)})$. Denote the population and stochastic

gradient at time $t$ as:

$$\widehat{g}_i^{(t)} = -y^{(t)} \left( \prod_{j \neq i} {w_j^{(t)}}^\top x_j^{(t)} \right) x_i^{(t)}$$
$$g_i^{(t)} = - \left( \prod_{j \neq i} w_{j,1}^{(t)} \right) e_1$$

Observe that $|\widehat{g}_{i,j}^{(t)}|, |g_{i,j}^{(t)}| \leq \prod_{l \neq i} \|w_l^{(t)}\|_1$; note that this is also true when $B > 1$. Let the learning rate be $\eta_i^{(t)} = \frac{1}{2\sqrt{2T\log(2nT/\delta)} \cdot \prod_{l \neq i} \|w_l^{(t)}\|_1}$. Let $\Delta_{i,j}^{(t)} = \eta_i^{(t)}(g_{i,j}^{(t)} - \widehat{g}_{i,j}^{(t)})$ and $s_{i,j}^{(t)} = \sum_{\tau=1}^{t} \Delta_{i,j}^{(\tau)}$. Observe that $\mathbb{E}\, \Delta_{i,j}^{(t)} = 0$ and therefore $s_{i,j}^{(1)}, \ldots, s_{i,j}^{(t)}$ form a martingale. Furthermore, observe that

$$|s_{i,j}^{(t)} - s_{i,j}^{(t-1)}| = |\Delta_{i,j}^{(t)}| \leq \frac{|g_{i,j}^{(t)}| + |\widehat{g}_{i,j}^{(t)}|}{\sqrt{T\log(2nT/\delta)} \cdot \prod_{l \neq i} \|w_l^{(t)}\|_1} \leq \frac{1}{\sqrt{2T\log(2nT/\delta)}}$$

Then, for every $t < T$ and $i, j$, by the Azuma-Hoeffding inequality, with probability $1 - \frac{\delta}{nT}$, $|s_{i,j}^{(t)}| \leq 1/2$. By the union bound, w.p. at least $1 - \delta$, for every $t < T$ and all $i, j$ it holds that $|s_{i,j}^{(t)}| \leq 1/2$. Let us assume this holds.

*Claim* B.11. For all $t \leq T + 1$, for $j > 1$, $|w_{i,j}^{(t)}| \leq 3/2$.

*Proof.* Note that $w_{i,j}^{(t+1)} = w_{i,j}^{(1)} + s_{i,j}^{(t)}$, therefore, we have,

$$|w_{i,j}^{(t+1)}| \leq |w_{i,j}^{(1)}| + |s_{i,j}^{(t)}| \leq 3/2.$$

since $|w_{i,j}^{(1)}| = 1$ and $|s_{i,1}^{(t-1)}| \leq 1/2$. □

Denote $\xi_i = \operatorname{sign}\left(w_{i,1}^{(1)}\right)$ and assume that $\prod_{j=1}^{k} \xi_j > 0$, and note that this happens w.p. $1/2$. We will assume this holds for the next lemma.

*Claim* B.12. For all $t \le T+1$ and $i \in [k]$ it holds that

$$\xi_i w_{i,1}^{(t)} = |w_{i,1}^{(t)}| \ge \frac{1}{2} + \frac{1}{4\sqrt{2}} \left(\frac{1}{3n'-2}\right)^{k-1} \sqrt{\frac{t-1}{\log(nT/\delta)}}.$$

*Proof.* Observe that the claim holds for $t = 1$ since $|w_{i,1}^{(1)}| = 1$. By induction on $t$, assume the claim holds for all $\tau \le t$. Now we will prove it holds for $t+1$.

Observe that for all $\tau \le t$, by the assumption:

$$g_{i,1}^{(\tau)} = -\prod_{j\neq i} w_{j,1}^{(\tau)} = -\xi_i \prod_{j\neq i} \xi_j w_{j,1}^{(\tau)} = -\xi_i \prod_{j\neq i} |w_{j,1}^{(\tau)}|.$$

Note that $w_{i,1}^{(t+1)} = w_{i,1}^{(1)} + \sum_{\tau=1}^{t} \eta_i^{(\tau)} \left(\prod_{j\neq i} w_{j,1}^{(\tau)}\right) + s_{i,1}^{(t)}$, then we have

$$\begin{aligned}
\xi_i w_{i,1}^{(t+1)} &= \xi_i w_{i,1}^{(1)} + \xi_i \sum_{\tau=1}^{t} \eta_i^{(\tau)} \left(\prod_{j\neq i} w_{j,1}^{(\tau)}\right) + \xi_i s_{i,1}^{(t)} \\
&= 1 + \sum_{\tau=1}^{t} \left(\frac{1}{4\sqrt{2\tau\log(nT/\delta)}} \cdot \prod_{j\neq i} \frac{\xi_j w_{j,1}^{(\tau)}}{\|w_j^{(\tau)}\|_1}\right) + \xi_i s_{i,1}^{(t)} && \text{(B.17)} \\
&\ge \frac{1}{2} + \frac{1}{2\sqrt{2T\log(2nT/\delta)}} \cdot \sum_{\tau=1}^{t} \left(\prod_{j\neq i} \frac{|w_{j,1}^{(\tau)}|}{\|w_j^{(\tau)}\|_1}\right) - |s_{i,1}^{(t)}| && \text{(B.18)} \\
&\ge \frac{1}{2} + \frac{1}{2\sqrt{2T\log(2nT/\delta)}} \cdot \sum_{\tau=1}^{t} \left(\prod_{j\neq i} \frac{|w_{j,1}^{(\tau)}|}{|w_{j,1}^{(\tau)}| + \frac{3(n'-1)}{2}}\right) && \text{(B.19)} \\
&\ge \frac{1}{2} + \frac{1}{2\sqrt{2T\log(2nT/\delta)}} \cdot \sum_{\tau=1}^{t} \left(\prod_{j\neq i} \frac{1}{1+3(n'-1)}\right) && \text{(B.20)} \\
&\ge \frac{1}{2}\left(1 + \left(\frac{1}{3n'-2}\right)^{k-1} \frac{t}{\sqrt{2T\log(2nT/\delta)}}\right).
\end{aligned}$$

(B.17) follows from observing that $\xi_i w_{i,1}^{(1)} = |w_{i,1}^{(1)}| = 1$ and $\prod_{j=1}^{k} \xi_j = 1$. (B.18) follows from the

inductive hypothesis $\xi_j w_{j,1}^{(\tau)} = |\xi_j w_{j,1}^{(\tau)}|$. (B.19) follows from our assumption that $|s_{i,1}^{(t)}| \leq 1/2$ and Claim B.11.(B.20) follows from the inductive hypothesis $|w_{i,1}^{(\tau)}| \geq 1/2$. □

Setting $T$ such that $T \geq 2\log(2nT/\delta)\alpha^2(3n'-2)^{2k-2}$ for $\alpha = 3\sqrt{(n'-1)\log(2k/\varepsilon)} - 1$, from the above claims, after iteration $T$, we have

$$w_{i,1}^{(T+1)} \geq \frac{\alpha+1}{2}$$
$$w_{i,j}^{(T+1)} \leq 3/2 \text{ for } j > 1.$$

Using Lemma B.9, we have with probability $1-\delta$,

$$\text{err}(w_{1:k}^{(T+1)}) \leq 2k \exp\left(-\frac{(\alpha+1)^2}{9(n'-1)}\right) = \varepsilon.$$

□

## B.3 Additional figures, experiments, and discussion

This section contains our unabridged empirical results, visualizations, and accompanying discussion. Additional example training curves (like the assortment in Figure 5.1 *(left)*) are shown in Figure B.2; more examples can be found in the subsections below.

Convergence times, success probabilities, and scaling laws. We first present the full empirical results outlined in Section 5.3 of the main paper. Figure B.3 shows convergence times $t_c$ on small parity instances for all of the architecture configurations enumerated in Section 5.3.1. In some of these settings, $t_c$ exhibits high variance due to unlucky initializations (see Figure B.4); thus, we report $10^{\text{th}}$ percentile convergence times. Figure B.5 gives coarse-grained estimates for how $t_c$ scales with $(n, k)$, based on small examples. For selected architectures, Figure B.6 shows how these

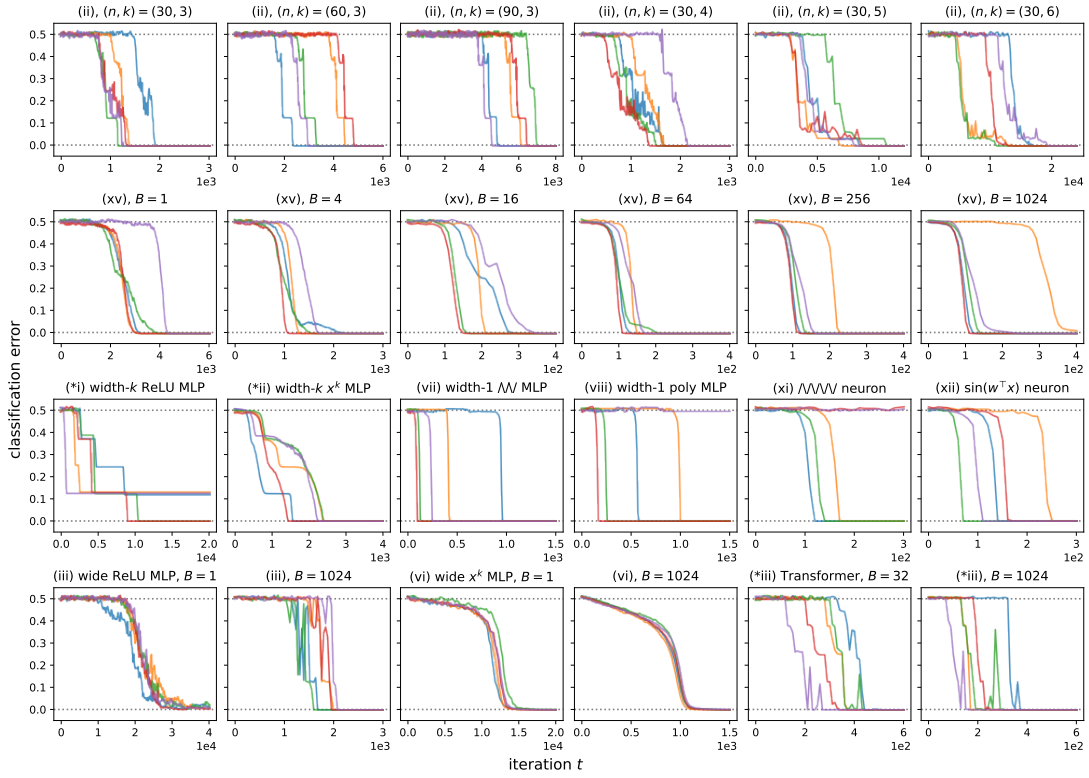



**Figure B.2:** Additional training curves (on the $(50, 3)$-parity problem, except the first row); full details are in Appendix B.4.1. *1st row:* The same architecture, initialization, and training algorithm (a width-100 ReLU MLP in this case), without an explicit sparse prior, adapts to the computational difficulty parameters $n, k$. *2nd row:* Our positive empirical results hold over a wide range of batch sizes $B$, all the way down to $B = 1$. Training is unstable (more outliers) at very large and very small batch sizes. *3rd row:* Even the least overparameterized neural networks, which are barely wide enough to represent parity, converge with reasonable probability (sometimes failing to reach a global minimum). *4th row:* Larger models (width-1000 MLPs and $d_{\text{emb}} = 1024$ Transformers) are robust to a wide range of batch sizes. Note the lack of plateaus in setting (vi), which is revisited in Appendix B.3.8.

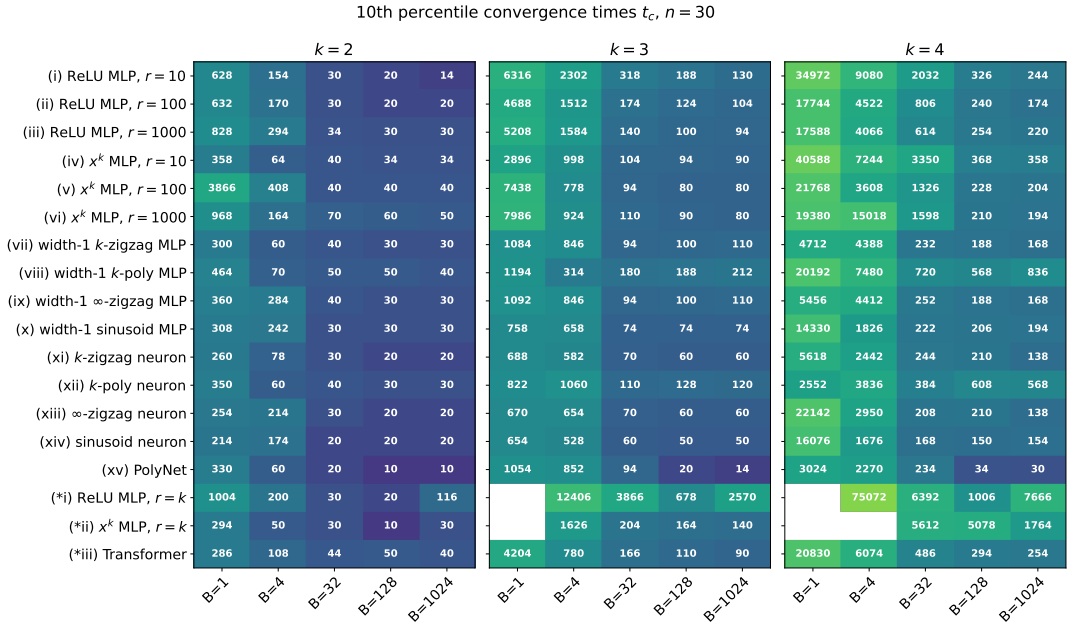



**Figure B.3:** 10th percentile convergence times $t_c$ of SGD (with $n = 30$, hinge loss, uniform initialization, and best learning rate $\eta \in \{1, 10^{-1}, 10^{-2}, 10^{-3}\}$), for various architectures, parity degrees $k$, and batch sizes $B$. See Appendix B.4 for full details.

convergence times scale with $n$ and $k$ more precisely: for small $n$, power law relationships $t_c \propto n^{\alpha \cdot k}$ (for small constants $\alpha$) are observed for all configurations. Note that for larger $n$, the exponent (i.e. the slope in the log-log plot) increases: with a constant learning rate and standard training, the $n^{\Theta(k)}$ does not continue indefinitely. All additional details are in Appendix B.4.

GUIDE TO THIS SECTION. The remainder of Appendix B.3 expands on the various discussions and figures from Sections 5.4 and 5.5.

- Appendix B.3.1 gives experimental evidence that Fourier gaps are present at iterates $w_t$ and initializations $w_0$ other than sign vectors, as well as for activation functions other than ReLU. This suggests that the feature amplification mechanism is robust, and illuminates directions for strengthening the theoretical results.


**Figure B.4:** Percentage of converged runs ($t_c < 10^5$) out of 25 random trials, with the same architectures and training parameters as Figure B.3. With sufficiently large batch sizes, training is extremely robust in settings (i) through (xv).

- Appendix B.3.2 discusses how the building blocks of deep learning (activation functions, biases, initializations, learning rates, and batch sizes) play multiple, sometimes conflicting roles in this setting.
- Appendix B.3.3 provides additional white-box visualizations of hidden progress from Figure 5.3.
- Appendix B.3.4 explores the implications of the feature amplification mechanism for scaling model size– namely, unlike random search, large width does not impart parallel speedups.
- Appendix B.3.5 shows that our results hold in the finite-sample setting (allowing for multiple passes over a training set of size $m$). In particular, we show that in low-data regimes, the models exhibit the *grokking* phenomenon.
- Appendix B.3.6 extends our results to *noisy* parities (which comprise the true "emblematic computationally-hard problem").

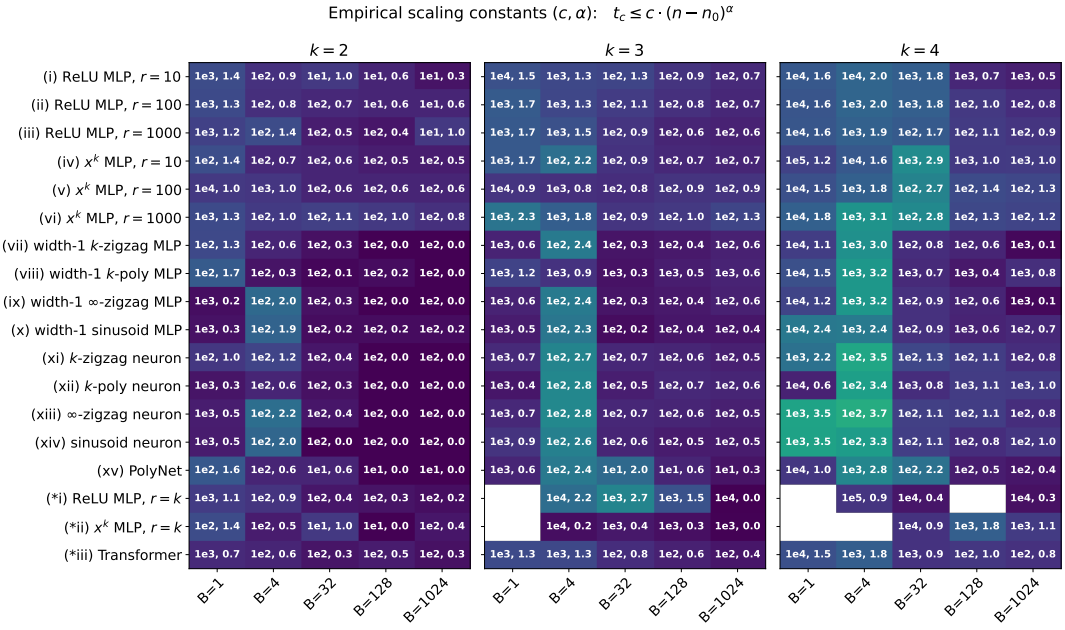



**Figure B.5:** Coarse scaling estimates for the 10th percentile convergence time of SGD (with hinge loss, uniform initialization, and best learning rate) on every architecture: $c, \alpha$ such that $t_c \le c \cdot (n - n_0)^\alpha$ on small parity instances $n \in \{10, 20, 30\}, k \in \{2, 3, 4\}$. Missing entries denote cases where $< 10\%$ of trials were convergent for any $n$ (see Figure B.4). Boxes are colored according to $\alpha$. See Appendix B.4 for full details.

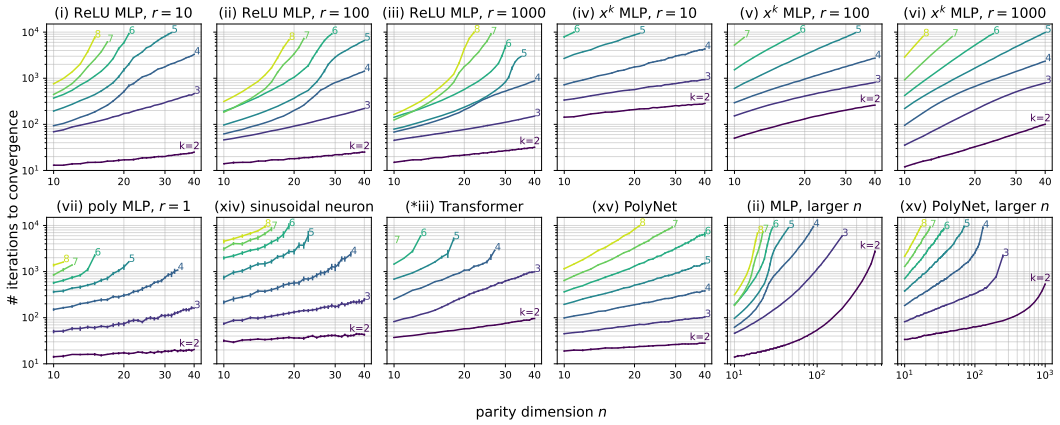



**Figure B.6:** Finer-grained plots of convergence times for selected architecture configurations, for $n \in \{10, 11, \ldots, 39, 40\}$ and $k \in \{2, 3, \ldots, 8\}$. Medians over 1000 runs are shown, with 95% bootstrap confidence intervals. *Top row:* standard MLP configurations (i) through (vi). *Bottom row:* Miscellaneous settings (1-neuron networks; Transformers; larger $n$). For details on each setting, see Appendix B.4.1.

- Appendix B.3.7 introduces a counterexample for "layer-by-layer learning", using parity distributions whose degrees are higher than those of the individual layers' polynomial activations. Preliminary experiments show that standard training works in this setting.
- Appendix B.3.8 presents examples of training curves for wide polynomial-activation MLPs, where, unlike the other settings, there is no initial plateau in the model's error.

### B.3.1 Fourier gaps at initialization and SGD iterates

Proposition B.7 shows that if the function $x \mapsto \sigma'(w_0^\top x)$ has a Fourier gap at $S$, then $S$ can be identified from a batch gradient at initialization $w_0$ with $B = O(1/\gamma^2)$ samples. Our end-to-end result (Theorem 5.5) requires ReLU activations and sign vector initialization, because the Fourier gap condition (Definition 5.4) arises from exact formulas for the Fourier coefficients of the majority function. Stronger end-to-end theoretical guarantees would follow from analogous Fourier gaps in more general population gradients. This requires $x \mapsto \sigma'(w^\top x)$ to satisfy these conditions simultaneously:

- *Fourier concentration:* upper bounds on the degree-$(k+1)$ coefficients $\widehat{f}(S \cup \{i\})$, for $i \notin S$. The term is borrowed from Klivans et al. (2004), who use upper bounds on Fourier coefficients of LTFs to approximate them (thus, learn halfspaces) with low-degree polynomials.
- *Fourier anti-concentration:* lower bounds on the degree-$(k-1)$ coefficients $\widehat{f}(S \setminus \{i\})$, for $i \in S$.

A natural question is: *which Boolean functions, other than majority, satisfy the $\gamma$-Fourier gap property at $S$, for $\gamma \geq n^{-\Omega(k)}$?*

We present some numerical evidence for large Fourier gaps in functions $x \mapsto \sigma'(w^\top x)$ other than majority, which arise from gradients of architectures other than ReLU MLPs with sign initialization. This shows that the mechanism of feature emergence is empirically robust in settings not fully explained by our current theory. Establishing corresponding theoretical guarantees would enable stronger end-to-end global convergence guarantees for MLPs and other architectures.

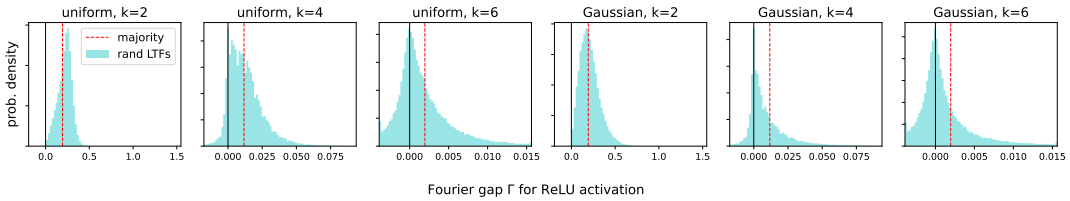



**Figure B.7:** Distributions of exact Fourier gaps $\Gamma_{\sigma,k}(w)$ when $\sigma = \mathrm{ReLU}$, and $w$ is an i.i.d. {uniform, Gaussian} random vector. These are derived from the Fourier coefficients of the corresponding random LTFs, computed here in exponential time for $n = 15$.

In these experiments, population gradients were computed by brute force integration over all $2^n$ Boolean inputs $x \in \{\pm 1\}^n$. In all cases, for various choices of $\sigma$, $w$, we measure a slightly relaxed notion of Fourier gap $\Gamma$ in the population gradient:

$$\Gamma_{\sigma,k}(w) := \max_{i \in [k]} |g_i| - \max_{i \notin [k]} |g_i|, \qquad g := \mathbb{E}[yx\sigma'(w^\top x)].$$

If $\Gamma > 0$, then *one*[*] coordinate from the parity can be identified from $O\left(\frac{\log n}{\Gamma^2}\right)$ samples of the gradient at $w$.

Random LTFs. For ReLU activations and symmetric Bernoulli (i.e. random sign) initialization $w_i \sim \mathrm{Unif}(\{\pm c\}^n)$, the Fourier coefficients are the same as those of majority; thus, there is a Fourier gap of $\gamma \geq n^{-\Omega(k)}$ at every set $S$ (and the same is true of $\Gamma$). We probe the Fourier gaps of linear threshold functions (LTFs) under other ubiquitous initializations: i.i.d. uniform and Gaussian. These are shown in Figure B.7, which indicates (at least for small $n, k$) that the Fourier gap is comparable to that of majority with non-negligible probability.

[*]Replacing the first max in the definition of $\Gamma$ with min would give us the same notion of Fourier gap as Definition 5.4: if *all* the relevant coordinates are larger than *all* of the irrelevant ones, estimating the population gradient allows us to recover the relevant coordinates.

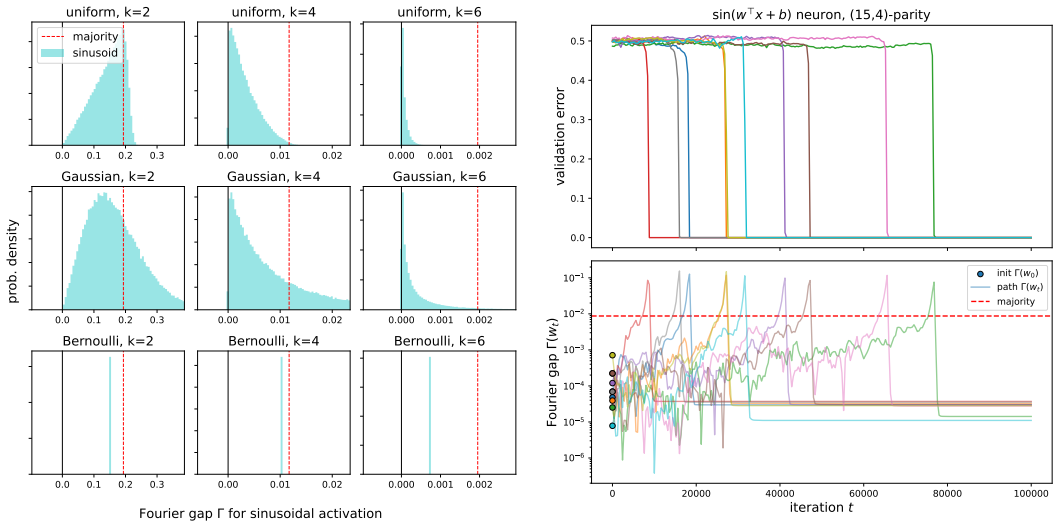



**Figure B.8:** Numerical evidence for Fourier gaps beyond random LTFs. *Left:* Fourier gaps of exact population gradients $\mathbb{E}_{\mathcal{D}_S}[yx_i \sin(w^\top x/\sqrt{n})]$ induced by sinusoidal activations, for random initializations $w$. In these small cases, they are comparable the the Fourier gaps of majority (red dashed line). *Right:* Fourier gaps of a sinusoidal neuron's population gradients along the SGD optimization path (10 trials shown). The Fourier gap is consistently positive at initialization, but somewhat smaller than that of majority (red dashed line). Interestingly, it amplifies through the course of training.

RANDOM NON-LTFS. The successful convergence of architectures with smoother activations (in the parity setting and beyond) motivates the question of whether large Fourier gaps are present in population gradients corresponding to functions other than LTFs. Figure B.8 *(left)* shows that this is the case for sinusoidal activations.

BOOLEAN FUNCTIONS ALONG THE SGD PATH. Finally, to further close the gap between Theorem 5.5 and the empirical results, it is necessary to address the fact that SGD accumulates gradients with respect to *time-varying* iterates, while our analysis approximates this using a large-batch gradient at a static iterate $w_0$. In fact, SGD seems to help in some cases: Figure B.8 *(right)* shows that when training a sinusoidal neuron, SGD amplifies the initial Fourier gap.

### B.3.2 Counterintuitive roles of the building blocks of deep learning

Even in this simple problem setting, the simultaneous computational and statistical considerations lead to counterintuitive consequences for the optimal configurations of architectures and algorithms for this setting. We encountered the following, in the search for architecture configurations for the empirical study:

- *Activation functions.* This mechanism of features emerging via Fourier gaps (see Definition 5.4) is strongest with non-smooth activations such as the ReLU, whose derivatives are discontinuous threshold functions. This is an orthogonal consideration to representational capacity and mitigation of local minima (under which one might conclude that degree-$k$ polynomial activations are optimal). In summary, in feature learning settings where the Fourier gaps and low-complexity solutions are simultaneous relevant, there is a *sharpness-smoothness tradeoff* for the activation function.

- *Biases.* The symmetry of the majority function (as well as all unbiased LTFs) causes its even-degree Fourier coefficients to be zero; thus, certain variants of the setups in Section 5.3 fail for odd $k$. Bias terms (trainable or fixed) are necessary to break this symmetry, in theory and practice. Simultaneously, biases serve the more conventional role of shifting the loss surface; see Section B.4.1 for how this affects the details of how the biases were chosen in the experiments.

- *Initializations.* The role of the initialization distribution is similarly twofold in this setting: $w_0$ should be close to the desired solution $w^*$, but it must also be selected such that SGD will successfully amplify the Fourier gap. A third consideration, which we do not attempt to study in this work, is that multiple randomly-initialized neurons will tend to learn the correct features at different times (see the weight trajectory visualizations in Figure 5.3 and Figure B.9, as well the staircase-like training curves seen for MLPs in Figure 5.1 *(left)*). We expect this *symmetry breaking*

phenomenon to be present in more complex feature learning settings. Finally, as shown in the training curves from setting (vi) in Figure B.2, and in more detail in Appendix B.3.8, the choice of activation function influences the qualitative behavior of the training curves: namely, whether the plateaus disappear at large widths and batch sizes.

- *Batch sizes and learning rates.* The empirical and theoretical results both suggest that SGD uses independent samples to gradually amplify a signal containing the correct features in the initial population gradient of the correlation $\left(\nabla_w \mathbb{E}[-yx_i\sigma'(w^\top x)]\right)|_{w=w_0}$. However, it would be truer to this mechanism to stay at the initialization $w_0$ until the algorithm has accumulated enough data to discern the correct indices (equivalently, scale up the batch size); in contrast, standard training takes gradients with respect to $w_t$ along the SGD path. We hypothesize that the bias incurred by this drifting $w_t$ (and thus drifting population gradient) accounts for the degradations seen in Figure 5.2 *(right)* and Figure B.6. However, Figure B.8 *(right)* shows that the movement of SGD can be helpful, amplifying the Fourier gap.

### B.3.3 Hidden progress measures

In this section, we provide an expanded discussion and plots for the investigations outlined in Figure 5.3 and the *"hidden progress measures"* section in Section 5.5.

For a neural network training pipeline which outputs a sequence of iterates $\theta_0, \ldots, \theta_T \in \Theta$, we define a *progress measure* $\rho : \Theta \to \mathbb{R}$ to be any function of the training algorithm's state* which is predictive of the time to convergence (i.e. conditioned on $\theta_t$, the random variables $\rho$ and $t_c - t$ are not independent). By this definition, the only algorithms which have no progress measures are those whose convergence times $t_c$ are *memoryless* (independent of $\theta_t$).

*In addition to the model's parameters, the full state of the training procedure should also include the auxiliary variables defined by the optimization algorithm; two ubiquitous ones in deep learning are the momentum vector and the adaptive preconditioner. Here, we only consider vanilla SGD, which maintains no auxiliary variables.

Note that many trivial progress measures exist: an example to keep in mind is that for the algorithm which exhaustively enumerates over a deterministic list of hypotheses (say, the possible $k$-element subsets $S$ in lexicographical order) and terminates when it finds the correct one, the current iteration $t$ is a progress measure. Thus, the purpose of demonstrating hidden progress measures $\rho$ is *not* to provide further evidence that SGD finds the features using Fourier gaps. Rather, it is to (1) further refute the hypothesis of SGD performing a memoryless Langevin-like random search, and (2) provide a preliminary exploration of how progress can be quantified even when the natural metrics of loss and accuracy appear to be flat.

Fourier gaps over time. The Fourier gap visualizations in Section B.3.1 already provides an example of a quantity which varies continuously as the model trains, despite no apparent progress in the loss and accuracy curves. However, none of our theoretical analyses capture the empirical observation that this quantity tends to amplify over time. Below, we consider other quantities which reveal hidden progress in parity learning, which are more straightforward and closer to our analyses.

Weight movement. The most direct observation of hidden progress simply comes from the movement of the neurons' weights at the relevant indices: that is, for a single neuron's weights $w_t \in \mathbb{R}^n$, the quantity $\rho([w_t]_i) \ : \ i \in S$. In the main paper, Figure 5.3 *(left, center)* directly visualizes the evolution of the weights $w_t$ for a single sinusoidal neuron (with a bias, but no second layer). Figure B.9 supplements these plots from the main paper with additional plots of weight trajectories, at different batch sizes $B$, as well as a width-10 MLP architecture. As seen in these plots, progress only becomes visible in the loss once the relevant weights become larger than all of the irrelevant weights.

$\ell_\infty$ path length. Finally, we present an example of a *single* measurement of the optimization path which captures hidden progress in this setting, which can be plotted alongside loss and accu-

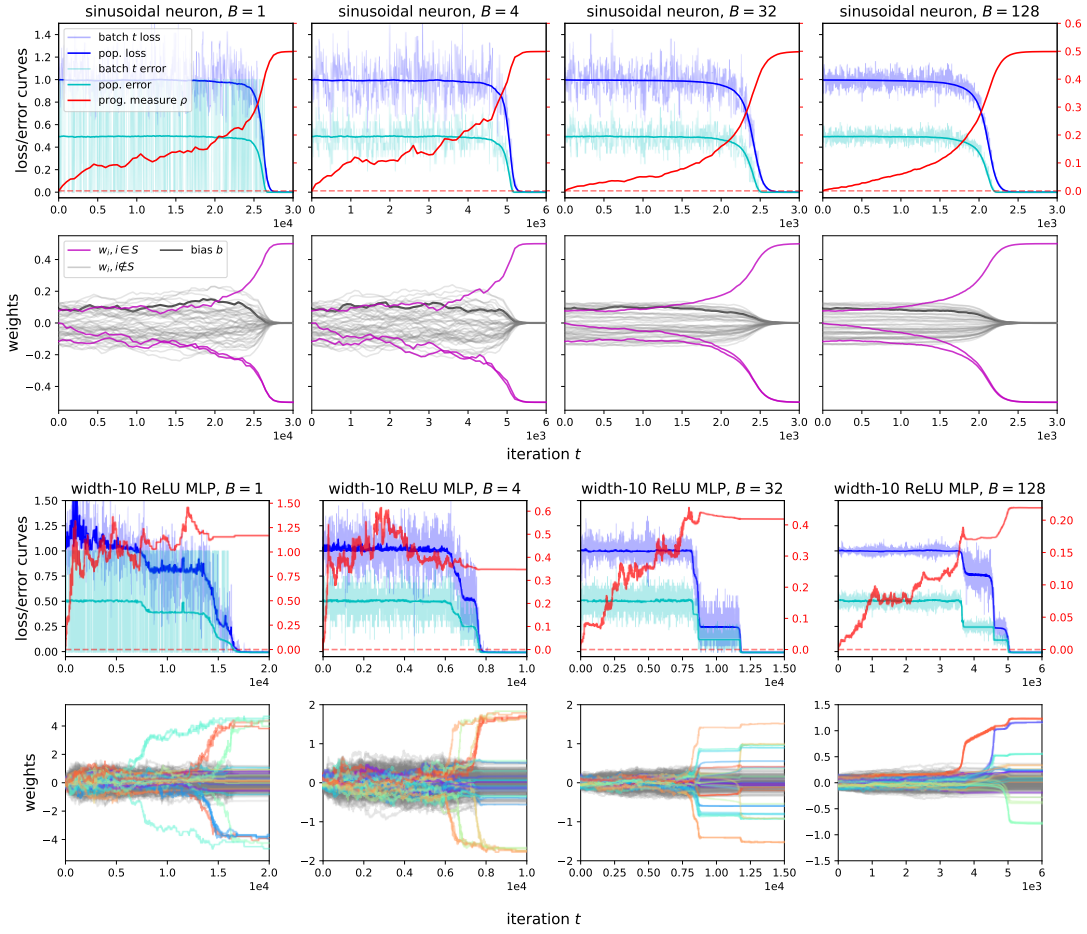



**Figure B.9:** Supplementary plots for visualizations of the optimization trajectory $w_t$, and the hidden progress measure $\rho(w_{0:t})$. In the MLP plots, the single scalar $\rho$ is the $\infty$-norm of the entire first layer $W$, and weights are color-coded by row (i.e. neuron). With small batch sizes $B \in \{1, 4\}$, per-iteration losses are averaged over a short window (lengths 16 and 4, respectively).

racy curves. For any iterates of a neuron's weights $w_t \in \mathbb{R}^n$, we choose the $\infty$-norm of the movement from initialization: $\rho(w_{0:t}) := ||w_t - w_0||_\infty$. We present a brief intuitive sketch of the motivation for this choice of $\rho(\cdot)$, and some additional visualizations.

From the theoretical analysis, under the approximation $\nabla\ell(w_t) \approx \nabla\ell(w_0)$ (so that feature learning is performed by estimating the initial population gradient to high precision), we can think of the $i$-th coordinate of $w_t$ as a biased random walk with constant variance $\sigma^2$; the Fourier gap condition entails that biases $\beta_i$ of these random walks are large when $i \in S$. Then, this choice of $\rho$ is an estimate for the drift term $t \cdot \max_i |\beta_i|$, which is larger than the $\sigma\sqrt{t}$ contribution of the variance for sufficiently large $t$.

This progress measure is shown alongside the loss curves in Figure B.9, in red. We do not attempt to characterize the dynamics of $\rho$; we only note that they are clearly distinguishable from the maximum of $n$ unbiased random walks, even when SGD appears to make no progress in terms of loss and accuracy. Studying hidden progress measures in deep learning more quantitatively, as well as in more general settings, presents a fruitful direction for future work.

### B.3.4 Convergence time vs. width

We provide supplementary plots for the experiment outlined in Figure 5.4 *(left)*, which probes whether extremely large widths ($r \gg n^k$) afford factor-$r$ parallel speedups of the parity learning mechanism (as one would expect from random search). On 3 parity instances $n \in \{30, 40, 50\}, k = 3$, we varied the width $r \in \{1, 2, 3, \ldots, 9, 10, 30, 100, 300, \ldots, 10^6, 3 \times 10^6\}$, keeping all other parameters the same ($B = 128, \eta = 0.1$).

Results. We did not find evidence of such parallel speedups over 1000 runs in each setting; see Figure B.10. This serves as further evidence that the mechanism by which standard training solves parity is best understood as deterministic and sequential, rather than behaving like random search

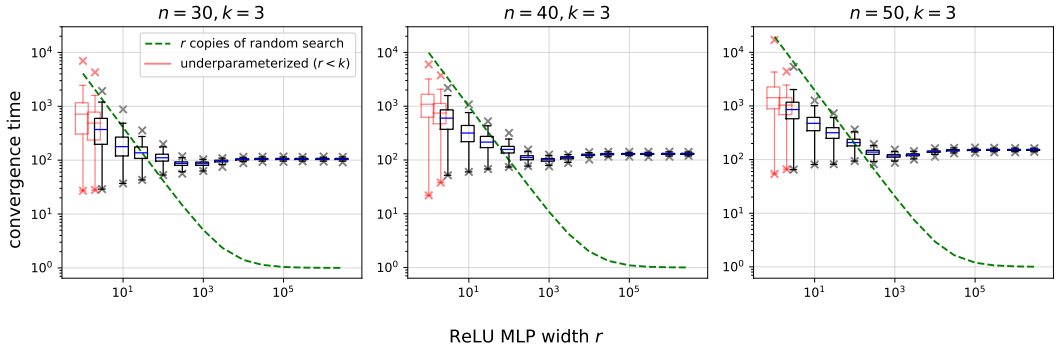



**Figure B.10:** Number of iterations for MLPs (with standard initialization and training) to converge on sparse parity problems, in terms of width $r$. Boxes denote interquartile ranges over 1000 runs; whiskers denote $\pm 1.5 \cdot$ IQR; $\times$ markers denote minimum and maximum outliers. Underparameterized models ($r < k$) are shown in red, and considered "converged" at 55% accuracy. Scaling $r$ this way does not lead to $r\times$ parallel speedups, like the expected success time for $r$ copies of random search (shown in green for comparison).

over size-$k$ subsets. A benefit of width appears to be *variance reduction*: the upper tail of long convergence times is mitigated by a large number of randomly-initialized neurons.

### B.3.5 Learning and grokking in the finite-sample case

We provide some supplementary plots for the experiments outlined in Figure 5.4 *(right)*. In these settings, a fixed architecture (width-100 MLP with ReLU activations) is trained with minibatch SGD in an otherwise fixed configuration (hinge loss, learning rate $\eta = 0.1$, batches of size $B = 32$) on a finite training sample of size $m$. We also vary a *weight decay* parameter $\lambda$.

As shown in Figure B.11, the weight decay parameter $\lambda$ modulates a delicate computational-statistical tradeoff: it improves generalization (expanding the range of $m$ for which training eventually finds the correct solution), but the model fails to train at large values of $\lambda$. For small $m$ and appropriately tuned $\lambda$, we observe *grokking*: the model initially overfits the training data, but finds a classifier that generalizes after a large number of iterations.

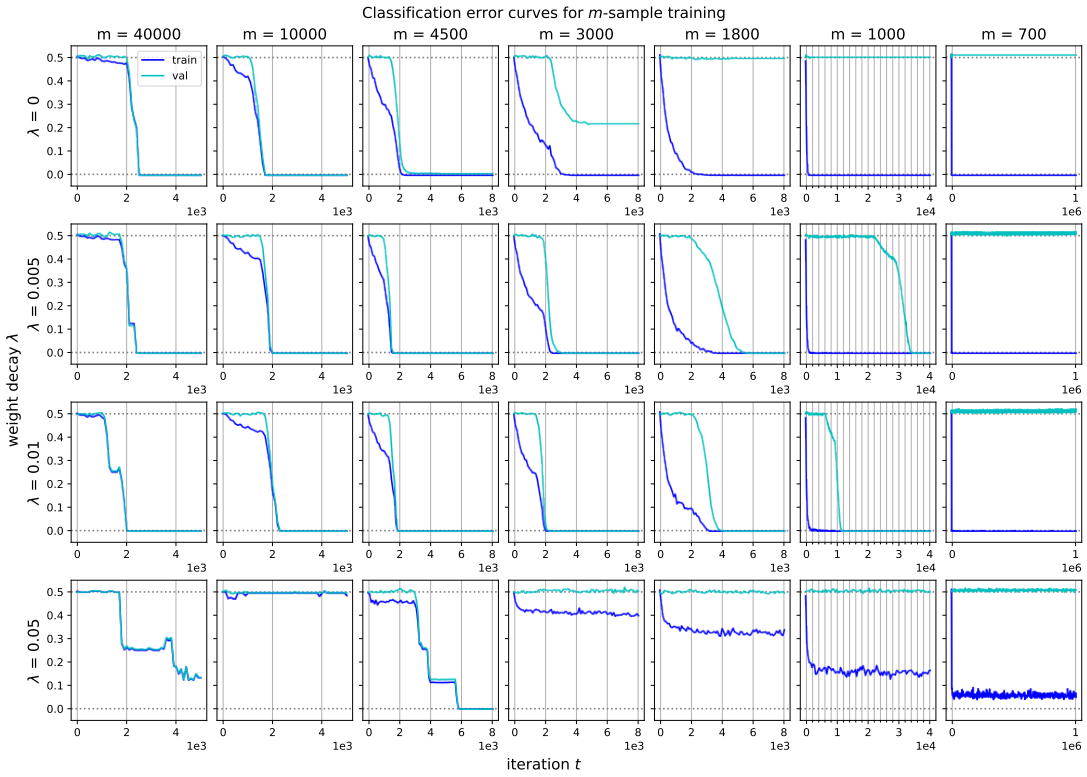



**Figure B.11:** Supplementary plots for the finite-sample setting. The same configuration (width-100 ReLU MLP, $B = 32, \eta = 0.1$) varying the sample size $m$ (decreasing from left to right) and weight decay $\lambda$ (increasing from top to bottom). When $m$ is sufficiently large (much larger than the statistical threshold $\Theta(k \log n)$), generalization error is negligible. When $m$ is too small, the model fails to train. In between, we observe eventual convergence to the correct solution, with training curves exhibiting the *grokking* phenomenon. Weight decay governs a statistical-computational tradeoff in this setting: larger $\lambda$ improves generalization, but can cause optimization to fail (bottom row).

### B.3.6 Learning noisy parities

The other empirical results in this work focus on noiseless parity distributions $\mathcal{D}_S$, to reduce the number of sources of variance and degrees of freedom. However, the setting of *random classification noise* is important for several reasons. In this section, we briefly demonstrate that our results extend to this case. Let $\mathcal{D}_S^{(\varepsilon)}$ denote the $(n, k, \varepsilon)$-*noisy parity distribution*, defined by flipping the labels in the $(n, k)$-parity distribution $\mathcal{D}_S$ independently with probability $\frac{1}{2} - \varepsilon$. Note that when $\varepsilon = 0$, the labels are completely random (thus, $S$ cannot be learned). By a standard PAC-learning argument, when $0 < \varepsilon \leq \frac{1}{2}$, the statistical limit for identifying $S$ from i.i.d. samples from $\mathcal{D}_S$ scales as $\Theta\left(\frac{k \log n}{\varepsilon^2}\right)$.

MOTIVATIONS. First, learning parities from *noisy* samples is the true "emblematic computationally-hard distribution". Without noise, there is a non-SQ algorithm which avoids the exponential-in-$k$ computational barrier: Gaussian elimination can identify $S$ in $O(n^3)$ time and $\Theta(n)$ samples. Second, viewing parities as an idealized setting in which to understand training dynamics, resource scaling, and emergence in deep learning, it is important to see that this phenomenon is robust to label noise.

THEORY. It is easy to incorporate label noise into the theoretical analysis, which works with correlations of the form $\mathbb{E}_{\mathcal{D}_S}[yf(x)]$; each coordinate of the population gradient of the correlation loss is a quantity of this form. In the noisy case, these quantities are replaced with

$$\mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}_S^{(\varepsilon)}}[yf(x)] = \varepsilon \cdot \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}_S}[yf(x)].$$

In particular, when architecture's population gradient has a Fourier gap with parameter $\gamma$ in the noiseless case implies a Fourier gap with parameter $\varepsilon \cdot \gamma$.

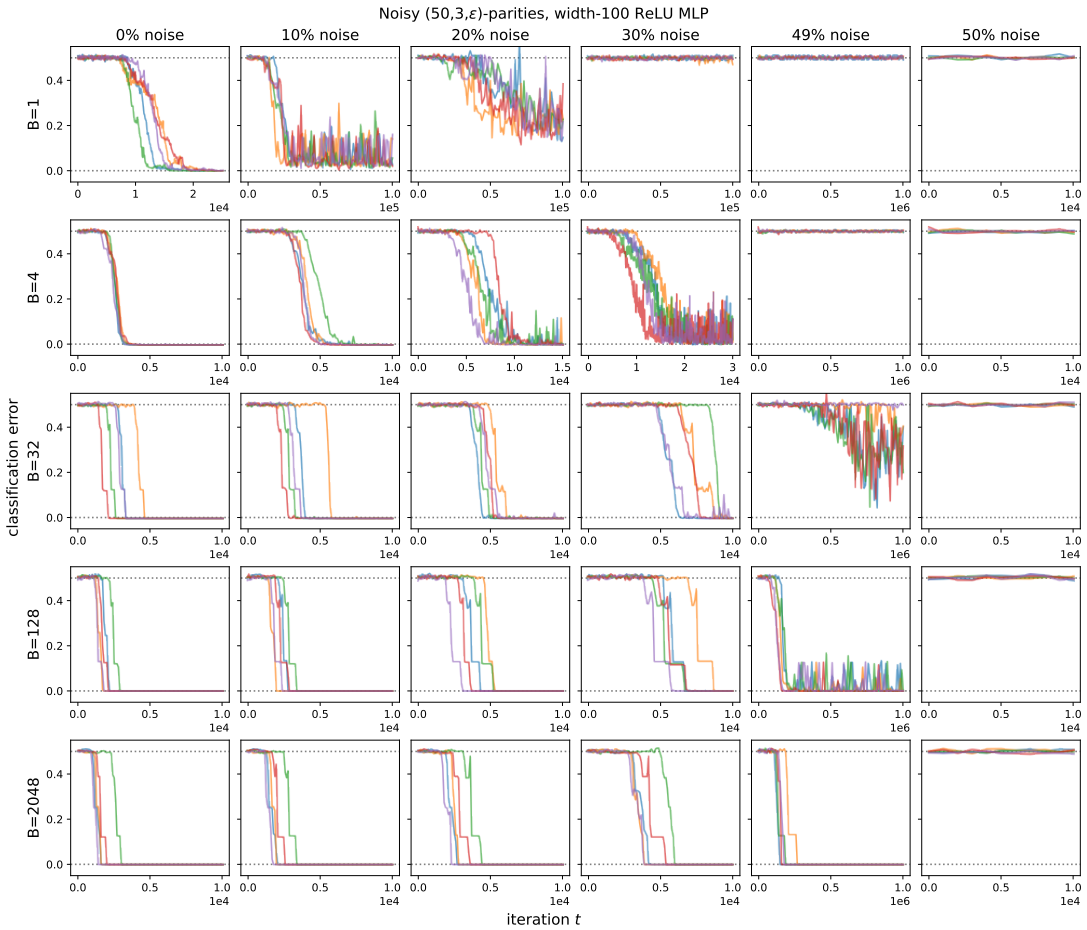



**Figure B.12:** Training curves (over 5 random seeds) for a width-100 ReLU MLP on noisy $(50, 3, \varepsilon)$-parity learning problems at various batch sizes $B$ and noise levels (flipping labels with probability $p$, so that $\varepsilon = 1 - 2p$). The models learn the features and converge successfully (measured by accuracy on the noiseless distribution), even with $49\%$ of labels flipped randomly (i.e. $\varepsilon = 0.01$). This is a preliminary illustration that the phenomena investigated in this paper are robust with respect to i.i.d. label noise. Note that the scale of $t$ is much larger for small batches and high noise.

Experiments. We find that the experimental findings are robust to label noise, in the sense that models are able to obtain nontrivial (and sometimes 100%) accuracy; see Figure B.12 for some training curves under various settings of $\varepsilon$. This provides concrete evidence against the (already extremely dubious) hypothesis that neural networks, with standard initialization and training, learn noiseless parities by implicitly simulating an efficient algorithm such as Gaussian elimination. Note that with a constant learning rate (here, $\eta = 0.1$) and label noise, the iterates of SGD do not always converge to 100% accurate solutions.

### B.3.7 Counterexample for layer-by-layer learning

Notation. Consider an $L$-layer MLP with activation $\sigma$, parameterized by weights and biases

$$\theta = (W_1, b_1, \ldots, W_{L-1}, b_{L-1}, u),$$

and defined by

$$f_{\mathsf{mlp}}(x; \theta) := (f_L \circ f_{L-1} \circ \cdots \circ f_2 \circ f_1)(x),$$

where $f_i$ denotes the function $z \mapsto \sigma(W_i z + b_i)$ for $1 \le i \le L-1$, and $f_L$ denotes $z \mapsto u^\top z$. The shapes of the parameters $W_i, b_i, u$ are selected such that each function composition is well-defined. Let the *intermediate activations* at layer $i$ be denoted by

$$z_i(x; \theta) := (f_i \circ \ldots \circ f_1)(x).$$

Finally, $r_i$ (the *width at layer $i$*) refers to the dimensionality of $z_i$ as defined above.

Construction where layer-by-layer learning is impossible. Notice that when $\sigma$ is a degree-2 polynomial (say, $\sigma(z) = z^2$), an $L$-layer MLP can represent parities up to degree $2^{L-1}$– for

example, a 3-layer MLP (which composes quadratic activations twice) can represent a 4-sparse parity as a 2-sparse parity of 2-sparse parities. However, Equation (B.1) implies the following:

- An individual layer cannot represent a parity of $k > 2$ inputs.
- The population gradient (as in Equation (B.4)) is zero (since every coordinate of the gradient is the correlation between a $k$-wise parity and a polynomial of degree 2).

Thus, this setting serves as an idealized counterexample for layer-by-layer learning: if SGD succeeds on parities with higher degree than the architecture's polynomial activations, it must do so by an end-to-end mechanism. Intuitively, earlier layers can only make progress by *knowing how their outputs will be used downstream*. Concretely, consider the population gradient of the correlation loss, with respect to a first-layer neuron's weights $w := (W_1)_{j,:}$. With layer-by-layer training, this gradient contains no information:

$$\nabla_w \mathbb{E}[-yx_i \underbrace{\sigma'(w^\top x)}_{\text{degree 1}} u_j] = 0.$$

However, in end-to-end training, the presence of downstream layers removes this barrier:

$$\nabla_w \mathbb{E}\left[ - yx_i \underbrace{\sigma'(w^\top x)}_{\text{degree 1}} \underbrace{\frac{\partial f_L \circ \ldots \circ f_2}{\partial (z_1)_j}}_{\text{degree } 2^{L-2}-1} \right],$$

giving the gradient greater representation capacity (in terms of polynomial degree). The question remains of whether end-to-end training works in this setting, which we resolve positively in small experiments.

Results: end-to-end training works empirically. We empirically observed successful training (to 100% accuracy) in a few settings (with SGD, learning rate $\eta = 0.01$, batch size $B = 32$, and default uniform initialization as described in Appendix B.4.1):

- $L = 3, n \in \{10, 20, 30\}, k \in \{1, 2, 3, 4\}$. Small widths suffice: $(r_1, r_2) = (2, 1)$. Over 10 random seeds, all models converged within 20000 iterations.

- $L = 4, n \in \{10, 20, 30\}, k \in \{1, 2, 3, 4, 5, 6\}$. Widths were chosen to be slightly larger for stability: $(r_1, r_2, r_3) = (10, 10, 1)$. Over 10 random seeds, all models converged within 50000 iterations. Additionally, models trained on $(n, k) \in \{(10, 7), (20, 7), (30, 7), (10, 8)\}$ converged within 500000 iterations.

As a sanity check, the models failed to converge in experimental setups where $k > 2^{L-1}$: $(L = 2, k \geq 3)$ and $(L = 3, k \geq 5)$.

Discussion. This construction serves as a simple counterexample to the *"deep only works if shallow is good"* principle of Malach & Shalev-Shwartz (2019), demonstrating a case where a deep network can get near-perfect accuracy even when greedy layerwise training (e.g. (Belilovsky et al., 2019)) cannot beat trivial performance. It remains to characterize these positive empirical results theoretically, as well as to investigate whether there are pertinent analogues in real data distributions.

### B.3.8 Lack of plateaus for wide polynomial-activation MLPs

An interesting qualitative observation from the training curves in Figure B.2 is that the validation accuracy curves in setting (vi) (width-1000 polynomial-activation MLPs) do not follow the same "plateau" or "staircase" pattern as the others. Figure B.13 shows a few additional examples of training curves for polynomial-activation MLPs, varying the width $r$ and batch size $B$. We find that the rate of descent of the validation error increases with both of these parameters; note that this does not occur with ReLU activations (where there are sharp phase transitions between plateaus at all batch sizes).

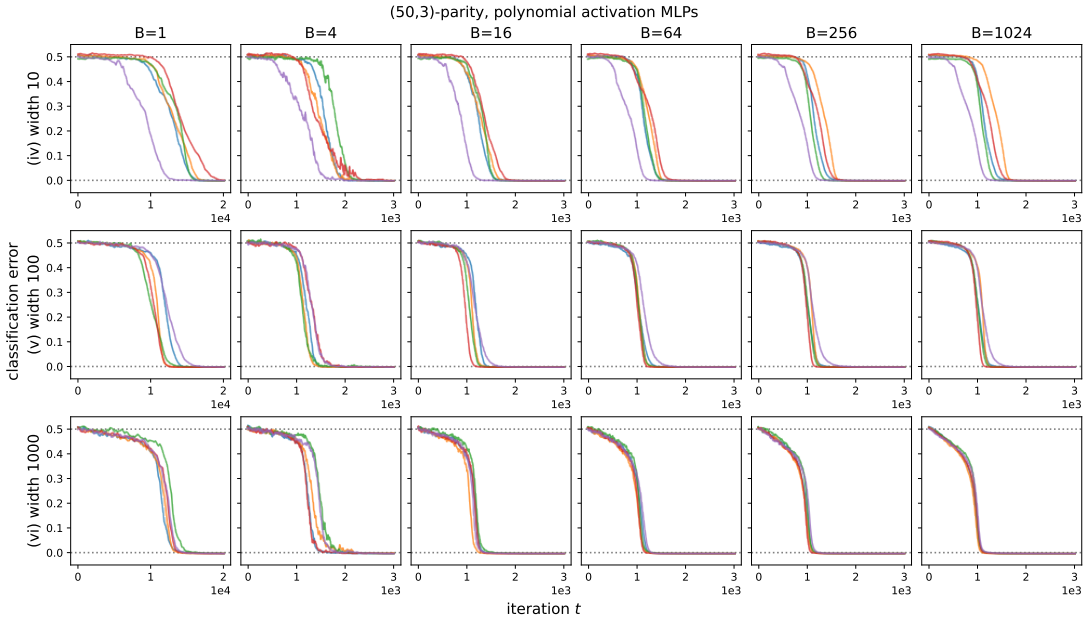



**Figure B.13:** Additional training curves for polynomial-activation MLP architectures (iv), (v), (vi), on the (50, 3)-sparse parity problem. Unlike the other architectures, these settings exhibit continuous progress when the width $r$ and batch size $B$ are large.

This constitutes an exception to this paper's theme of "hidden progress" behind flat loss (or error) curves: with enough overparameterization *and* "over-sampling", the continuous progress of SGD in this setting is no longer hidden, and manifests in the training curves. This phenomenon seems to be specific to certain activation functions (i.e. $x^k$ but not ReLU); we leave it for future work to understand why and when it occurs, as well as potential practical implications.

## B.4 Details for all experiments

### B.4.1 Deep learning configurations

Losses. Our "robust space" of empirical results use the following loss functions:

- Hinge: $\ell(y, \widehat{y}) := (1 - y\widehat{y})_+$.

- Square: $\ell(y, \widehat{y}) := (y - \widehat{y})^2$.
- Cross entropy: $\ell(y, \widehat{y}) := -\log \frac{e^{y\widehat{y}}}{1+e^{y\widehat{y}}}$.

Additionally, the theoretical analysis considers the *correlation loss* $\ell(y, \widehat{y}) := -y\widehat{y}$.

In the configurations corresponding to all of the figures and convergence time experiments, we used the hinge loss. This was a relatively arbitrary choice (i.e. they appeared to be interchangeable upon running small experiments); an advantage of the hinge and square losses over cross entropy is that for architectures that can realize the parity function, there is a zero-loss solution with finite weights.

Initializations. Our empirical results use the following i.i.d. weight initializations:

- Uniform on the interval $[-c, c]$, where the scale $c$ is chosen for all affine transformation parameters using the "Xavier initialization" convention (Glorot & Bengio, 2010). The experiments are quite tolerant to the particular choice of $c$ (as these are not deep networks); this choice, which is the default in deep learning packages, emphasizes that our positive empirical results hold under a *standard* initialization scheme.
- Gaussian with mean 0 and variance $\sigma^2$, selected using the "Kaiming initialization" convention (He et al., 2015).
- Bernoulli (i.e. random sign) initialization: the discrete distribution $\mathrm{Unif}(\{-c, c\})$, for the same choice of $c$ as for the uniform distribution.

### 2-layer MLPs

We consider 2-layer MLPs $f(x; W, b, u) = u^\top \sigma(Wx + b)$ for two choices of activations:

- ReLU: $\sigma(z) := (z)_+ = \max(0, z)$.

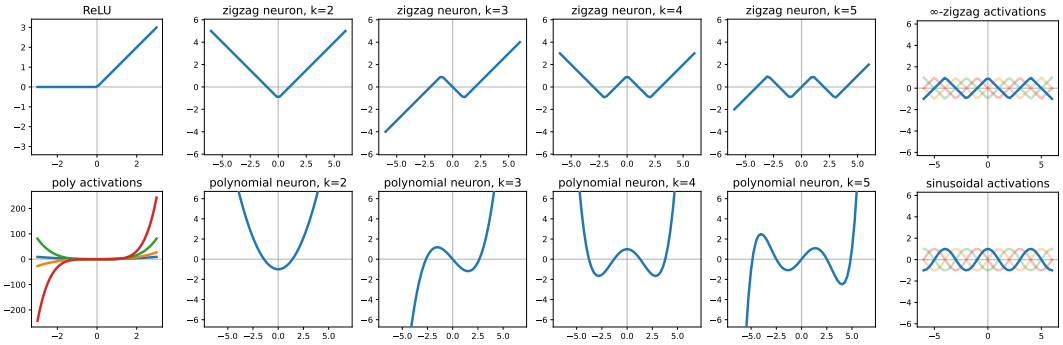



**Figure B.14:** Visualizations of all activation functions considered in this work. As discussed in Section B.4.1, care must be taken to ensure that the architectures can realize sparse parities with the same bias $b$ (in particular, $b = 0$) across varying $k$. Multiple $k$-dependent displacements are shown for the $\infty$-zigzag and sinusoidal activations.

- Degree-$k$ polynomial: $\sigma(z) := z^k$.

In both cases, whenever $r \geq k$ (and, in the case of polynomial activations, choosing the degree to be $k$), there exists a width-$r$ MLP which can represent $k$-sparse parities: for all $(n, k)$ and $|S| = k$, there is a setting of $W, b, u$ such that $f(x; W, b, u) = \chi_S(x)$.

Note that if the output $f(x; \theta)$ is a degree-$k' < k$ polynomial in $x$ (e.g. an MLP with $\sigma(z) = z^{k'}$ activations), the architecture is incapable of representing a parity of $k$ inputs. In fact, it is incapable of representing any function that has a nonzero correlation with parity; this follows from orthogonality (Equation (B.1)).

### Single neurons

To explore the limits of concise parameterization for architectures capable of learning parities, we propose a variety of non-standard activation functions which allow a single neuron to learn sparse parities. These constructions leverage the fact that the parity is a nonlinear function of the sum of its inputs $w_S^\top x$, where $w_S := \sum_{i \in S} e_i$.

*k*-ZIGZAG ACTIVATION. $\sigma(\cdot)$ is the piecewise linear function which interpolates the $k+1$ points $\{(k,+1),(k-2,-1),(k-4,+1),\ldots,(-k+2,\pm 1),(-k,\mp 1)\}$ with $k$ linear regions $\{(-\infty,-k],[-k,-k+2],\ldots,[k-2,k],[k,+\infty)\}$. Then, $\sigma(w_S^\top x)=\chi_S(x)$.

OSCILLATING POLYNOMIAL ACTIVATION. $\sigma(\cdot)$ is the degree-$k$ polynomial which interpolates the same points as above.

$\infty$-ZIGZAG ACTIVATION. The *infinite* extension of the zigzag activation is the *triangle wave* function $\sigma(\cdot)$ which linearly interpolates the infinite set of points $\bigcup_{i\in\mathbb{Z}}\{(2i,+1),(2i+1,-1)\}$. This can express parities of *arbitrary* degree. The $+1$ and $-1$ can be swapped (resulting in an activation which is equivalent when shifted by a bias term). However, in our experiments, we choose the sign convention depending on $k$ such that $\sigma(w_S^\top x) = +1$. This allows different convergence time curves to be more directly comparable across different $k$, since it removes the effects of the bias of the global minimizer alternating with $k$.

SINUSOIDAL ACTIVATION. $\sigma(z) := \sin(z)$. The sinusoidal neuron $\sin(w^\top x+b)$ can also express parities of arbitrary degree, since it can interpolate the same set of points as the $\infty$-zigzag activation. In the experiments, we pick a shift $\beta$ and use the activation $\sigma(z) := \sin(\frac{\pi}{2}z+\beta)$, such that $\sigma(z)$ interpolates the same points as the sign convention selected for the $\infty$-zigzag activation. In the experiments in Section 5.3, the sinusoidal activation is additionally scaled by a factor of 2 ($z\mapsto\sigma(2z)$); this is interchangeable with scaling the learning rate and initialization, and is done to obtain more robust convergence in the particular setting of $(n,k)=(50,3)$.

Figure B.14 visualizes all families of activations considered in this paper.

### Parity Transformer

The Transformer experiments use a slightly simplified version of the architecture introduced in (Vaswani et al., 2017). In particular, it omits dropout, layer normalization, tied input/output embedding weights, and a positional embedding on the special `[CLS]` token. Including these does not change the results significantly; they are all present in the preliminary findings in (Edelman et al., 2022) (in which an "off-the-shelf" Transformer implementation successfully learns sparse parities). We specify the architecture below.

Our *Parity Transformer* has the following hyperparameters: sequence length $n$, token embedding dimension $d_{\text{emb}}$, attention embedding dimension $d_{\text{attn}}$, feedforward embedding dimension $d_{\text{mlp}}$, and number of heads $H$. Its trainable parameters (together denoted by $\theta$) are:

- Token embeddings $E_{-1}, E_{+1}, E_{\texttt{[CLS]}} \in \mathbb{R}^{d_{\text{emb}}}$ and position embeddings $P_1, \ldots, P_n \in \mathbb{R}^{d_{\text{emb}}}$. Let $\theta_{\mathsf{emb}}$ denote this subset of parameters.
- Attention head matrices: $W_Q^{[h]}, W_K^{[h]}, W_V^{[h]} \in \mathbb{R}^{d_{\text{emb}} \times d_{\text{attn}}}$ and $W_{\text{out}}^{[h]} \in \mathbb{R}^{d_{\text{attn}} \times d_{\text{emb}}}$, for $h = 1, \ldots, H$. Let $\theta_{\mathsf{attn}}$ denote this subset of parameters.
- MLP weights and biases: $W_1 \in \mathbb{R}^{d_{\text{mlp}} \times d_{\text{emb}}}, b_1 \in \mathbb{R}^{d_{\text{mlp}}}, W_2 \in \mathbb{R}^{d_{\text{mlp}} \times d_{\text{emb}}}, b_2 \in d_{\text{emb}}$. Let $\theta_{\mathsf{mlp}}$ denote this subset of parameters.
- Classification head: $u \in \mathbb{R}^{d_{\text{emb}}}$.

Then,

$$f_{\mathsf{tf}}(x; \theta) := u^\top \left( f_{\mathsf{mlp}}(f_{\mathsf{attn}}(f_{\mathsf{emb}}(x; \theta_{\mathsf{emb}}); \theta_{\mathsf{attn}}); \theta_{\mathsf{mlp}}) \right),$$

where these submodules are defined by

- Embedding $f_{\mathsf{emb}} : \{\pm 1\}^n \to \mathbb{R}^{(n+1) \times d_{\text{emb}}}$: $f_{\mathsf{emb}}(x; \theta_{\mathsf{emb}})_{i,:} := E_{x_i} + P_i$ for $i \in [n]$. We will include an the extra index `[CLS]`, for which $f_{\mathsf{emb}}(x; \theta_{\mathsf{emb}})_{\texttt{[CLS]},:} := E_{\texttt{[CLS]}}$ (with no positional

embedding). [CLS] stands for "classification", as in "use the output at this position to classify the sequence". This is a standard construction which makes the classifier permutation-invariant.

- Attention block $f_{\mathsf{attn}} : \mathbb{R}^{(n+1)\times d_{\text{emb}}} \to \mathbb{R}^{d_{\text{emb}}}$:

$$f_{\mathsf{attn}}(X; \theta_{\mathsf{attn}}) = X_{[\mathtt{CLS}],:} + \sum_{h=1}^{H} \mathsf{softmax}\left(\frac{1}{\sqrt{d_{\text{attn}}}} X_{[\mathtt{CLS}],:} W_Q^{[h]} (X W_K^{[h]})^\top\right) W_V^{[h]} W_{\text{out}}^{[h]},$$

  where $\mathsf{softmax}(z) := \exp(z)/1^\top \exp(z)$. Note that we have specialized this architecture to a single output, at the [CLS] position.

- MLP $f_{\mathsf{mlp}} : \mathbb{R}^{d_{\text{emb}}} \to \mathbb{R}^{d_{\text{emb}}}$:

$$f_{\mathsf{mlp}}(z; \theta_{\mathsf{mlp}}) := z + W_2 \sigma(W_1 x + b_1) + b_2,$$

  where $\sigma(\cdot) = \mathsf{GeLU}(\cdot)$ (the Gaussian error linear unit) is the standard choice in Transformers.

Training. Each matrix-shaped parameter was initialized using PyTorch's default "Xavier uniform" convention. Unlike the other settings considered in this paper, we were unable to observe successful convergence beyond a few small $(n, k)$ using standard SGD. As is common practice when training Transformers, we used Adam (Kingma & Ba, 2014) with default adaptive parameters $\beta_1 = 0.9, \beta_2 = 0.999, \varepsilon = 10^{-8}$ in our experiments. While there are more fine-grained accounts of why Adam outperforms vanilla SGD (Zhang et al., 2020; Agarwal et al., 2020), finding the optimal optimizer configuration and investigating ablations of this optimizer are outside the scope of this work. In this work, we only tune Adam's learning rate $\eta$.

### PolyNet

For positive integers $n, k$, the PolyNet architecture is parameterized by weights and biases $\theta := \{(w_i \in \mathbb{R}^n, b_i \in \mathbb{R})\}_{i=1}^k$, and is defined by

$$f_{\mathsf{PolyNet}}(x; \theta) := \prod_{i=1}^{k} (w_i^\top x + b_i).$$

Even with all biases $b_i$ set to 0, this architecture can realize a $k$-wise parity, by setting $\{w_i\} = \{e_j : j \in S\}$ in any permutation.

### Details for Figure 5.1 (left)

Figure 5.1 (left) shows training curves from 8 representative configurations, with online i.i.d. samples from the same distribution, corresponding to the $(n = 50, k = 3)$-sparse parity problem. The first row encompasses various MLP settings with standard activations:

- Setting (i): width-10 MLP with ReLU activation ($B = 32, \eta = 0.5$).
- Setting (i): width-10 MLP with ReLU activation, with large batches ($B = 1024, \eta = 0.05$).
- Setting (ii): width-100 MLP with ReLU activation, with tiny batches ($B = 1, \eta = 0.05$).
- Setting (iv): width-10 MLP with polynomial $\sigma(z) = z^3$ activation ($B = 32, \eta = 0.05$).

The second row shows other settings:

- Setting (vii): width-1 MLP with a piecewise linear $k$-zigzag activation ($B = 32, \eta = 0.2$).
- Setting (x): width-1 MLP with a sinusoidal activation (scaled and shifted for $k = 3$; see the discussion in Section B.4.1) ($B = 32, \eta = 0.05$).

- Setting (*iii): Parity Transformer, with $d_{\text{emb}} = 1024, d_{\text{attn}} = 8, H = 128$ ($B = 32, \eta = 5 \times 10^{-4}$).

- Setting (xv): degree-3 PolyNet ($B = 32, \eta = 0.07$).

### Details for Figure B.2

The first row uses the width-10 ReLU MLP configuration (*ii*), holding $B = 32$ and $\eta = 0.1$ while varying the task difficulty across 6 settings: $(n, k) \in \{(30, 3), (60, 3), (90, 3), (30, 4), (30, 5), (30, 6)\}$. The remaining plots are all for the $(50, 3)$ setting.

The second row uses the $k = 3$ PolyNet configuration (*xv*), varying $(B, \eta) \in \{(1, 0.005), (4, 0.01), (16, 0.1), (64, 0.1), (256, 0.1), (1024, 0.1)\}$.

The third row uses the minimally-wide configurations (*i), (*ii), (vii), (viii), (xi), (xii) (thus presenting an example for each non-standard activation), holding batch size $B = 1$. $\eta = 0.1$ in each of the cases except (*ii), where $\eta = 0.01$.

The fourth row uses three large architectures: settings (iii), (vi), and (*iii), with $(B, \eta) \in \{(1, 0.1), (1024, 0.1), (1, 0.001), (1024, 0.01), (32, 0.0003), (1024, 0.0003).\}$ (*iii) uses the Adam optimizer instead of SGD.

### Details for Figure B.6

Figure B.6 contains scaling plots in various settings for the median convergence time $t_c$. Below, we give comprehensive details about these settings. For each of these runs, we chose $B = 32$ (settings with smaller batch sizes exhibited additional variance; with larger batch sizes, the models were slower to converge), as well as the hinge loss. We used SGD with constant learning rate $\eta$ (enumerated below), except in setting (*iii).

The top row shows MLP settings (i) through (vi). From left to right:

- Setting (i): width-10 MLP with ReLU activation ($\eta = 1$).

- Setting (ii): width-100 MLP with ReLU activation ($\eta = 1$).
- Setting (iii): width-1000 MLP with ReLU activation ($\eta = 1$).
- Setting (iv): width-10 MLP with $\sigma(z) = z^k$ activation ($\eta = 0.01$).
- Setting (v): width-100 MLP with $\sigma(z) = z^k$ activation ($\eta = 0.01$).
- Setting (vi): width-1000 MLP with $\sigma(z) = z^k$ activation ($\eta = 0.01$).

The bottom row shows miscellaneous settings. From left to right:

- Setting (vii): width-1 MLP with degree-$k$ oscillating polynomial activation interpolating the parity function ($\eta = 0.01$).
- Setting (xiv): single sinusoidal neuron with no second layer ($\eta = 0.01$).
- Setting (*iii): Parity Transformer, with $d_{\text{emb}} = 1024, d_{\text{attn}} = 8, H = 128$ ($B = 32, \eta = 3 \times 10^{-4}$).
- Setting (iv): degree-$k$ PolyNet ($\eta = 0.05$).
- Setting (ii): width-100 MLP with ReLU activation ($\eta = 1$), showing an expanded range of $n$ for smaller $k$.
- Setting (xv): width-100 MLP with $\sigma(z) = z^k$ activation ($\eta = 1$), showing an expanded range of $n$ for smaller $k$.

### B.4.2 Training curves and convergence time plots

For all example training curves in all figures (in Sections 5.3 and 5.5, as well as the appendix), population losses and accuracies are approximated using a batch of size 8192, sampled once at the beginning of training from the same distribution $\mathcal{D}_S$. All plots of single representative training runs use a

fixed random seed (`torch.manual_seed(0)`); when $R$ training runs are shown, seeds $0, \ldots, R-1$ are used.

In Figures B.3 and B.4, validation accuracies were recorded every 10 iterations, and a run was recorded as converged if it reached $100\%$ accuracy within $10^5$ iterations; we report the $10^{\text{th}}$ percentile over 25 random seeds, to reduce variance arising from the more initialization-sensitive settings. In Figure B.5, coarse-grained scaling estimates for the ($10^{\text{th}}$ percentile) convergence time are computed as follows: for $n \in \mathcal{N} := \{10, 20, 30\}$, the smallest $\alpha$ is chosen such that $t_c \leq c \cdot (n - n_0)^\alpha$, choosing $n_0 = \min \mathcal{N} - 1 = 9$, so that $c = t_c$ at $n = 10$. These estimates are calculated to give quantitative order-of-magnitude upper bounds for the convergence time. Indeed, the power-law convergence times do not extrapolate at a constant learning rate; see Figure 5.2 *(right)*, the "larger $n$" plots in Figure B.6, and the discussion on batch sizes and learning rates in Appendix B.3.2.

To reduce computational load, for the larger-scale probes of convergence times $t_c$, validation accuracies were instead checked on a sample of size 128. For the underparameterized networks (i.e. unable to represent parity, but can still get a meaningful gradient signal), this threshold was changed to 10 consecutive batches with accuracy at least $55\%$. Note that for parity learning in particular, a weak learner can be converted into a strong learner: there is an efficient algorithm (Goldreich & Levin, 1989; Kushilevitz & Mansour, 1993) which, given a classifier which achieves $1/2 + \varepsilon$ accuracy on $\mathcal{D}_S$ for a constant $\varepsilon > 0$, outputs $S$ with high probability.

In the median convergence time plots in Figure 5.1 *(right)*, Figure 5.2 *(right)*, and Figure B.6, error bars for median convergence times in all plots are $95\%$ confidence intervals, computed from 100 bootstrap samples. Each point on the each curve corresponds to 1000 random trials. Halted curves signify more than $50\%$ of runs failing to converge within $T = 10^5$ iterations (hence, infinite medians).

### B.4.3 Implementation, hardware, and compute time

All training experiments were implemented using PyTorch (Paszke et al., 2019).

Although most of the networks considered in the main empirical results are relatively small, a large ($\sim 10^8$) total number of models were trained to certify the "robust space" of results and obtain precise scaling curves. These individual experiments were not large enough to benefit from GPU acceleration; on an internal cluster, the CPU compute expenditure totaled approximately 1500 CPU hours.

A subset of these experiments stood to benefit from GPU acceleration: width $r \geq 100$ MLPs; scaling behaviors for $n \geq 100$; all experiments involving Transformers. These were performed with NVIDIA Tesla P100, Tesla P40, and RTX A6000 GPUs on an internal cluster, consuming a total of approximately 200 GPU hours.

# C

# Pareto Frontiers

## C.1 Additional related work

Learning parities with neural networks. Parities (or XORs) have been shown to be computationally hard to learn for SQ algorithms including gradient-based methods. Various works make additional assumptions to avoid these hardness results to show that neural networks can be efficiently trained to learn parities (Daniely & Malach, 2020; Shi et al., 2021; Frei et al., 2022a;

Malach et al., 2021). More recently, Barak et al. (2022) have focused on understanding how neural networks training on parities without any additional assumption behaves at this computational-statistical limit. They show that one step of gradient descent on a single neuron is able to recover the indices corresponding to the parity with $n^{O(k)}$ samples/computation. Abbe et al. (2023) improve this bound to $O(n^{k-1})$ online SGD steps and generalize the result to handle hierarchical staircases of parity functions which requires a multi-step analysis. Telgarsky (2022) studies the problem of 2-sparse parities with two-layer neural networks trained with vanilla SGD (unlike our restricted two-step training algorithm) and studies the margins achieved post training. They use the margins to get optimal sample complexity $\tilde{O}(n^2/\varepsilon)$ in the NTK regime. Going beyond NTK, they analyze gradient flow (with certain additional modifications) on an exponential wide 2-layer network (making it computationally inefficient) to get the improved sample complexity of $\tilde{O}(n/\varepsilon)$. In contrast to this, our goal is to improve sample complexity while maintaining computational efficiency, using random guessing via the sparse initialization.

Learning single-index/multi-index models over Gaussians with neural networks. Another line of work (Arous et al., 2021; Ba et al., 2022; Damian et al., 2022; Bietti et al., 2022; Damian et al., 2023) has focused on learning functions that depend on a few directions, in particular, single-index and multi-index models over Gaussians using neural nets. These can be thought of as a continuous analog to our sparse parity problem. In a similar analysis (as parities) of online SGD for single index models, Arous et al. (2021) propose the notion of an *information exponent* which captures the initial correlation between the model and the target function, and get convergence results similar to the parity setting with sample complexity $O(n^{k-1})$ for information exponent $k$ (can be thought similar to the $k$ in the parity learning problem). Damian et al. (2023) improve this result by showing that a smoothed version of GD achieves the optimal sample complexity (for CSQ algorithms) of $(n^{k/2})$. Going beyond CSQ algorithms, Chen & Meka (2020) provide a filtered-PCA

algorithm that achieves polynomial dependence on the dimension $n$ in both compute and sample complexity. Note that this is not achievable for CSQ algorithms. For the parity learning problem, the SQ computational lower bounds are $\Omega(n^k)$ (as described in Section 6.3.1).

Empirical inductive biases of large MLPs. Our experiments on tabular benchmarks suggest that wide and sparsely-initialized vanilla MLPs can sometimes close the performance gap between neural networks and decision tree ensemble methods. This corroborates recent findings that vanilla MLPs have strong enough inductive biases to generalize nontrivially in natural data modalities, despite the overparameterization and lack of architectural biases via convolution or recurrence. Notably, many state-of-the-art computer vision models have removed convolutions (Dosovitskiy et al., 2020; Tolstikhin et al., 2021); recently, (Bachmann et al., 2023) demonstrate that even large vanilla NLPs can compete with convolutional models for image classification. Yang et al. (2022a) find monotonic improvements in terms of model width, which are stabilized by their theoretically-motivated hyperparameter scaling rules.

Multi-resource scaling laws for deep learning. Many empirical studies (Kaplan et al., 2020; Henighan et al., 2020; Hoffmann et al., 2022; Zhai et al., 2022), motivated by the pressing need to allocate resources effectively in large-scale deep learning, corroborate the presence and regularity of neural scaling laws. Precise statements and hypotheses vary; Kaplan et al. (2020) fit power-law expressions which predict holdout validation log-perplexity of a language model in terms of dataset size, model size, and training iterations ($m$, $r$, $T$ in our notation). The present work shows how such a joint dependence on $m \times r \times T$ can arise from a single feature learning problem with a computational-statistical gap. Numerous works attempt to demystify neural scaling laws with theoretical models (Bahri et al., 2021; Hutter, 2021; Michaud et al., 2023); ours is unique in that it does not suppose a long-tailed data distribution (the statistical complexity of identifying a sparse parity is

benign). We view these accounts to be mutually compatible: we *do not* purport that statistical query complexity is the unique origin of neural scaling laws, nor that there is a *single* such mechanism.

## C.2 Proofs

### C.2.1 Multi-resource lower bound for sparse parity learning

For some target function $f$ and some parameters $\theta$, we denote the population gradient over the distribution $\mathcal{D}$ by:

$$g(f,\theta) = \mathop{\mathbb{E}}_{\mathbf{x}\sim\mathcal{D}}\left[\nabla_\theta \ell(h_\theta(\mathbf{x}), f(\mathbf{x}))\right]$$

and we denote by $g_i(\cdot,\cdot)$ the gradient w.r.t. the $i$-th coordinate of $\theta$.

Similarly, denote the empirical gradient by:

$$\hat{g}(f,\theta) = \frac{1}{m}\sum_{\mathbf{x}\in\mathcal{S}} \nabla_\theta \ell(h_\theta(\mathbf{x}), f(\mathbf{x}))$$

and $\hat{g}_i(\cdot,\cdot)$ denotes the $i$-th coordinate of the empirical gradient.

**Lemma C.1.** *For every $\theta$ and every $i$ it holds that*

$$\mathop{\mathbb{E}}_{S\sim\binom{n}{k}}\left[\left(g_i(\chi_S,\theta) - \ell_0(h_\theta(\mathbf{x}))\cdot\frac{\partial}{\partial\theta_i}h_\theta(\mathbf{x})\right)^2\right] \leq \frac{1}{\binom{n}{k}}$$

*Proof.* Fix some $i \in [r]$,

$$\begin{aligned}
&\mathop{\mathbb{E}}_{S\sim\binom{n}{k}}\left[\left(g_i(\chi_S,\theta) - \ell_0(h_\theta(\mathbf{x}))\cdot\frac{\partial}{\partial\theta_i}h_\theta(\mathbf{x})\right)^2\right] \\
&= \mathop{\mathbb{E}}_{S\sim\binom{n}{k}}\left[\mathop{\mathbb{E}}_{\mathbf{x}\sim\mathcal{D}}\left[\chi_S(\mathbf{x})\cdot\frac{\partial}{\partial\theta_i}h_\theta(\mathbf{x})\right]^2\right] \\
&= \frac{1}{\binom{n}{k}}\sum_{S\in\binom{n}{k}}\mathop{\mathbb{E}}_{\mathbf{x}\sim\mathcal{D}}\left[\chi_S(\mathbf{x})\cdot\frac{\partial}{\partial\theta_i}h_\theta(\mathbf{x})\right]^2 \le \frac{1}{\binom{n}{k}}
\end{aligned}$$

where the last inequality is from Parseval, using the assumption $\|\nabla h_\theta(\mathbf{x})\|_\infty \le 1$.

□

*Proof of Proposition 6.3.* Using the Lemma C.1 we get:

$$\begin{aligned}
&\mathop{\mathbb{E}}_{S\sim\binom{n}{k}}\left[\max_{i,t}\left(g_i(\chi_S,\theta_t^*) - \ell_0(h_{\theta_t^*}(\mathbf{x}))\cdot\frac{\partial}{\partial\theta_i}h_{\theta_t^*}(\mathbf{x})\right)^2\right] \\
&\le \mathop{\mathbb{E}}_{S\sim\binom{n}{k}}\left[\sum_{i=1}^{r}\sum_{t=1}^{T}\left(g_i(\chi_S,\theta_t^*) - \ell_0(h_{\theta_t^*}(\mathbf{x}))\cdot\frac{\partial}{\partial\theta_i}h_{\theta_t^*}(\mathbf{x})\right)^2\right] \\
&\le \frac{rT}{\binom{n}{k}} \le \delta\tau^2
\end{aligned}$$

Therefore, taking expectation over the choice of $\theta_0$

$$\begin{aligned}
&\mathop{\mathbb{E}}_{\theta_0}\mathop{\mathbb{E}}_{S\sim\binom{n}{k}}\left[\max_{i,t}\left(g_i(\chi_S,\theta_t^*) - \ell_0(h_{\theta_t^*}(\mathbf{x}))\cdot\frac{\partial}{\partial\theta_i}h_{\theta_t^*}(\mathbf{x})\right)^2\right] \\
&= \mathop{\mathbb{E}}_{S\sim\binom{n}{k}}\mathop{\mathbb{E}}_{\theta_0}\left[\max_{i,t}\left(g_i(\chi_S,\theta_t^*) - \ell_0(h_{\theta_t^*}(\mathbf{x}))\cdot\frac{\partial}{\partial\theta_i}h_{\theta_t^*}(\mathbf{x})\right)^2\right] \le \delta\tau^2/2
\end{aligned}$$

So, there exists some $S \in \binom{n}{k}$ s.t.

$$\mathop{\mathbb{E}}_{\theta_0}\left[\max_{i,t}\left(g_i(\chi_S,\theta_t^*) - \ell_0(h_{\theta_t^*}(\mathbf{x}))\cdot\frac{\partial}{\partial\theta_i}h_{\theta_t^*}(\mathbf{x})\right)^2\right] \le \delta\tau^2/2$$

Observe the noise variable $\xi_t = \ell_0(h_{\theta_t^*}(\mathbf{x})) \cdot \frac{\partial}{\partial \theta_i} h_{\theta_t^*}(\mathbf{x}) - \hat{g}_i(\chi_S, \theta_t^*, \mathcal{S}_t)$. From Markov's inequality and Assumption 6.2, with probability at least $1 - \delta$ over the choice of $\theta_0$, for all $t \leq T$ and $i \in [r]$:

$$\left| \hat{g}_i(\chi_S, \theta_t^*, \mathcal{S}_t) - \ell_0(h_{\theta_t^*}(\mathbf{x})) \cdot \frac{\partial}{\partial \theta_i} h_{\theta_t^*}(\mathbf{x}) \right| \leq \tau$$

therefore, we get that $\xi_1, \ldots, \xi_T \in [-\tau, \tau]^r$ (i.e., this is a valid choice of adversarial noise variables), and SGD follows the trajectory $\theta_1^\star, \ldots, \theta_T^\star$. □

### C.2.2 Feature selection with an over-sparse initialization and a wide network

#### Warmup: existence of good subnetworks

Let $h_{\mathbf{w}}(\mathbf{x}) = \sigma(\langle \mathbf{w}, \mathbf{x} \rangle)$ be a single ReLU neuron, where $\sigma(x) = \max\{x, 0\}$. Fix some $4k < s \leq n$. Assume we initialize $\mathbf{w} \in \{0, 1\}^n$ by randomly choosing $s$ coordinates and setting them to 1, and setting the rest to zero. Fix some subset $S \subseteq \binom{n}{k}$. We say that $\mathbf{w}$ is a *good* neuron if $S \subseteq \mathbf{w}$. We say that $\mathbf{w}$ is a *bad* neuron if it is not a *good* neuron.

**Lemma C.2.** *With probability at least $(s/2n)^k$ over the choice of $\mathbf{w}$, $\mathbf{w}$ is a* good *neuron.*

*Proof.* There are $\binom{n}{s}$ choices for $\mathbf{w}$, and there are $\binom{n-k}{s-k}$ *good* choices for $\mathbf{w}$. Observe that:

$$\binom{n}{s} = \frac{n(n-1)\cdots(n-k+1)}{s(s-1)\cdots(s-k+1)} \binom{n-k}{s-k} \leq \left(\frac{2n}{s}\right)^k \binom{n-k}{s-k}$$

and therefore the required follows. □

**Lemma C.3.** *Assume $k$ is even, $s$ is odd and $k < \frac{s}{4}$. There exist constants $C_k, c_k$ s.t. if $\mathbf{w}$ is a* good *neuron, then:*

1. *For all $i \in S$,*

$$\mathop{\mathbb{E}}_{\mathbf{x}} \left[ \frac{\partial}{\partial w_i} h_{\mathbf{w}}(\mathbf{x}) \cdot \chi_S(\mathbf{x}) \right] = C_{k,s}$$

2. *For all $i \notin S$,*

$$\mathop{\mathbb{E}}_{\mathbf{x}}\left[\frac{\partial}{\partial w_i} h_{\mathbf{w}}(\mathbf{x}) \cdot \chi_S(\mathbf{x})\right] = w_i c_{k,s}$$

*and furthermore, there exists a constant $\kappa_k$ s.t. $\left|C_{k,s}\right| > \kappa_k \binom{s}{k-1}^{-1/2}$ and $\frac{\left|c_{k,s}\right|}{\left|C_{k,s}\right|} \leq \frac{4k}{s}$.*

*Proof.* First, consider the case where $i \in S$. Therefore,

$$\begin{aligned}
\mathop{\mathbb{E}}_{\mathbf{x}\sim\{\pm 1\}^n}\left[\frac{\partial}{\partial w_i} h_{\mathbf{w}}(\mathbf{x}) \cdot \chi_S(\mathbf{x})\right] &= \mathop{\mathbb{E}}_{\mathbf{x}\sim\{\pm 1\}^n}\left[\sigma'(\langle \mathbf{w}, \mathbf{x}\rangle) \cdot x_i \cdot \chi_S(\mathbf{x})\right] \\
&= \mathop{\mathbb{E}}_{\mathbf{x}\sim\{\pm 1\}^n}\left[\mathbb{1}_{\left\{\sum_{j\in\mathbf{w}} x_j \geq 0\right\}} \chi_{S\setminus\{i\}}(\mathbf{x})\right] \\
&= \mathop{\mathbb{E}}_{\mathbf{x}\sim\{\pm 1\}^s}\left[\left(\frac{1}{2}\mathrm{Maj}_s(\mathbf{x}) + \frac{1}{2}\right)\chi_{S\setminus\{i\}}(\mathbf{x})\right] \\
&= \frac{1}{2}\widehat{\mathrm{Maj}_s}(S \setminus \{i\})
\end{aligned}$$

where $S \setminus \{i\}$ is interpreted as a subset of $[s]$. From symmetry of the Majority function, all Fourier coefficients of the same order are equal. Therefore, the first condition holds for $C_{k,s} = \frac{1}{2}\widehat{\mathrm{Maj}_s}(k-1)$, where $\widehat{\mathrm{Maj}_s}(k-1)$ denotes the $(k-1)$-th order Fourier coefficient.

When $i \in S \setminus \mathbf{w}$ we get:

$$\begin{aligned}
\mathop{\mathbb{E}}_{\mathbf{x}\sim\{\pm 1\}^n}\left[\frac{\partial}{\partial w_i} h_{\mathbf{w}}(\mathbf{x}) \cdot \chi_S(\mathbf{x})\right] &= \mathop{\mathbb{E}}_{\mathbf{x}\sim\{\pm 1\}^n}\left[\sigma'(\langle \mathbf{w}, \mathbf{x}\rangle) \cdot x_i \cdot \chi_S(\mathbf{x})\right] \\
&= \mathop{\mathbb{E}}_{\mathbf{x}\sim\{\pm 1\}^n}\left[\mathbb{1}_{\left\{\sum_{j\in\mathbf{w}} x_j > 0\right\}} \chi_{S\cup\{i\}}(\mathbf{x})\right] \\
&= \mathop{\mathbb{E}}_{\mathbf{x}\sim\{\pm 1\}^s}\left[\left(\frac{1}{2}\mathrm{Maj}_s(\mathbf{x}) + \frac{1}{2}\right)\chi_{S\cup\{i\}}(\mathbf{x})\right] \\
&= \frac{1}{2}\widehat{\mathrm{Maj}_s}(S \cup \{i\})
\end{aligned}$$

Finally, when $i \notin S$ we get:

$$\begin{aligned}\mathop{\mathbb{E}}_{\mathbf{x}\sim\{\pm1\}^n}\left[\frac{\partial}{\partial w_i}h_{\mathbf{w}}(\mathbf{x})\cdot\chi_S(\mathbf{x})\right] &= \mathop{\mathbb{E}}_{\mathbf{x}\sim\{\pm1\}^n}\left[\sigma'(\langle\mathbf{w},\mathbf{x}\rangle)\cdot x_i\cdot\chi_S(\mathbf{x})\right] \\ &= \mathbb{E}\left[\sigma'(\langle\mathbf{w},\mathbf{x}\rangle)\cdot\chi_S(\mathbf{x})\right]\mathop{\mathbb{E}}_{x_i}[x_i] = 0\end{aligned}$$

Therefore, the second condition holds with $c_k = \frac{1}{2}\widehat{\mathrm{Maj}_s}(k+1)$.

Now, from Theorem 5.22 in O'Donnell (2014) we have:

$$\binom{s}{k-1}\cdot\widehat{\mathrm{Maj}_s}(k-1)^2 = \sum_{S'\in\binom{s}{k-1}}\widehat{\mathrm{Maj}_s}(S')^2 \geq \rho(k-1)$$

where $\rho(v) = \frac{2}{\pi v 2^v}\binom{v-1}{\frac{v-1}{2}}$. So, we get $\left|C_{k,s}\right| \geq \frac{1}{2}\sqrt{\rho(k-1)}\binom{s}{k-1}^{-1/2}$. Using the same Theorem, we also have:

$$\binom{s}{k+1}\cdot\widehat{\mathrm{Maj}_s}(k+1)^2 = \sum_{S'\in\binom{s}{k+1}}\widehat{\mathrm{Maj}_s}(S')^2 \leq 2\rho(k+1) < 2\rho(k-1)$$

So, we get:

$$\begin{aligned}\frac{\left|c_{k,s}\right|}{\left|C_{k,s}\right|} &\leq \frac{\sqrt{2\rho(k-1)}\binom{s}{k+1}^{-1/2}}{\sqrt{\rho(k-1)}\binom{s}{k-1}^{-1/2}} = \sqrt{2}\sqrt{\frac{\binom{s}{k+1}}{\binom{s}{k-1}}} \\ &= \sqrt{\frac{2k(k+1)}{(s-k+2)(s-k+1)}} \leq \sqrt{\frac{4k^2}{(1/4)s^2}} = 4\frac{k}{s}\end{aligned}$$

□

**Lemma C.4.** *Assume that $k < \frac{s}{4}$. If $\mathbf{w}$ is a* bad *neuron, then:*

$$\left\|\mathop{\mathbb{E}}_{\mathbf{x}}\left[\frac{\partial}{\partial\mathbf{w}}h_{\mathbf{w}}(\mathbf{x})\cdot\chi_S(\mathbf{x})\right]\right\|_1 \leq C_{k,s}$$

*Proof.* First, assume that $|S \setminus \mathbf{w}| \geq 2$. In this case, there exist $i, i' \in S$ s.t. $w_i = w_{i'} = 0$ and $i \neq i'$. Fix some $j \in [n]$, and choose some $j' \in i, i'$ s.t. $j' \neq j$.

$$\begin{aligned}
\mathop{\mathbb{E}}_{\mathbf{x}\sim\{\pm 1\}^n}\left[\frac{\partial}{\partial w_j} h_{\mathbf{w}}(\mathbf{x}) \cdot \chi_S(\mathbf{x})\right] &= \mathop{\mathbb{E}}_{\mathbf{x}\sim\{\pm 1\}^n}\left[\sigma'(\langle \mathbf{w}, \mathbf{x}\rangle) \cdot \chi_S(\mathbf{x}) \cdot x_j\right] \\
&= \mathop{\mathbb{E}}_{\mathbf{x}\sim\{\pm 1\}^n}\left[\sigma'(\langle \mathbf{w}, \mathbf{x}\rangle) \cdot \chi_{S\setminus\{j'\}}(\mathbf{x}) \cdot x_j x_{j'}\right] \\
&= \mathop{\mathbb{E}}_{x_j'}\left[x_{j'}\right] \cdot \mathop{\mathbb{E}}_{\mathbf{x}_{[n]\setminus\{j'\}}}\left[\sigma'(\langle \mathbf{w}, \mathbf{x}\rangle) \cdot \chi_{S\setminus\{j'\}}(\mathbf{x}) \cdot x_j\right] = 0
\end{aligned}$$

and this gives the required.

Now, assume that $S \setminus \mathbf{w} = \{i\}$ for some index $i$. For every $j \neq i$, similarly to the previous analysis, we have:

$$\mathop{\mathbb{E}}_{\mathbf{x}\sim\{\pm 1\}^n}\left[\frac{\partial}{\partial w_j} h_{\mathbf{w}}(\mathbf{x}) \cdot \chi_S(\mathbf{x})\right] = 0$$

Finally, similarly to the proof of Lemma C.3, we have:

$$\mathop{\mathbb{E}}_{\mathbf{x}\sim\{\pm 1\}^n}\left[\frac{\partial}{\partial w_i} h_{\mathbf{w}}(\mathbf{x}) \cdot \chi_S(\mathbf{x})\right] = \frac{1}{2}\widehat{\mathrm{Maj}_s}(S \setminus \{i\}) = C_{k,s}$$

and so we get the required. □

### End-to-end result

We train the following network:

$$f_{\mathbf{W},\mathbf{b},\mathbf{u},\beta}(\mathbf{x}) = \sum_{i=1}^{r} u_i \sigma\left(\langle \mathbf{w}_i, \mathbf{x}\rangle + b_i\right) + \beta$$

We fix some $k \leq s < n$ and initialize the network as follows:

- Randomly initialize $\mathbf{w}_1, \ldots, \mathbf{w}_{r/2} \in \{0,1\}^n$ s.t. $\|\mathbf{w}_i\|_1 = s$ (i.e., each $\mathbf{w}_i$ has $s$ active coordi-

nates), with a uniform distribution over all $\binom{n}{s}$ subsets.

- Randomly initialize $b_1, \ldots, b_{r/2} \sim \left\{\beta_1, \ldots, \beta_{k/2-1}\right\}$ where $\beta_i = \frac{1}{2k}\left(-k + 2i + 1/16\right)$.
- Randomly initialize $u_1, \ldots, u_{r/2} \sim \{\pm 1\}$ uniformly at random.
- Initialize $\mathbf{w}_{r/2+1}, \ldots, \mathbf{w}_r, b_{r/2+1}, \ldots, b_r$ s.t. $\mathbf{w}_i = \mathbf{w}_{i-r/2}$ and $b_i = b_{i-r/2}$ (symmetric initialization).
- Initialize $u_{r/2+1}, \ldots, u_r$ s.t. $u_i = -u_{i-r/2}$.
- Initialize $\beta = 0$

Let $\ell(\hat{y}, y) = \max(1 - y\hat{y}, 0)$ be the hinge-loss function. Given some distribution $\mathcal{D}$ over $\mathcal{X} \times \{\pm 1\}$, define the loss of $f$ over the distribution by:

$$L_{\mathcal{D}}(f) = \mathop{\mathbb{E}}_{(\mathbf{x},y)\sim\mathcal{D}}\left[\ell(f(\mathbf{x}), y)\right]$$

Similarly, given a sample $\mathcal{S} \subseteq \mathcal{X} \times \{\pm 1\}$, define the loss of $f$ on the sample by:

$$L_{\mathcal{S}}(f) = \frac{1}{|\mathcal{S}|}\sum_{(\mathbf{x},y)\in\mathcal{S}} \ell(f(\mathbf{x}), y)$$

We train the network by gradient descent on a sample $\mathcal{S}$ with $\ell_2$ regularization (weight decay):

$$\theta^{(t+1)} = (1 - \lambda^{(t)})\theta^{(t)} - \eta^{(t)}\nabla L_{\mathcal{S}}(f_{\theta^{(t)}})$$

We allow choosing learning rate $\eta$, the weight decay $\lambda$ differently for each layer, separately for the weights and biases, and for each iteration.

**Lemma C.5.** *Fix some $\tau > 0, \delta > 0$. Let $\mathcal{S}$ be a set of $m$ examples chosen i.i.d. from $\mathcal{D}$. Then, if $m \geq \frac{4\log(4nr/\delta)}{\tau^2}$, with probability at least $1-\delta$ over the choice of $\mathcal{S}$, it holds that:*

$$\left\|\nabla_{\mathbf{W},\mathbf{b}} L_{\mathcal{D}}(f_{\mathbf{W},\mathbf{b},\mathbf{u},\beta}) - \nabla_{\mathbf{W},\mathbf{b}} L_{\mathcal{S}}(f_{\mathbf{W},\mathbf{b},\mathbf{u},\beta})\right\|_\infty \leq \tau$$

*Proof.* Denote by $\theta \in \mathbb{R}^{nr+r}$ the set of all parameters in $\mathbf{W}, \mathbf{b}$. For every $i \in [nr+r]$, from Hoeffding's inequality, we have:

$$\Pr\left(\left|\frac{\partial}{\partial\theta_i} L_{\mathcal{D}}(f_\theta) - \frac{\partial}{\partial\theta_i} L_{\mathcal{S}}(f_\theta)\right| \geq \tau\right) \leq 2\exp(-m\tau^2/4) \leq \frac{\delta}{2nr}$$

and the required follows from the union bound. □

Given some initialization of $\mathbf{W}, \mathbf{b}$, for every $j$ denote by $I_j \subseteq [r/2]$ the set of indices of neurons with *good* weights and bias equal to $\beta_j$. We say that an initialization is $r'$*-good* if for all $j$ we have $r'/2 \leq |I_j| \leq 2r'$.

Let $g$ be some vector-valued function. We define:

$$\|g\|_{\infty,2} = \sup_x \|g(x)\|_2$$

For some mapping $\psi : \mathcal{X} \to \mathbb{R}^r$ and some $B > 0$, denote by $\mathcal{H}_{\psi,B}$ the class of linear functions of norm at most $B$ over the mapping $\psi$:

$$\mathcal{H}_{\psi,B} = \{h_{\psi,\mathbf{u}} : \|\mathbf{u}\|_2 \leq B\}$$

where $h_{\psi,\mathbf{u}}(\mathbf{x}) = \langle\psi(\mathbf{x}), \mathbf{u}\rangle$.

Denote by $\varphi^{(t)}$ the output of the first layer of $f_{\mathbf{W},\mathbf{b},\mathbf{u},\beta}$ after $t$ iterations of GD*.

*We assume 1 is appended to the vector for allowing bias

**Lemma C.6.** *Fix some $\delta > 0$. Fix some $r'$-*good *initialization $\mathbf{W}, \mathbf{b}$. Let $\eta = \frac{1}{2k}\left|C_{k,s}\right|^{-1}$, and let $\tau \leq \frac{1}{\eta 16kn}$. Assume that $m \geq \frac{4\log(4nr/\delta)}{\tau^2}$. Then, there exists some mapping $\psi$ s.t. the following holds:*

1. $\|\psi\|_{\infty,2} \leq \sqrt{8kr'}$
2. *There exists $\mathbf{u}^\star$ s.t. $L_{\mathcal{D}}(h_{\psi,\mathbf{u}^\star}) \leq \frac{\sqrt{4kr'}B_k}{\sqrt{s}}$, with $\|\mathbf{u}^\star\|_2 \leq B_k$ for some constant $B_k$.*
3. *With probability at least $1-\delta$ over the choice of $\mathcal{S} \sim \mathcal{D}^m$, we have*

$$\left\|\psi - \varphi^{(1)}\right\|_{\infty,2} \leq 4kr'n\eta\tau$$

*Proof.* We construct two mappings $\psi, \psi^\star$ as follows.

- We will denote $\psi_0(\mathbf{x}) = \psi_0^\star(\mathbf{x}) = 1$ to allow a bias term.
- For every $i$, if $\mathbf{w}_i$ is a bad neuron, we set $\psi_i(\mathbf{x}) = \psi_i^\star(\mathbf{x}) = 0$.
- For every good neuron $\mathbf{w}_i$ s.t. $i \in I_j$, we set:
  - $\psi_i^\star(\mathbf{x}) = \sigma\left(\eta C_{k,s}\sum_{j'\in S} x_{j'} + \beta_j\right)$
  - $\psi_i(\mathbf{x}) = \sigma\left(\eta C_{k,s}\sum_{j'\in S} x_{j'} + \eta c_{k,s}\sum_{j'\in \mathbf{w}_i\setminus S} x_{j'} + \beta_j\right)$
  - $\psi_{i+r/2}^\star(\mathbf{x}) = \sigma\left(-\eta C_{k,s}x_{j'} + \beta_j\right)$
  - $\psi_{i+r/2}(\mathbf{x}) = \sigma\left(-\eta C_{k,s}\sum_{j'\in S} x_{j'} + \eta c_{k,s}\sum_{j'\in \mathbf{w}_i\setminus S} x_{j'} + \beta_j\right)$

We will show that $\psi^\star$ achieves loss zero, and that $\psi$ approximates it. We assume $C_{k,s} > 0$ and the case of $C_{k,s} < 0$ is derived similarly.

First, notice that from Lemma C.3,

$$\left|\eta c_{k,s}\right| = \frac{1}{2k}\frac{\left|c_{k,s}\right|}{\left|C_{k,s}\right|} \leq \frac{2}{s}$$

**Claim:** for every $\mathbf{u}$, $\left|L_{\mathcal{D}}(h_{\psi,\mathbf{u}}) - L_{\mathcal{D}}(h_{\psi^\star,\mathbf{u}})\right| \leq \frac{\|\mathbf{u}\|\sqrt{4kr'}}{\sqrt{s}}$

**Proof:** Observe that

$$\begin{aligned}
\left|L_{\mathcal{D}}(h_{\psi,\mathbf{u}}) - L_{\mathcal{D}}(h_{\psi^\star,\mathbf{u}})\right| &\leq \mathop{\mathbb{E}}_{\mathcal{D}}\left[\left|\ell(h_{\psi,\mathbf{u}}(\mathbf{x}),y) - \ell(h_{\psi^\star,\mathbf{u}}(\mathbf{x}),y)\right|\right] \leq \mathop{\mathbb{E}}_{\mathcal{D}}\left[\left|h_{\psi^\star,\mathbf{u}}(\mathbf{x}) - h_{\psi,\mathbf{u}}(\mathbf{x})\right|\right] \\
&= \mathop{\mathbb{E}}_{\mathcal{D}}\left[\left|\langle \psi(\mathbf{x}) - \psi^\star(\mathbf{x}), \mathbf{u}\rangle\right|\right] \overset{\text{C.S}}{\leq} \|\mathbf{u}\| \mathop{\mathbb{E}}_{\mathcal{D}}\left[\|\psi(\mathbf{x})^\star - \psi(\mathbf{x})\|\right] \\
&\overset{\text{Jensen}}{\leq} \|\mathbf{u}\| \sqrt{\mathop{\mathbb{E}}_{\mathcal{D}}\left[\|\psi(\mathbf{x})^\star - \psi(\mathbf{x})\|^2\right]} = \|\mathbf{u}\| \sqrt{\sum_i \mathop{\mathbb{E}}_{\mathcal{D}}\left[(\psi_i^\star(\mathbf{x}) - \psi_i(\mathbf{x}))^2\right]} \\
&\leq \|\mathbf{u}\| \sqrt{\sum_{i \text{ is good}} \mathop{\mathbb{E}}_{\mathcal{D}}\left[\left(\eta c_{k,s} \sum_{j' \in \mathbf{w}_i \setminus S} x_{j'}\right)^2\right]} \leq \|\mathbf{u}\| \sqrt{4kr'} \left|\eta c_{k,s}\right| \sqrt{s} \\
&\leq \frac{\|\mathbf{u}\| \sqrt{4kr'}}{\sqrt{s}}
\end{aligned}$$

**Claim:** There exists $\mathbf{u}^\star$ with norm $\|\mathbf{u}^\star\| \leq B_k$ and $L_{\mathcal{D}}(h_{\psi^\star,\mathbf{u}^\star}) = 0$.

**Proof:** For every $\mathbf{x}$, denote $s_{\mathbf{x}} = \sum_{j \in S} x_j$, and observe that $s_{\mathbf{x}} \in \{-k, -k+2, \ldots, k-2, k\} =: \mathcal{S}$ and $\chi_S(\mathbf{x}) = s_{\mathbf{x}} \mod 2$. For every $j$, denote $v_j^+(s) = \sigma(\frac{1}{2k}s + \beta_j)$ and $v_j^-(s) = \sigma(-\frac{1}{2k}s + b_j)$. Then, for every $s \in \mathcal{S}$ denote $\mathbf{v}(s) = (1, v_1^+(s), \ldots, v_{k/2-1}^+, v_1^-(s), \ldots, v_{k/2-1}^-(s)) \in \mathbb{R}^k$. Observe that $V = \{\mathbf{v}(s)\}_{s \in \mathcal{S}} \in \mathbb{R}^{k \times k}$ has linearly independent rows, and therefore there exists $\nu = (\nu_0, \nu_1^+, \ldots, \nu_{k/2-1}^+, \nu_1^-, \ldots, \nu_{k/2-1}^-) \in \mathbb{R}^k$ s.t. $\mathbf{v}(s)^\top \nu = s \mod 2$. Now, define $\mathbf{u}^\star$ s.t. for every bad $i$ we set $u_i^\star = 0$, and for every $i \in I_j$ we set $u_i^\star = \frac{1}{|I_j|}\nu_j^+$ and $u_{i+r/2,-}^\star = \frac{1}{|I_j|}\nu_j^-$, and set $u_0^\star = \nu_0$. Observe that for every $\mathbf{x} \in \mathcal{X}$ we have:

$$\langle \psi(\mathbf{x}), \mathbf{u}^\star \rangle = \nu_0 + \sum_j \frac{1}{|I_j|} \sum_{i \in I_j} (v_j^+(s_{\mathbf{x}})\nu_j^+ + v_j^-(s_{\mathbf{x}})\nu_j^-) = \left\langle \mathbf{v}(s_{\mathbf{x}})^\top \nu \right\rangle = s_{\mathbf{x}} \mod 2 = \chi_S(\mathbf{x})$$

and therefore $L_{\mathcal{D}}(h_{\psi,\mathbf{u}^\star}) = 0$. Additionally, observe that

$$\|\mathbf{u}^\star\|^2 = \nu_0^2 + \sum_j \frac{1}{|I_j|^2}((\nu_j^+)^2 + (\nu_j^-)^2) \leq \|\nu\|_2^2$$

Now we prove the statements in the main lemma:

1. For every $\mathbf{x} \in \mathcal{X}$ we have

$$\|\psi(\mathbf{x})\|_2^2 = \sum_j \sum_{i \in I_j} (\psi_i(\mathbf{x})^2 + \psi_{i+r/2}(\mathbf{x})^2) \leq 4 \sum_j |I_j| \leq 8kr'$$

2. Follows from the two previous claims.

3. Assume we choose $\lambda = 1$ for the weights of the first layer, $\lambda = 0$ for the biases of the first layer, $\eta = \frac{1}{2k} \left|C_{k,s}\right|^{-1}$ for the weights of the first layer, and $\eta = 0$ for all other parameters. From Lemma C.5, w.p. at least $1 - \delta$ we have:

$$\left\|\nabla_{\mathbf{W},\mathbf{b}} L_{\mathcal{D}}(f_{\mathbf{W},\mathbf{b},\mathbf{u},\beta}) - \nabla_{\mathbf{W},\mathbf{b}} L_{\mathcal{S}}(f_{\mathbf{W},\mathbf{b},\mathbf{u},\beta})\right\|_\infty \leq \tau$$

Denote by $\mathbf{w}_i^{(1)}$ the $i$-th weight after the first gradient step, and denote $\mathbf{w}_i^\star := -\eta \nabla_{\mathbf{w}_i} L_{\mathcal{D}}(f_{\mathbf{W},\mathbf{b},\mathbf{u},\beta})$ and $\widehat{\mathbf{w}}_i := -\eta \nabla_{\mathbf{w}_i} L_{\mathcal{S}}(f_{\mathbf{W},\mathbf{b},\mathbf{u},\beta})$. By the choice of $\lambda$, we get $\mathbf{w}_i^{(1)} = \widehat{\mathbf{w}}_i$. Observe that for all $i$:

$$\|\widehat{\mathbf{w}}_i - \mathbf{w}_i^\star\|_\infty = \eta \left\|\nabla_{\mathbf{w}_i} L_{\mathcal{S}}(f_{\mathbf{W},\mathbf{b},\mathbf{u},\beta}) - \nabla_{\mathbf{w}_i} L_{\mathcal{D}}(f_{\mathbf{W},\mathbf{b},\mathbf{u},\beta})\right\| \leq \eta\tau$$

**Claim**: For all $i, j$, if $w_{i,j}^\star = 0$, then $\left|w_{i,j}^\star - w_{i,j}^{(1)}\right| \leq \frac{1}{16kn}$.

**Proof**: We have $\left|\widehat{w}_{i,j}\right| \leq \eta\tau$, and the claim follows from the fact that $\eta\tau \leq \frac{1}{16kn}$.

First, consider the case where $\mathbf{w}_i$ is a *bad* neuron. In this case, by Lemma C.4, we have $\|\mathbf{w}_i^\star\|_1 \leq \frac{1}{2k}$ and $\|\mathbf{w}_i^\star\|_0 \leq 1$, and from the previous claim we get $\left\|\mathbf{w}_i^{(1)} - \mathbf{w}_i^\star\right\|_1 \leq \eta\tau + \frac{1}{16k}$. Therefore, for all $\mathbf{x} \in \mathcal{X}$ we get:

$$\left|\left\langle \mathbf{w}_i^{(1)}, \mathbf{x} \right\rangle\right| = \left|\left\langle \mathbf{w}_i^{(1)} - \mathbf{w}_i^\star + \mathbf{w}_i^\star, \mathbf{x} \right\rangle\right| \leq \left(\|\mathbf{w}_i^\star\|_1 + \left\|\mathbf{w}_i^\star - \mathbf{w}_i^{(1)}\right\|_1\right) \|\mathbf{x}\|_\infty \leq \frac{10}{16k} + \eta\tau$$

Since at initialization we have $b_i \leq -\frac{15}{16k}$, and we have $\eta\tau \leq \frac{1}{16k}$, for all $\mathbf{x} \in \mathcal{X}$:

$$\sigma\left(\left\langle \mathbf{w}_i^{(1)}, \mathbf{x} \right\rangle + b_i\right) = 0 = \psi_i(\mathbf{x})$$

Now, assume that $\mathbf{w}_i$ is a *good* neuron. In this case, by Lemma C.3, we have

$$\mathbf{w}_{i,j}^{\star} = \begin{cases} \eta C_{k,s} & j \in S \\ w_{i,j}\eta c_{k,s} & j \notin S \end{cases}$$

Observe that $\psi_i(\mathbf{x}) = \sigma(\langle \mathbf{w}_i^{\star}, \mathbf{x} \rangle + b_i)$, and therefore:

$$\left|\psi_i(\mathbf{x}) - \sigma\left(\left\langle \mathbf{w}_i^{(1)}, \mathbf{x} \right\rangle + b_i\right)\right| \leq \left|\left\langle \mathbf{w}_i^{(1)} - \mathbf{w}_i^{\star}, \mathbf{x} \right\rangle\right| \leq \left\|\mathbf{w}_i^{(1)} - \mathbf{w}_i^{\star}\right\|_1 \leq n\eta\tau$$

Similarly, in this case we will get $\left|\psi_{i+r/2}(\mathbf{x}) - \sigma\left(\left\langle \mathbf{w}_{i+r/2}^{(1)}, \mathbf{x} \right\rangle + b_{i+r/2}\right)\right| \leq n\eta\tau$. Now the required follows from all we showed.

□

**Lemma C.7.** *Fix some mappings $\psi, \psi'$ and some $\mathbf{w}$. Then, for for every distribution $\mathcal{D}$:*

$$\left|L_{\mathcal{D}}(h_{\psi,\mathbf{w}}) - L_{\mathcal{D}}(h_{\psi',\mathbf{w}})\right| \leq \|\mathbf{w}\| \left\|\psi - \psi'\right\|_{\infty,2}$$

*and for every sample $\mathcal{S}$:*

$$\left|L_{\mathcal{S}}(h_{\psi,\mathbf{w}}) - L_{\mathcal{S}}(h_{\psi',\mathbf{w}})\right| \leq \|\mathbf{w}\| \left\|\psi - \psi'\right\|_{\infty,2}$$

*Proof.* Observe that, since $\ell$ is 1-Lipschitz:

$$\begin{aligned}
|L_{\mathcal{D}}(h_{\psi',\mathbf{w}}) - L_{\mathcal{D}}(h_{\psi,\mathbf{w}})| &\leq \mathop{\mathbb{E}}_{(\mathbf{x},y)\sim\mathcal{D}} \left[\left|\ell(h_{\psi',\mathbf{w}}(\mathbf{x}),y) - \ell(h_{\psi',\mathbf{w}}(\mathbf{x}),y)\right|\right] \\
&\leq \mathop{\mathbb{E}}_{\mathcal{D}} \left[\left|h_{\psi',\mathbf{w}}(\mathbf{x}) - h_{\psi,\mathbf{w}}(\mathbf{x})\right|\right] \leq \mathop{\mathbb{E}}_{\mathcal{D}} \left[\left|\left\langle \psi'(\mathbf{x}) - \psi(\mathbf{x}), \mathbf{w}\right\rangle\right|\right] \\
&\leq \mathop{\mathbb{E}}_{\mathcal{D}} \left[\left\|\psi'(\mathbf{x}) - \psi(\mathbf{x})\right\| \|\mathbf{w}\|\right] \leq \|\mathbf{w}\| \left\|\psi - \psi'\right\|_{\infty,2}
\end{aligned}$$

and similarly we get:

$$\left|L_{\mathcal{S}}(h_{\psi',\mathbf{w}}) - L_{\mathcal{S}}(h_{\psi,\mathbf{w}})\right| \leq \|\mathbf{w}\| \left\|\psi - \psi'\right\|_{\infty,2}$$

□

**Lemma C.8.** *Fix some mapping $\psi$, and let $\mathcal{S}$ be a sample of size $m$ sampled i.i.d. from $\mathcal{D}$. Then, with probability at least $1-\delta$ over the choice of $\mathcal{S}$, for every $\psi'$ and for every $h \in \mathcal{H}_{\psi',B}$, we have:*

$$L_{\mathcal{D}}(h) \leq L_{\mathcal{S}}(h) + \frac{(2B\left\|\psi\right\|_{\infty,2} + 1)\sqrt{2\log(2/\delta)}}{\sqrt{m}} + 2B\left\|\psi - \psi'\right\|_{\infty,2}$$

*Proof.* First, observe that using Theorem 26.12 in Shalev-Shwartz & Ben-David (2014), with probability at least $1-\delta$ over the choice of $\mathcal{S}$, for every $h_{\psi,\mathbf{w}} \in \mathcal{H}_{\psi,B}$ we have:

$$L_{\mathcal{D}}(h_{\psi,\mathbf{w}}) \leq L_{\mathcal{S}}(h_{\psi,\mathbf{w}}) + \frac{(2B\left\|\psi\right\|_{\infty,2} + 1)\sqrt{2\log(2/\delta)}}{\sqrt{m}}$$

In this case, using the previous lemma, for every $\psi'$ and every $h_{\psi',\mathbf{w}} \in \mathcal{H}_{\psi',B}$ we have:

$$\begin{aligned}
L_{\mathcal{D}}(h_{\psi',\mathbf{w}}) &\leq L_{\mathcal{D}}(h_{\psi,\mathbf{w}}) + B\left\|\psi - \psi'\right\|_{\infty,2} \\
&\leq L_{\mathcal{S}}(h_{\psi,\mathbf{w}}) + \frac{(2B\left\|\psi\right\|_{\infty,2}+1)\sqrt{2\log(2/\delta)}}{\sqrt{m}} + B\left\|\psi - \psi'\right\|_{\infty,2} \\
&\leq L_{\mathcal{S}}(h_{\psi',\mathbf{w}}) + \frac{(2B\left\|\psi\right\|_{\infty,2}+1)\sqrt{2\log(2/\delta)}}{\sqrt{m}} + 2B\left\|\psi - \psi'\right\|_{\infty,2}
\end{aligned}$$

□

**Lemma C.9.** *Fix $\varepsilon, \delta \in (0,1/2)$. Let $\psi$ be some mapping s.t. there exists $\mathbf{w}^\star$ satisfying $\|\mathbf{w}^\star\| \leq B$ and $L_{\mathcal{D}}(h_{\psi,\mathbf{w}^\star}) \leq \varepsilon$. Let $\mathcal{S}$ be a sample of size $m$ from $\mathcal{D}$. With probability at least $1-2\delta$ over the choice of $\mathcal{S}$, there exists a choice of learning rate, weight decay and truncation parameters s.t. if $\left\|\varphi^{(1)} - \psi\right\|_{\infty,2} \leq \frac{\varepsilon}{B}$ and $\frac{T}{\log(T)} \geq \frac{100}{\varepsilon^2}\left(\left\|\psi\right\|_{\infty,2} + B^{-1}\right)^2$ and $m \geq \frac{(4\sqrt{2}B\|\psi\|_{\infty,2}+1)^2\log(2/\delta)}{\varepsilon^2}$, GD returns a function $h$ s.t. $L_{\mathcal{D}}(h) \leq 7\varepsilon$.*

*Proof.* Consider the following convex function:

$$L(\mathbf{w}) = L_{\mathcal{S}}(h_{\varphi^{(1)},\mathbf{w}}) + \frac{\lambda}{2}\left\|\mathbf{w}\right\|^2$$

**Claim 1**: for all $\mathbf{x}$ and $y$, we have $\left|\ell(h_{\psi,\mathbf{w}^\star}(\mathbf{x}),y)\right| \leq B\left\|\psi\right\|_{\infty,2}$.

**Proof**: Observe that,

$$\left|\ell(h_{\psi,\mathbf{w}^\star}(\mathbf{x}),y)\right| \leq \left|h_{\psi,\mathbf{w}^\star}(\mathbf{x})\right| = \left|\langle\psi(\mathbf{x}),\mathbf{w}^\star\rangle\right| \leq \left\|\psi(\mathbf{x})\right\|_2\left\|\mathbf{w}^\star\right\| \leq B\left\|\psi\right\|_{\infty,2}$$

**Claim 2**: W.p. at least $1-\delta$ we have $L_{\mathcal{S}}(h_{\psi,\mathbf{w}^\star}) \leq \varepsilon + \frac{B\|\psi\|_{\infty,2}\sqrt{2\log(2/\delta)}}{\sqrt{m}}$

**Proof**: from Hoeffding's inequality, using the previous claim:

$$\Pr\left[L_S(h) \geq L_{\mathcal{D}}(h) + t\right] \leq \exp\left(-\frac{mt^2}{2\left\|\psi\right\|_{\infty,2}^2 B^2}\right)$$

And therefore,

$$\Pr\left[L_{\mathcal{S}}(h_{\psi,\mathbf{w}^\star}) \geq L_{\mathcal{D}}(h_{\psi,\mathbf{w}^\star}) + \frac{\left\|\psi\right\|_{\infty,2} B\sqrt{2\log(2/\delta)}}{\sqrt{m}}\right] \leq \delta/2$$

and the required follows from the assumption $L_{\mathcal{D}}(h_{\psi,\mathbf{w}^\star}) \leq \varepsilon$

**Claim 3**: $L(\mathbf{w}^\star) \leq 2\varepsilon + \frac{B\left\|\psi\right\|_{\infty,2}\sqrt{2\log(2/\delta)}}{\sqrt{m}} + \frac{\lambda B^2}{2}$.

**Proof**: from the previous claim, we have $L_{\mathcal{S}}(h_{\psi,\mathbf{w}^\star}) \leq \varepsilon + \frac{B\left\|\psi\right\|_{\infty,2}\sqrt{2\log(2/\delta)}}{\sqrt{m}}$. Using Lemma C.7, we get $L_{\mathcal{S}}(h_{\varphi^{(1)},\mathbf{w}^\star}) \leq \varepsilon + \frac{B\left\|\psi\right\|_{\infty,2}\sqrt{2\log(2/\delta)}}{\sqrt{m}} + B\left\|\psi - \varphi^{(1)}\right\|_{\infty,2}$ and the required follows.

**Claim 3**: there exists a step-size schedule for GD s.t. $L_{\mathcal{S}}(\mathbf{w}_T) \leq \inf_{\mathbf{w}} L_{\mathcal{S}}(\mathbf{w}) + \frac{100(\left\|\psi\right\|_{\infty,2}+\varepsilon/B)^2(1+\log(T))}{\lambda T}$.

**Proof**: Using Shamir & Zhang (2013)

Combining the previous claims, we get:

$$L(\mathbf{w}_T) \leq 2\varepsilon + \frac{B\left\|\psi\right\|_{\infty,2}\sqrt{2\log(2/\delta)}}{\sqrt{m}} + \frac{\lambda B^2}{2} + \frac{100(\left\|\psi\right\|_{\infty,2} + \varepsilon/B)^2(1+\log(T))}{\lambda T}$$

Now, choosing $\lambda = \frac{\varepsilon}{B^2}$ we get:

$$L(\mathbf{w}_T) \leq 2\varepsilon + \frac{B\left\|\psi\right\|_{\infty,2}\sqrt{2\log(2/\delta)}}{\sqrt{m}} + \frac{100B^2(\left\|\psi\right\|_{\infty,2} + \varepsilon/B)^2(1+\log(T))}{\varepsilon T}$$

So, if $\frac{T}{\log(T)} \geq \frac{100}{\varepsilon^2}\left(\left\|\psi\right\|_{\infty,2} + B^{-1}\right)^2$ and $m \geq \frac{(4\sqrt{2}B\left\|\psi\right\|_{\infty,2}+1)^2\log(2/\delta)}{\varepsilon^2}$ we have $L(\mathbf{w}_T) \leq 4\varepsilon$ and therefore $\left\|\mathbf{w}_T\right\|^2 \leq \frac{2}{\lambda}L(\mathbf{w}_T) \leq 8B^2$.

In this case, using Lemma C.8, we w.p. at least $1-\delta$:

$$L_{\mathcal{D}}(h_{\varphi^{(1)},\mathbf{w}_T}) \leq L_{\mathcal{S}}(h_{\varphi^{(1)},\mathbf{w}_T}) + \frac{(4\sqrt{2}B\left\|\psi\right\|_{\infty,2}+1)\sqrt{2\log(2/\delta)}}{\sqrt{m}} + 2B\left\|\psi-\varphi^{(1)}\right\|_{\infty,2} \leq 7\varepsilon$$

□

**Lemma C.10.** *Fix some $\delta$. Assume we initialize a network of size $r \geq 20k(2n/s)^k \log\left(\frac{2k}{\delta}\right)$. Then, w.p. at least $1-\delta$, $\mathbf{W}, \mathbf{b}$ is $r'$-good for $r' = \frac{r}{2k}(s/2n)^k$.*

*Proof.* From Lemma C.2, the probability of drawing a good neuron is $\geq (s/2n)^k$. So, for every $i$, the probability of drawing a *good* neuron with bias $\beta_i$ is at least $\frac{1}{k}(s/2n)^k$. Denote by $r'_i$ the number of good neurons with bias $\beta_i$. Observe that $\mathbb{E}[r'_i] = \frac{r}{2k}(s/2n)^k$. Using Chernoff's bound, we have:

$$\Pr\left[r'_i > \frac{r}{4k}(s/2n)^k\right] \leq \exp\left(-\frac{r}{20k}(s/2n)^k\right) \leq \frac{\delta}{2k}$$

and similarly $\Pr\left[r'_i < \frac{3r}{4k}(s/2n)^k\right] \leq \frac{\delta}{2k}$. So, using the union bound we get the required. □

**Theorem C.11.** *Fix $\delta \in (0,1/2), \varepsilon \in (0,1/2)$, and assume we choose:*

- $s \geq \alpha_1^{(k)} \frac{\log(1/\delta)}{\varepsilon^2}$.
- $r = \left\lceil \alpha_2^{(k)} (n/s)^k \log(1/\delta) \right\rceil$
- $m \geq \alpha_3^{(k)} \binom{s}{k-1} n^2 \log(nr/\delta)\log(1/\delta)$
- $T \geq \alpha_4^{(k)} \frac{\log(1/\delta)\log(T)}{\varepsilon^2}$

*for some constants $\alpha_1^{(k)}, \alpha_2^{(k)}, \alpha_3^{(k)}, \alpha_4^{(k)}$. Then, with probability at least $1-\delta$ over the choice of sample size and initialization, gradient descent returns after $T$ iterations a function $h$ s.t. $L_{\mathcal{D}}(h) \leq \varepsilon$.*

*Proof.* Choose $r = \left\lceil 20k(2n/s)^k \log\left(\frac{2k}{\delta}\right)\right\rceil$.

- From Lemma C.10, with probability at least $1-\delta$ we get an $r'$-*good* init with $r' = \frac{r}{2k}(s/2n)^k$ and notice that $10\log(2k/\delta) \le r' \le 20\log(2k/\delta)$.
- Let $\eta = \frac{1}{2k}\left|C_{k,s}\right|^{-1} \ge \frac{\kappa_k}{2k}\sqrt{\binom{s}{k-1}}$ and

$$\tau = \frac{\varepsilon}{40B_k n\kappa_k\sqrt{\binom{s}{k-1}}\log(2k/\delta)} \le \min\left(\frac{\varepsilon}{4B_k k r' n\eta}, \frac{1}{16\eta k n}\right)$$

  Choosing
  - $m \ge \frac{4\cdot 40^2 B_k^2\kappa_k n^2\binom{s}{k-1}\log(2k/\delta)^2\log(4nr/\delta)}{\varepsilon^2} \ge \frac{4\log(4nr/\delta)}{\tau^2}$
  - $s \ge \frac{80kB_k^2\log(2k/\delta)}{\varepsilon^2} \ge \frac{4kr'B_k^2}{\varepsilon^2}$

  from Lemma C.6, the conditions for Lemma C.9 are satisfied with
  1. $B = B_k$.
  2. $\|\psi\|_{\infty,2} \le \sqrt{8kr'} \le 4\sqrt{10k\log(2k/\delta)}$
  3. $\left\|\psi - \varphi^{(1)}\right\|_{\infty,2} \le 4kr'n\eta\tau \le \frac{\varepsilon}{B_k}$
- Choosing
  - $\frac{T}{\log(T)} \ge \frac{100\left(4\sqrt{10k\log(2k/\delta)}+B^{-1}\right)^2}{\varepsilon^2} \ge \frac{100}{\varepsilon^2}\left(\|\psi\|_{\infty,2}+B^{-1}\right)^2$
  - $m \ge \frac{(16\sqrt{2}B\sqrt{10k\log(2k/\delta)}+1)^2\log(2/\delta)}{\varepsilon^2} \ge \frac{(4\sqrt{2}B\|\psi\|_{\infty,2}+1)^2\log(2/\delta)}{\varepsilon^2}$

  using Lemma C.9 we get w.p. at least $1-2\delta$ we have $L_{\mathcal{D}}(h) \le 7\varepsilon$.

□

### C.2.3 Feature selection with an under-sparse initialization and a narrow network

Fix some subset $S \subseteq \binom{n}{k}$ to be the true parity function. Let $k$ be even.

We train the following network:

$$f_{\mathbf{W},\mathbf{b},\mathbf{u},\beta}(\mathbf{x}) = \sum_{i=1}^{r} u_i \sigma(\langle \mathbf{w}_i, \mathbf{x} \rangle + b_i) + \beta.$$

We initialize the network as follows:

- Randomly initialize $\mathbf{w}_1, \ldots, \mathbf{w}_{r/2} \in \{\varepsilon, 1\}^n$ s.t. $s$ coordinates have weight 1 and rest have weight $\varepsilon < \frac{1}{(n-s)}$, with a uniform distribution over all $\binom{n}{s}$ subsets.
- Randomly initialize $b_1, \ldots, b_{r/2} \sim \left\{-\varepsilon\frac{k-1}{2k}, -\varepsilon\frac{k-3}{2k}, \ldots, -\varepsilon\frac{1}{2k}, \varepsilon\frac{1}{2k}, \ldots, \varepsilon\frac{k-3}{2k}, \varepsilon\frac{k-1}{2k}\right\}$.
- Randomly initialize $u_1, \ldots, u_{r/2} \in \{\pm 1\}$ uniformly at random.
- Initialize $\mathbf{w}_{r/2+1}, \ldots, \mathbf{w}_r, b_{r/2+1}, \ldots, b_r$ s.t. $\mathbf{w}_i = \mathbf{w}_{i-r/2}$ and $b_i = b_{i-r/2}$ (symmetric initialization).
- Initialize $u_{r/2+1}, \ldots, u_r$ s.t. $u_i = -u_{i-r/2}$.
- Initialize $\beta = 0$

Similar to the over-sparse case, we consider hinge-loss. We consider one-step of gradient descent on a sample $\mathcal{S}$ with $\ell_2$ regularization (weight decay):

$$\theta^{(1)} = (1-\lambda)\theta^{(0)} - \eta \operatorname{trunc}(\nabla L_{\mathcal{S}}(f_{\theta^{(0)}}), \gamma)$$

with learning rate $\eta$, truncation parameter $\gamma$, and the weight decay $\lambda$ chosen differently for each layer and separately for the weights and biases. Note that truncation just zeros out gradients with magnitude $\leq \gamma$.

MAJORITY AND HALF. We will make use of two Boolean functions: (1) Majority, and (2) Half (derivative of Majority). For input $x \in \{\pm 1\}^n$, we define

$$\mathrm{Maj}(x) = \mathrm{sign}\left(\sum_{i=1}^{n} x_i\right)$$

where $\mathrm{sign}(a) = 1$ is $a > 0$ else $-1$. The derivative of the Majority function is denoted by $D_n\mathrm{Maj}_n = \mathrm{Half}$ and is defined for input $x \in \{\pm 1\}^{n-1}$ as:

$$\mathrm{Half}(x) = 1\left(\sum_{i=1}^{n} x_i = 0\right).$$

The corresponding Fourier coefficients corresponding to set $S$ are denoted by $\widehat{\mathrm{Maj}}_n(S)$ and $\widehat{\mathrm{Half}}_{n-1}(S)$. Note that both functions are permutation invariant, so the Fourier coefficients only depend on the size of the set.

**Lemma C.12** (O'Donnell (2014)). *For any integers $m \geq j$, we have*

$$\widehat{\mathrm{Half}}_{2m}(2j) = \widehat{\mathrm{Maj}}_{2m+1}(2j+1) = (-1)^j \frac{\binom{m}{j}}{\binom{2m}{2j}} \frac{\binom{2m}{m}}{2^{2m}}.$$

POPULATION GRADIENTS AT INITIALIZATION. Consider a fixed weight $\mathbf{w}$ with the set of 1 weights indicated by $S' \subseteq S$. We first start with computing the population gradients for this neuron ($h_{\mathbf{w}}(\mathbf{x}) = \sigma(\langle \mathbf{w}, \mathbf{x} \rangle)$ be a single ReLU neuron where $\sigma(x) = \max\{x, 0\}$) at initialization for varying $S \cap S' = \bar{S}$ and parity and non-parity variables.

**Lemma C.13** (Population gradient at initialization for parity variables). *Assuming $s < k$ with $s, k$ even, for $i \in S$, we have*

$$\mathop{\mathbb{E}}_{\mathbf{x} \sim \{\pm 1\}^n}\left[\frac{\partial}{\partial w_i} h_{\mathbf{w}}(\mathbf{x}) \cdot \chi_S(\mathbf{x})\right] = \frac{1}{2}\widehat{\mathrm{Half}}_s(\bar{S} \setminus \{i\})\widehat{\mathrm{Maj}}_{n-s}(S \setminus (\bar{S} \cup \{i\})).$$

*Proof.* For $i \in S$, we have

$$\mathop{\mathbb{E}}_{\mathbf{x}\sim\{\pm 1\}^n}\left[\frac{\partial}{\partial w_i} h_{\mathbf{w}}(\mathbf{x}) \cdot \chi_S(\mathbf{x})\right]$$
$$= \mathop{\mathbb{E}}_{\mathbf{x}\sim\{\pm 1\}^n}\left[\sigma'(\langle \mathbf{w}, \mathbf{x}\rangle) \cdot x_i \cdot \chi_S(\mathbf{x})\right]$$

Since $i \in S$:

$$= \mathop{\mathbb{E}}_{\mathbf{x}\sim\{\pm 1\}^n}\left[\mathbb{1}_{\left\{\sum_{j\in S'} x_j + \varepsilon \sum_{j\notin S'} x_j > 0\right\}} \chi_{S\setminus\{i\}}(\mathbf{x})\right]$$

Splitting based on value of $\sum_{j\in S'} x_j$:

$$= \mathop{\mathbb{E}}_{\mathbf{x}\sim\{\pm 1\}^n}\left[\mathbb{1}_{\left\{\sum_{j\in S'} x_j = 0\right\}} \mathbb{1}_{\left\{\sum_{j\notin S'} x_j > 0\right\}} \chi_{S\setminus\{i\}}(\mathbf{x})\right] + \mathop{\mathbb{E}}_{\mathbf{x}\sim\{\pm 1\}^n}\left[\mathbb{1}_{\left\{\sum_{j\in S'} x_j > 0\right\}} \mathbb{1}_{\left\{\sum_{j\in S'} x_j + \varepsilon \sum_{j\notin S'} x_j > 0\right\}} \chi_{S\setminus\{i\}}(\mathbf{x})\right]$$

Using the fact that $\left|\varepsilon \sum_{j\notin S'} x_j\right| \leq (n-s)\varepsilon < 1$:

$$= \mathop{\mathbb{E}}_{\mathbf{x}\sim\{\pm 1\}^n}\left[\mathbb{1}_{\left\{\sum_{j\in S'} x_j = 0\right\}} \mathbb{1}_{\left\{\sum_{j\notin S'} x_j \geq 0\right\}} \chi_{S\setminus\{i\}}(\mathbf{x})\right] + \mathop{\mathbb{E}}_{\mathbf{x}\sim\{\pm 1\}^n}\left[\mathbb{1}_{\left\{\sum_{j\in S'} x_j > 0\right\}} \chi_{S\setminus\{i\}}(\mathbf{x})\right]$$

Splitting between variables in $S'$ and outside $S'$:

$$= \mathop{\mathbb{E}}_{\mathbf{x}\sim\{\pm 1\}^n}\left[\mathbb{1}_{\left\{\sum_{j\in S'} x_j = 0\right\}} \chi_{\bar{S}\setminus\{i\}}(\mathbf{x})\right] \mathop{\mathbb{E}}_{\mathbf{x}\sim\{\pm 1\}^n}\left[\mathbb{1}_{\left\{\sum_{j\notin S'} x_j \geq 0\right\}} \chi_{S\setminus(\bar{S}\cup\{i\})}(\mathbf{x})\right] + \mathop{\mathbb{E}}_{\mathbf{x}\sim\{\pm 1\}^n}\left[\mathbb{1}_{\left\{\sum_{j\in S'} x_j > 0\right\}} \chi_{S\setminus\{i\}}(\mathbf{x})\right]$$

Replacing indicators with Maj and Half appropriately:

$$= \mathop{\mathbb{E}}_{\mathbf{x}\sim\{\pm1\}^n}\left[\mathrm{Half}_s(\mathbf{x}_{S'})\chi_{\bar{S}\setminus\{i\}}(\mathbf{x})\right]\mathop{\mathbb{E}}_{\mathbf{x}\sim\{\pm1\}^n}\left[\frac{1}{2}\left(\mathrm{Maj}_{n-s}(\mathbf{x}_{[n]\setminus S'})+1\right)\chi_{S\setminus(\bar{S}\cup\{i\})}(\mathbf{x})\right]$$
$$+\mathop{\mathbb{E}}_{\mathbf{x}\sim\{\pm1\}^n}\left[\frac{1}{2}\left(\mathrm{Maj}_{s}(\mathbf{x}_{S'})+1\right)\chi_{\bar{S}\setminus\{i\}}(\mathbf{x})\right]\mathop{\mathbb{E}}_{\mathbf{x}\sim\{\pm1\}^{n-s}}\left[\chi_{S\setminus(\bar{S}\cup\{i\})}(\mathbf{x})\right]$$

Replacing indicators with Fourier coefficients appropriately:

$$= \frac{1}{2}\widehat{\mathrm{Half}}_s(\bar{S}\setminus\{i\})\left(\widehat{\mathrm{Maj}}_{n-s}(S\setminus(\bar{S}\cup\{i\}))+\mathop{\mathbb{1}}_{\{S\subseteq\bar{S}\cup\{i\}\}}\right)+\frac{1}{2}\left(\widehat{\mathrm{Maj}}_{s}(\bar{S}\setminus\{i\})+\mathop{\mathbb{1}}_{\{\bar{S}\setminus\{i\}=\varphi\}}\right)\mathop{\mathbb{1}}_{\{S\subseteq\bar{S}\cup\{i\}\}}$$
$$= \frac{1}{2}\widehat{\mathrm{Half}}_s(\bar{S}\setminus\{i\})\widehat{\mathrm{Maj}}_{n-s}(S\setminus(\bar{S}\cup\{i\}))+\frac{1}{2}\left(\widehat{\mathrm{Maj}}_{s}(\bar{S}\setminus\{i\})+\mathop{\mathbb{1}}_{\{\bar{S}\setminus\{i\}=\varphi\}}+\widehat{\mathrm{Half}}_s(\bar{S}\setminus\{i\})\right)\mathop{\mathbb{1}}_{\{S\subseteq\bar{S}\cup\{i\}\}}$$

Since $k, s$ are even and $k < s$, $|\bar{S}\cup\{i\}| \le s+1 < k = |S|$ therefore $\mathbb{1}_{\{S\subseteq\bar{S}\cup\{i\}\}} = 0$:

$$= \frac{1}{2}\widehat{\mathrm{Half}}_s(\bar{S}\setminus\{i\})\widehat{\mathrm{Maj}}_{n-s}(S\setminus(\bar{S}\cup\{i\})).$$

This gives us the desired result. □

**Lemma C.14** (Population gradient at initialization for non-parity variables)**.** *Assuming $s < k$ with $s, k$ even, for $i \notin S$, we have*

$$\mathop{\mathbb{E}}_{\mathbf{x}\sim\{\pm1\}^n}\left[\frac{\partial}{\partial w_i}h_{\mathbf{w}}(\mathbf{x})\cdot\chi_S(\mathbf{x})\right] = \frac{1}{2}\widehat{\mathrm{Half}}_s(\bar{S}\cup(S'\cap\{i\}))\widehat{\mathrm{Maj}}_{n-s}((S\cup\{i\})\setminus S').$$

*Proof.* For $i \notin S$, using similar calculations as in the proof of Lemma C.13, we have

$$
\begin{aligned}
&\mathop{\mathbb{E}}_{\mathbf{x}\sim\{\pm1\}^n}\left[\frac{\partial}{\partial w_i}h_{\mathbf{w}}(\mathbf{x})\cdot\chi_S(\mathbf{x})\right]\\
&=\mathop{\mathbb{E}}_{\mathbf{x}\sim\{\pm1\}^n}\left[\sigma'(\langle\mathbf{w},\mathbf{x}\rangle)\cdot x_i\cdot\chi_S(\mathbf{x})\right]\\
&=\mathop{\mathbb{E}}_{\mathbf{x}\sim\{\pm1\}^n}\left[\mathbb{1}_{\left\{\sum_{j\in S'}x_j+\varepsilon\sum_{j\notin S'}x_j>0\right\}}\chi_{S\cup\{i\}}(\mathbf{x})\right]\\
&=\mathop{\mathbb{E}}_{\mathbf{x}\sim\{\pm1\}^n}\left[\mathbb{1}_{\left\{\sum_{j\in S'}x_j=0\right\}}\mathbb{1}_{\left\{\sum_{j\notin S'}x_j>0\right\}}\chi_{S\cup\{i\}}(\mathbf{x})\right]+\mathop{\mathbb{E}}_{\mathbf{x}\sim\{\pm1\}^n}\left[\mathbb{1}_{\left\{\sum_{j\in S'}x_j>0\right\}}\chi_{S\cup\{i\}}(\mathbf{x})\right]\\
&=\mathop{\mathbb{E}}_{\mathbf{x}\sim\{\pm1\}^n}\left[\mathbb{1}_{\left\{\sum_{j\in S'}x_j=0\right\}}\chi_{\bar{S}\cup(S'\cap\{i\})}(\mathbf{x})\right]\mathop{\mathbb{E}}_{\mathbf{x}\sim\{\pm1\}^n}\left[\mathbb{1}_{\left\{\sum_{j\notin S'}x_j>0\right\}}\chi_{(S\cup\{i\})\setminus S'}(\mathbf{x})\right]\\
&\quad+\mathop{\mathbb{E}}_{\mathbf{x}\sim\{\pm1\}^n}\left[\mathbb{1}_{\left\{\sum_{j\in S'}x_j>0\right\}}\chi_{S\cup\{i\}}(\mathbf{x})\right]\\
&=\mathop{\mathbb{E}}_{\mathbf{x}\sim\{\pm1\}^n}\left[\mathrm{Half}_s(\mathbf{x}_{S'})\chi_{\bar{S}\cup(S'\cap\{i\})}(\mathbf{x})\right]\mathop{\mathbb{E}}_{\mathbf{x}\sim\{\pm1\}^n}\left[\frac{1}{2}\left(\mathrm{Maj}_{n-s}(\mathbf{x}_{[n]\setminus S'})+1\right)\chi_{(S\cup\{i\})\setminus S'}(\mathbf{x})\right]\\
&\quad+\mathop{\mathbb{E}}_{\mathbf{x}\sim\{\pm1\}^n}\left[\frac{1}{2}\left(\mathrm{Maj}_s(\mathbf{x}_{S'})+1\right)\chi_{\bar{S}\cup(S'\cap\{i\})}(\mathbf{x})\right]\mathop{\mathbb{E}}_{\mathbf{x}\sim\{\pm1\}^n}\left[\chi_{(S\cup\{i\})\setminus S'}(\mathbf{x})\right]\\
&=\frac{1}{2}\widehat{\mathrm{Half}}_s(\bar{S}\cup(S'\cap\{i\}))\widehat{\mathrm{Maj}}_{n-s}((S\cup\{i\})\setminus S')\\
&\quad+\frac{1}{2}\left(\widehat{\mathrm{Maj}}_s(\bar{S}\cup(S'\cap\{i\}))+\mathbb{1}_{\left\{\bar{S}=\varnothing\wedge i\notin S'\right\}}+\widehat{\mathrm{Half}}_s(\bar{S}\setminus\{i\})\right)\mathbb{1}_{\{S\cup\{i\}\subseteq S'\}}
\end{aligned}
$$

Using the fact that $|S'|<|S|$ therefore $\mathbb{1}_{\{S\cup\{i\}\subseteq S'\}}=0$:

$$
=\frac{1}{2}\widehat{\mathrm{Half}}_s(\bar{S}\cup(S'\cap\{i\}))\widehat{\mathrm{Maj}}_{n-s}((S\cup\{i\})\setminus S').
$$

□

Similar to the over-sparse setting, we say a neuron is *good* if $S'\subset S$, that is, if the selected variables are a subset of the relevant variables. Then we have,

**Lemma C.15** (Population gradient at initialization for good neurons). *Assuming $s < k$ with $s, k$ even, for good neurons, for $\bar{S} = S' \cap S$, we have*

$$\mathop{\mathbb{E}}_{\mathbf{x}\sim\{\pm 1\}^n}\left[\frac{\partial}{\partial w_i} h_{\mathbf{w}}(\mathbf{x}) \cdot \chi_S(\mathbf{x})\right] = \begin{cases} \frac{1}{2}\widehat{\mathrm{Half}}_s(s)\widehat{\mathrm{Maj}}_{n-s}(k-s-1) & \text{if } i \in S \setminus S', \\ \frac{1}{2}\widehat{\mathrm{Half}}_s(s)\widehat{\mathrm{Maj}}_{n-s}(k-s+1) & \text{if } i \notin S, \\ 0 & \text{otherwise.} \end{cases}$$

*Proof.* Using Lemma C.13 and C.14, we get

1. $i \in S, i \in S'$:

$$\mathop{\mathbb{E}}_{\mathbf{x}\sim\{\pm 1\}^n}\left[\frac{\partial}{\partial w_i} h_{\mathbf{w}}(\mathbf{x}) \cdot \chi_S(\mathbf{x})\right] = \frac{1}{2}\widehat{\mathrm{Half}}_s(\bar{S} \setminus \{i\})\widehat{\mathrm{Maj}}_{n-s}(S \setminus \bar{S})$$

2. $i \in S, i \notin S'$:

$$\mathop{\mathbb{E}}_{\mathbf{x}\sim\{\pm 1\}^n}\left[\frac{\partial}{\partial w_i} h_{\mathbf{w}}(\mathbf{x}) \cdot \chi_S(\mathbf{x})\right] = \frac{1}{2}\widehat{\mathrm{Half}}_s(\bar{S})\widehat{\mathrm{Maj}}_{n-s}(S \setminus (\bar{S} \cup \{i\}))$$

3. $i \notin S, i \in S'$:

$$\mathop{\mathbb{E}}_{\mathbf{x}\sim\{\pm 1\}^n}\left[\frac{\partial}{\partial w_i} h_{\mathbf{w}}(\mathbf{x}) \cdot \chi_S(\mathbf{x})\right] = \frac{1}{2}\widehat{\mathrm{Half}}_s(\bar{S} \cup \{i\})\widehat{\mathrm{Maj}}_{n-s}(S \setminus \bar{S})$$

4. $i \notin S, i \notin S'$:

$$\mathop{\mathbb{E}}_{\mathbf{x}\sim\{\pm 1\}^n}\left[\frac{\partial}{\partial w_i} h_{\mathbf{w}}(\mathbf{x}) \cdot \chi_S(\mathbf{x})\right] = \frac{1}{2}\widehat{\mathrm{Half}}_s(\bar{S})\widehat{\mathrm{Maj}}_{n-s}((S \setminus \bar{S}) \cup \{i\})$$

Since $S' \subseteq S$, the arguments to the $\widehat{\mathrm{Half}}$ for (1) and (3) are odd. The corresponding Fourier coefficients for odd sets for Half are 0 (see O'Donnell (2014)), thus we have the desired result. □

**Theorem C.16** (Formal version of Theorem 6.5)**.** *Fix $s, k, \varepsilon$ such that $s < k, \varepsilon > 0$. Then for network width $r \geq \Omega((n/k)^s)$, the initialization scheme proposed here guarantees that for every $(n, k)$-parity distribution $\mathcal{D}$, with probability at least $0.99$ over the choice of sample and initialization, after one step of batch gradient descent with sample size $m = O((n/k)^{k-s-1})$ and appropriate choice of learning rate, there is a subnetwork in the one-layer ReLU MLP that has at least $\frac{1}{2} - \varepsilon$ correlation with the parity function.*

*Proof.* To compute the parity function, we need to have $k$ neurons which identify the correct coordinates and have the appropriate biases. In order to identify the correct coordinates, we will focus only on good neurons, and on population gradient. We will then argue by standard concentration arguments that this holds from samples.

Let us consider a good neuron with $S' \subseteq S$. Firstly note that the scale of the bias is set such that it does not affect the gradient, Since it is at most $\varepsilon/2$. Thus we can assume the no bias case for gradient computation. For all $i \in S \setminus S'$, that is, the set of relevant variables that are not selected in the initialization, let $\xi_{S\setminus S'}$ denote the gradient at initialization, and for all $i \notin S$, that is, the set of irrelevant variables, let $\xi_{[n]\setminus S}$ denote the gradient at initialization. Then using Lemma C.15 and Lemma C.12, we have

$$\frac{\left|\xi_{S\setminus S'}\right|}{\left|\xi_{[n]\setminus S}\right|} = \frac{n-s}{k-s-1} > 1.$$

With $\eta$ and $\gamma$ being 0 on the bias terms and the second layer, $\eta = -\frac{\varepsilon}{2k\left|\xi_{S\setminus S'}\right|}$ for the first layer weights, $\lambda = 1 - \frac{\varepsilon}{2k}$ and $\gamma = \left|\xi_{[n]\setminus S}\right|$, one step of truncated population gradient descent gives us, for all $i \in S$, $w_i = \frac{\varepsilon}{2k}$ and for all $i \notin S$, $|w_i| = \frac{\varepsilon^2}{k}$. Since our initialization has two copies of the same neuron with second layer weights 1 and $-1$, one of them will have the gradient in the correct direction, ensuring the above, in particular, the one with the weight being $\operatorname{sign}(\xi_{S\setminus S'})$. Since $\varepsilon$ can be set to be arbitrarily small, terms with $\varepsilon^2$ can be ignored in comparison to terms with $\varepsilon$. Thus the non-parity coefficients do not affect the output of the function and we get the appropriate parity

coefficients (scaled by $\varepsilon/2k$). To extend these guarantees to the batch gradient setting, we need to compute gradients to precision

$$\tau = \left|\xi_{[n]\setminus S}\right|/2 = c_s(n-s)^{-\frac{k-s-1}{2}}$$

for some constant $c_s$ dependent only on $s$. This implies a sample complexity of $O(c_s^2(n-s)^{k-s-1})$ using standard Chernoff bound. As for the width, we need to ensure that we have the required number of good neurons with appropriate bias. The probability of a randomly initialized neuron to be good is

$$\frac{\binom{k}{s}}{\binom{n}{s}} = \Theta\left(\left(\frac{k}{n}\right)^s\right).$$

The probability of choosing the appropriate bias is $1/(k+1)$. Thus to be able to choose $k+1$ good neurons with correct biases, we need width $O(k^2(n/k)^s)$. □

We provide some additional remarks on the proof of Theorem 6.5:

- The sample complexity compared to the dense initialization studied by Barak et al. (2022) improves by a factor of $n^s$, at the cost of a higher width by a factor of $n^s$.
- We conjecture that Theorem 6.5 can be strengthened to an end-to-end guarantee for the full network, like Theorem 6.4. The technical challenge lies in analyzing how weight decay uniformly prunes the large irrelevant coordinates, without decaying the good subnetwork. We believe that this occurs robustly (from the experiments on sparse initialization), but requires a more refined analysis of the optimization trajectory.

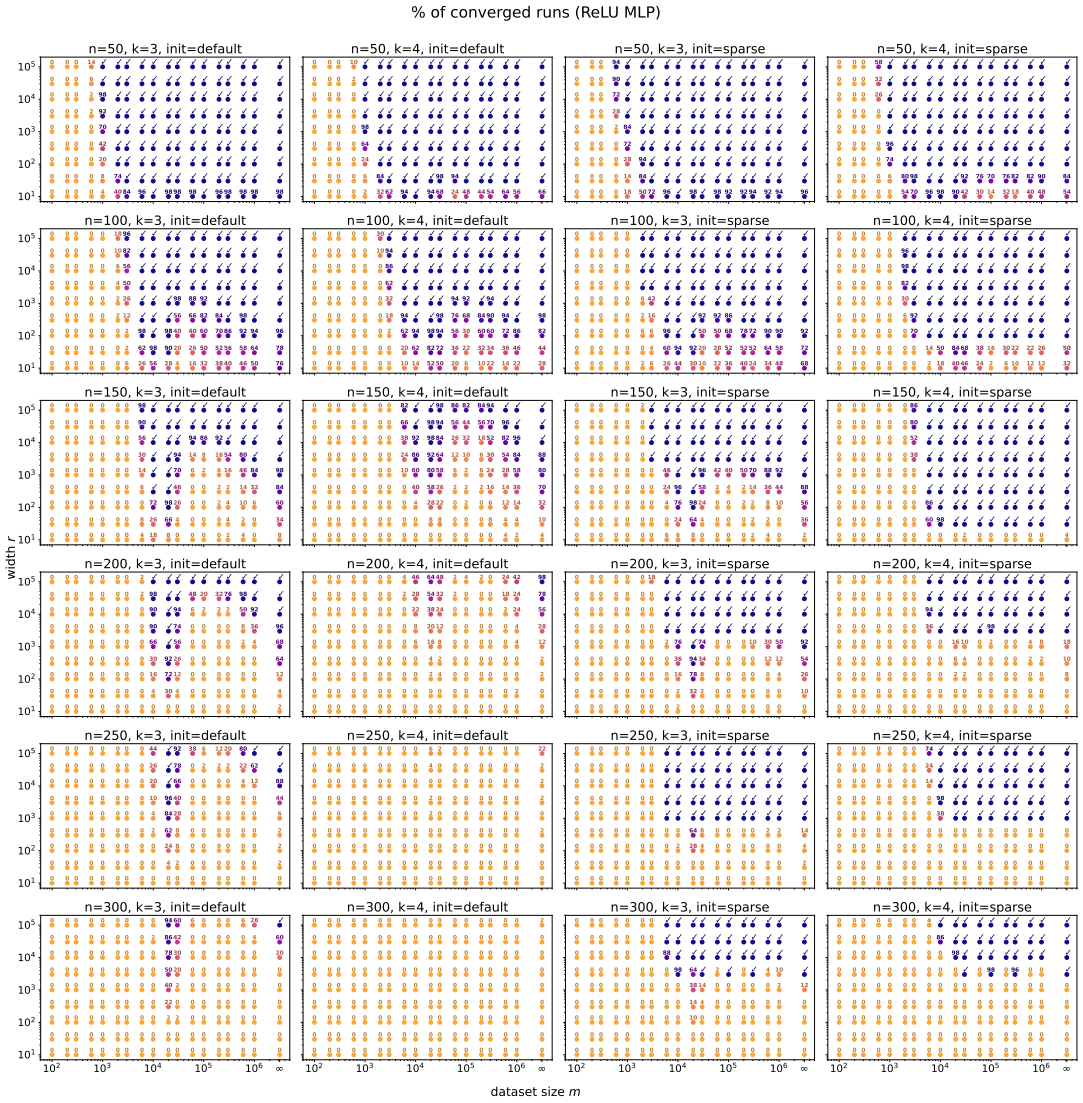



**Figure C.1:** Full results for the sweep over dataset sizes and MLP widths. Each point represents $50$ independent training runs of a 2-layer ReLU MLP on the offline sparse parity problem (input dimension $n$, parity degree $k$, initialization scheme, width $r$, dataset size $m$). All other algorithmic choices (learning rate $\eta = 0.1$, weight decay $\lambda = 0.01$, batch size $B = 32$, number of training iterations $T = 10^5$) are kept the same. We note the following: (1) **A success frontier**, where wide networks can learn at small sample sizes $m \ll BT$ (far outside online regime); (2) **Monotonic benefit of width**: overparameterization does not worsen the failure mode of overfitting in this setting, and amplifies success probabilities; (3) **Benefit of sparse axis-aligned initialization**: for sufficiently large $n$ where default initialization no longer works, sparse initialization scheme enlarges the feasible regime; (4) **Non-monotonic effects of dataset size:** there are unpredictable failures as we vary $m$ (horizontal slices of these plots).

## C.3 Full experimental results

### C.3.1 Full sweeps over dataset size and width

In our main set of large-scale synthetic experiments, we train a large number of 2-layer MLPs to solve various $(n, k)$-sparse parity problems, from $m$ samples. The full set of hyperparameters is listed below:

- Problem instance sizes: $n \in \{50, 100, 150, \ldots, 300\}$, $k \in \{3, 4\}$. Note that Barak et al. (2022) investigate empirical *computational time* scaling curves for larger $k$ (up to 8) in the online ($m = \infty$) setting. We are able to observe convergence for larger $k$ in the offline setting, but we omit these results from the systematic grid sweep (the feasible regime is too small in terms of $n$).
- Dataset size: $m \in \{100, 200, 300, 600, 1000, 2000, 3000, \ldots, 600000, 1000000\}$. We also include runs in the online regime (rightmost columns in Figure C.1), which correspond to the regime studied by Barak et al. (2022).
- Network width: $r \in \{10, 30, 100, 300, \ldots, 10000, 30000, 100000\}$.
- Initialization scheme: PyTorch default (uniform on the interval $[-1/\sqrt{n}, 1/\sqrt{n}]$), and random 2-sparse rows. In coarser-grained hyperparameter searches, we found 2 to be the optimal sparsity constant for large-width ($\geq 1000$) regimes studied in this paper; we do not fully understand why this is the case. We also keep the PyTorch default initialization scheme (uniform on the width-dependent interval $[-1/\sqrt{r}, 1/\sqrt{r}]$) for the second layer, and use default-initialized biases.

At each point in this hyperparameter space, we conduct 50 training runs, and record the *success probability*, defined as the probability of achieving test error $\leq 10\%$ on a held-out sample of size

$10^4$ within $T = 10^5$ training iterations. The hyperparameters for SGD, selected via coarse-grained hyperparameter search to optimize for convergence time in the $n = 200, k = 3, r = 10000$ setting, are as follows: minibatch size $B = 32$; learning rate $\eta = 0.1$; weight decay $\lambda = 0.01$.

Figure C.1 summarizes all of our runs, and overviews all of the findings (1) through (4) enumerated in the main paper. We go into more detail below:

(1) **A "success frontier": large width can compensate for small datasets.** We observe convergence and perfect generalization when $m \ll n^k$. In such regimes, which are far outside the online setting considered by Barak et al. (2022), high-probability sample-efficient learning is enabled by large width. Note that neither our theoretical or empirical results have sufficient granularity to predict or measure the precise way the smallest feasible sample size $m$ scales with the other size parameters (like $n, k, r$). The theoretical upper bounds show that if $r = \Omega(n^k)$, idealized algorithms (modified for tractability of analysis) can obtain $O(\text{poly}(n))$ or even $O(\log(n))$ sample complexity, and smaller $r$ can yield milder reductions of the exponent.

(2) **Width is monotonically beneficial, and buys data, time, and luck.** Despite the capacity of wider neural networks to overfit larger datasets, we find that there are *monotonic sample-efficiency benefits* to increasing network width, in *all* of the hyperparameter settings considered in the grid sweep. This can be quickly quantitatively confirmed by starting at any point in Figure C.1, and noting that success probabilities only increase* going upwards (increasing $r$, keeping all other parameters equal). Along some of these vertical slices, we observe that transitions from 0% to 100% are present: at these corresponding dataset sizes $m$, *large width makes sample-efficient learning possible*. Figure 6.2 (center) shows this in greater detail, by choosing a denser grid of sample sizes $m$ near the empirical statistical limit.

*Small exceptions (such as $98\% \to 96\%$ for $n = 100, k = 4, m = 2000$) are all within the standard error margins of Bernoulli confidence intervals.

(3) **Sparse axis-aligned initialization buys data, time, and luck.** Used in conjunction with a wide network, we observe that a sparse, axis-aligned initialization scheme yields strong improvements on all of these axes. This can be seen by comparing the pairs of subplots in columns 1 vs. 3 and 2 vs. 4 in Figure C.1. We found that $s = 2$ (i.e. initialize every hidden-layer neuron with a random 2-hot weight vector) works best for the settings considered in this study.

(4) **Intriguing effects of dataset size.** Unlike the monotonicity along vertical slices in Figure C.1, some of the *horizontal* slices exhibit non-monotonic success probabilities. Namely, as $m$ increases, keeping all else the same, the network enters and exits a first feasible regime; then, at large enough sample sizes (including the online setting), learning is observed to succeed again. Sparse initialization reduces this counterintuitive behavior, but not entirely (see, e.g., the $n = 200, k = 3$ cell). We do not attempt to explain this phenomenon; however, we found in preliminary investigations that the locations of the transitions are sensitive to the choice of weight decay hyperparameter. Figure 6.2 (right) shows this in greater detail, plotting median convergence times (as defined above) instead of success probabilities.

### C.3.2 Lottery ticket subnetworks

Our theoretical analysis of sparse networks and experimental findings suggest that width provides a form of parallelization: wider networks have a higher probability of containing lucky neurons which have sufficient Fourier gaps at initialization to learn from the dataset. In Figure C.2 we perform an experiment in the style of Frankle & Carbin (2018), showing that indeed the neurons which end up being important in a wide sparsely-initialized network form an unusually lucky 'winning ticket' subnetwork. When we rewind the weights of this subnetwork to initialization, its test error starts out poor, but when we train just this subnetwork from initialization its performance quickly

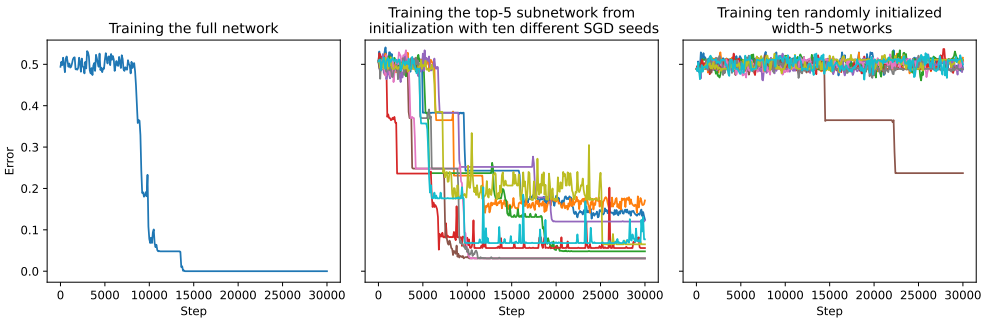



**Figure C.2:** (Lottery tickets) Left: Training a width-100 MLP on the (n=50, k=5)-online sparse parity task, where each hidden neuron initialized with 2 non-zero incoming weights. Center: We prune all but the top 5 neurons by weight norm at the end of training; rewind weights to the original initialization, and retrain with various SGD random seeds. Right: The same as Center, but the weights are randomly reinitialized in each run.

improves, unlike the large majority of randomly initialized subnetworks of the same size. For this experiment, batch size=32 and learning rate=0.1.

### C.3.3 Training wide & sparsely-initialized MLPs on natural tabular data

As a preliminary investigation of whether our findings translate well to realistic settings, we conduct experiments on the tabular benchmark curated by Grinsztajn et al. (2022). For simplicity, we use all 16 numerical classification tasks from the benchmark. These datasets originate from diverse domains such as health care, finance, and experimental physics. Our main comparison is between MLPs (with algorithmic choices inspired by our sparse parity findings) and random forests Breiman (2001). We chose random forests as a baseline because they are known to achieve competitive performance with little tuning. Figure C.3 summarizes our results, in which we find that wide and sparsely-initialized MLPs improve sample efficiency on small tabular datasets. We describe these experiments in full detail below.

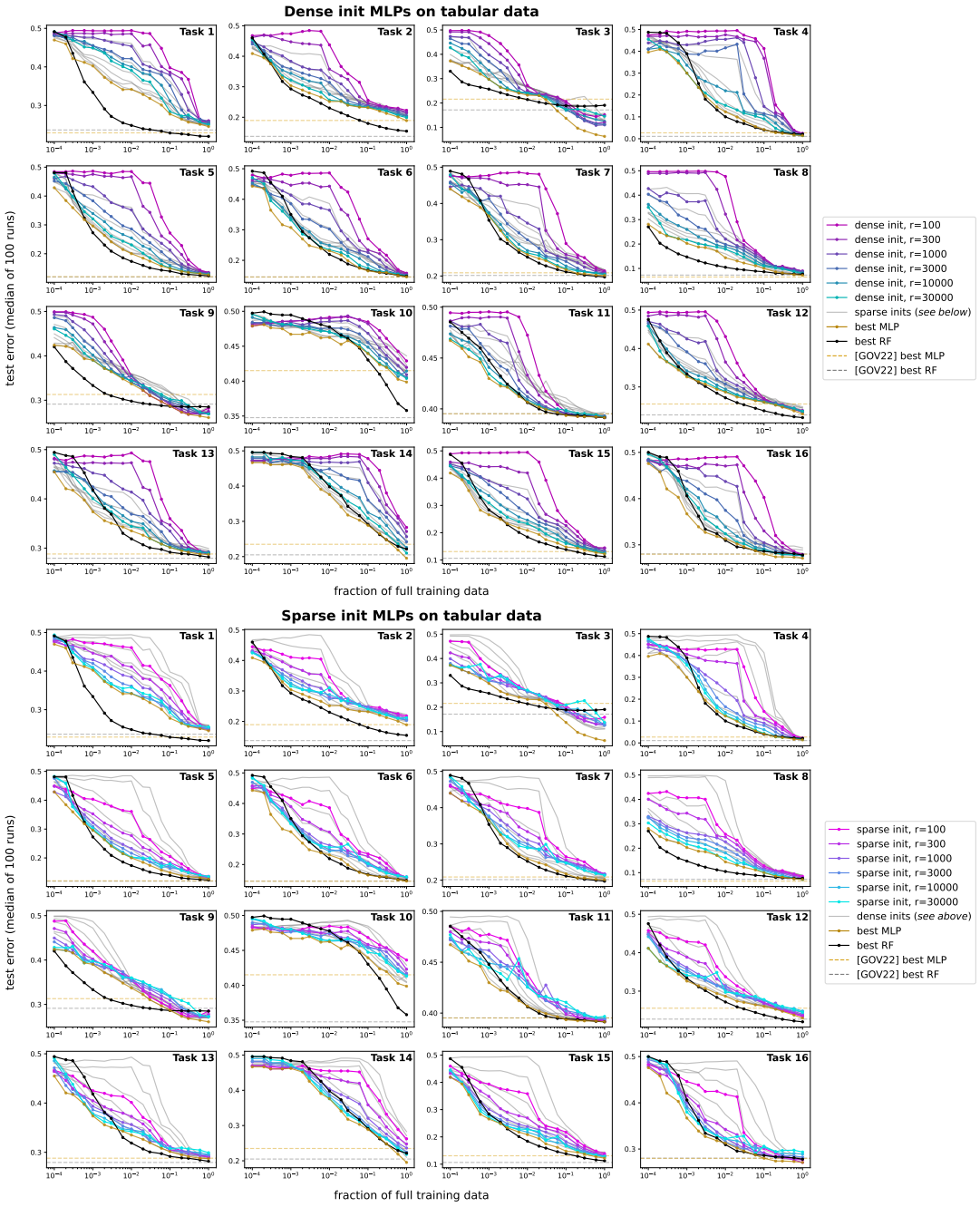



**Figure C.3:** Full results of our tabular data experiments (following Grinsztajn et al. (2022)), varying MLP width $r$, sample size $m$ (via downsampling the training data), and initialization sparsity. Both width and sparsity tend to improve generalization, especially for small datasets. See Tables C.1 and C.2 for further details.

Data preprocessing. We standardize the dataset, centering each feature to have mean 0 and normalizing each feature to have standard deviation 1.* For each task, we set aside 10% of the data for the test set, and downsample varying fractions of the remaining data to form a training set. We vary the downsampling fraction on an exponential grid: {1, 0.6, 0.3, 0.2, 0.1, 0.06, 0.03, 0.02, 0.01, 0.006, 0.003, 0.002, 0.001, 0.0006, 0.0003, 0.0002, 0.0001}. In each of 100 i.i.d. trials for each setting, we re-randomize the validation split as well as the downsampled training set.

Algorithms. MLPs are trained using the same architectural choices noted in Section C.3.1, except the hyperparameters noted below. We use the scikit-learn Pedregosa et al. (2011) RandomForestClassifier. Hyperparameters not mentioned are set to library defaults.

Hyperparameter choices for MLPs.

- Width $r$: $\{100, 300, 1000, 3000, 10000, 30000\}$. For deeper networks, the hidden layers are set to $r \times r$.
- Depth (number of hidden layers): $\{1, 2, 3\}$. In all of these settings, depth-1 networks are nearly uniformly outperformed by deeper ones of the same width.
- Sparsity of initialization: $s \in \{2, 4\}$, and also dense (uniform) initialization. 4-sparse initialization is nearly uniformly outperformed by 2-sparse (in agreement with our experiments on sparse parity).
- Weight decay: $\{0, 0.01\}$. In these settings, this hyperparameter had a minimal effect.
- Learning rate: $10^{-3}$
- Batch size: 256

*Note that Grinsztajn et al. (2022) instead transform each feature such that its empirical marginal distribution approximates a standard Gaussian.

- Training epochs: 100

Hyperparameter choices for RandomForestClassifier.

- `max_depth`: $\{-1, 2, 3, 4, 10\}$
- `max_features`: $\{1, 2, 3, 4, \text{sqrt}, \text{None}\}$
- `n_estimators`: $\{10, 100, 1000\}$

Larger widths. For all of the tasks except 2 and 9 (since these are significantly larger datasets), we include extra runs with even larger networks: depth-2 MLPs with non-uniform width (i.e. sequence of hidden layer dimensions) $\{(100000, 10000), (10000, 100000)\}$.

Plots in Figure C.3. For clarity of presentation, in the plots where we vary MLP width and initialization sparsity, we fix depth to 3 and sparsity level $s = 2$. Qualitative trends are similar for other settings. The gold "best MLP" curves show the best median-of-100 validation losses across all architectures in the search space.

Comparison with full-data baselines. To ensure that our baseline algorithm choices for MLPs and random forests are reasonable, we present a comparison with the results of Grinsztajn et al. (2022) (who performed extensive hyperparameter search) on the full datasets. These are shown as the dotted lines in Figure C.3, as well as Table C.2.

Results. We note the following findings from the results in Figure C.3 and Table C.2:

(2T) **Wide networks resist overfitting on small tabular datasets.** Like in the synthetic experiments, width yields monotonic end-to-end benefits for sample-efficient learning on these datasets. This suggests that the "parallel search + pruning" mechanisms analyzed in our

| Task | OpenML identifier | # features | # examples |
|---|---|---|---|
| 1 | credit | 11 | 16714 |
| 2 | electricity | 8 | 38474 |
| 3 | covertype | 11 | 566602 |
| 4 | pol | 27 | 10082 |
| 5 | house_16H | 17 | 13488 |
| 6 | MagicTelescope | 11 | 13376 |
| 7 | bank-marketing | 8 | 10578 |
| 8 | MiniBooNE | 51 | 72998 |
| 9 | Higgs | 25 | 940160 |
| 10 | eye_movements | 21 | 7608 |
| 11 | Diabetes130US | 8 | 71090 |
| 12 | jannis | 55 | 57580 |
| 13 | default-of-credit-card-clients | 21 | 13272 |
| 14 | Bioresponse | 420 | 3434 |
| 15 | california | 9 | 20634 |
| 16 | heloc | 23 | 10000 |

**Table C.1:** Metadata for the 16 tabular classification benchmarks, curated by Grinsztajn et al. (2022) (January 2023 version, benchmark suite ID 337) and publicly available via OpenML.

paper are empirically at play in these settings, and that these networks' capacity to overfit does not preclude nontrivial feature learning and generalization. These comparisons can be seen by comparing the colored curves (which represent depths 2 and 3) within each subplot in Figure C.3. In some (but not all) cases, these MLPs perform competitively with hyperparameter-tuned random forest classifiers.

(3T) **Sparse axis-aligned initialization sometimes improves end-to-end performance.** These comparisons can be seen by comparing vertically adjacent sparse vs. dense subplots in Figure C.3. This effect is especially pronounced on datasets which are downsampled to be orders of magnitude smaller. We believe that this class of drop-in replacements for standard initialization merits further investigation, and may contribute to closing the remaining performance gap between deep learning and tree ensembles on small tabular datasets.

| Task | Best MLP | | | Best RF | | | (Grinsztajn et al., 2022) | | |
|---|---|---|---|---|---|---|---|---|---|
| | full | 10% | 1% | full | 10% | 1% | MLP | RF | Best model |
| 1 | 24.5 | 27.8 | 34.2 | **22.0** | **22.9** | **24.8** | 22.9 | 23.6 | 22.5 (GBT) |
| 2 | 18.9 | 23.2 | 25.1 | **15.4** | **18.0** | **23.0** | 23.6 | 13.7 | 12.8 (XGB) |
| 3 | **6.3** | **13.7** | 23.2 | 19.1 | 18.9 | **21.4** | 21.6 | 17.1 | 17.1 (RF) |
| 4 | **1.4** | 4.2 | 12.3 | 2.1 | **4.0** | **10.0** | 6.3 | 1.9 | 1.7 (XGB) |
| 5 | 12.7 | 14.9 | 20.0 | **12.4** | **13.9** | **17.4** | 12.1 | 12.0 | 11.1 (XGB) |
| 6 | **14.6** | **16.5** | 23.1 | 14.8 | 16.8 | **21.9** | 14.6 | 14.5 | 13.9 (FTT) |
| 7 | 20.2 | 21.8 | 25.9 | **19.7** | **21.1** | **25.2** | 20.9 | 20.1 | 19.5 (GBT) |
| 8 | **7.2** | 10.0 | 14.4 | 7.7 | **8.6** | **10.5** | 6.7 | 7.3 | 6.2 (XGB) |
| 9 | **26.1** | **28.7** | 33.7 | 28.5 | **28.7** | **29.9** | 31.4 | 29.1 | 28.6 (XGB) |
| 10 | 39.9 | 43.9 | **46.3** | **35.8** | **43.0** | 47.8 | 41.8 | 34.8 | 33.4 (XGB) |
| 11 | **39.1** | **39.3** | 40.7 | **39.1** | 39.4 | **40.6** | 39.5 | 39.5 | 39.4 (XGB) |
| 12 | 23.0 | 26.1 | 28.7 | **22.0** | **23.8** | **27.2** | 25.5 | 22.7 | 22.0 (XGB) |
| 13 | 28.7 | 30.1 | 33.5 | **28.2** | **29.2** | **31.9** | 28.9 | 28.0 | 28.0 (RF) |
| 14 | **19.5** | **28.9** | **37.6** | 22.2 | 29.2 | 39.8 | 23.4 | 20.5 | 20.5 (RF) |
| 15 | 12.2 | 14.8 | 20.8 | **11.2** | **13.8** | **18.4** | 12.9 | 10.6 | 9.7 (XGB) |
| 16 | **27.1** | **28.0** | 31.3 | 27.8 | 28.6 | **31.0** | 27.8 | 28.1 | 27.4 (ResNet) |

**Table C.2:** Numerical comparisons for the tabular data experiments, to accompany Figure C.3. We report the median test error (%) over 100 i.i.d. training runs of MLPs and random forests. The 3 subcolumns in each group denote models trained on the full dataset and their {10%, 1%} downsampled counterparts. For all of these results, bootstrap 95% confidence intervals have width $< 2\%$. We observe that wide and/or sparsely-initialized MLPs are competitive with tree-based methods. In the rightmost 3 columns, we provide test errors for the same tasks, reported by (Grinsztajn et al., 2022) (GBT = gradient boosted tree, XGB = XGBoost, FTT = Feature Tokenizer + Transformer (Gorishniy et al., 2021)). Note that our cross-validation protocol differs slightly (to handle variance incurred by downsampling), which may explain performance discrepancies. We include these only to illustrate that our full-data accuracies are commensurate with those in prior works focused exclusively on benchmarking models for tabular data.

### C.3.4 Software, compute infrastructure, and resource costs

GPU-accelerated training and evaluation pipelines were implemented in PyTorch (Paszke et al., 2019). Each training run was performed on one GPU in an internal cluster, with NVIDIA P40, P100, V100, and RTX A6000 GPUs. A single $T = 10^5$ training run took 10 seconds on average (with early termination for the vast majority of grid sweeps); across all of the results in this paper, around $2 \times 10^5$ training runs were performed, in a total of 600 GPU-hours. Note that the precise evaluation of test error (at batch size $10^4$) constitutes a significant portion of the computational cost; this necessitated mindful GPU parallelization.

# D

# Feature Emergence

## D.1 Further Related Work

Two closely related works to ours are Gromov (2023) and Bronstein et al. (2022). Gromov (2023) provides an analytic construction of various two-layer quadratic networks that can solve the modular addition task. The construction used in the proof of Theorem 7.7 is a special case of the given scheme. Bronstein et al. (2022)shows that all max margin solutions of a one-hidden-layer ReLU net-

work (with fixed top weights) trained on read-once DNFs have neurons which align with clauses. However, the proof techniques are significantly different. For any given neural network not satisfying the desired conditions ((neurons aligning with the clauses), Bronstein et al. (2022) construct a perturbed network satisfying the conditions which exhibits a better margin. We rely on the max-min duality for certifying a maximum margin solution, as shown in Section 7.3.3.

**Margin maximization.** One branch of results on margin maximization in neural networks involve proving that the optimization of neural networks leads to an implicit bias towards margin maximization. Soudry et al. (2018) show that logistic regression converges in direction to the max margin classifier. Wei et al. (2019a) prove that the global optimum of weakly-regularized cross-entropy loss on homogeneous networks reaches the max margin. Similarly, Lyu & Li (2019) and Ji & Telgarsky (2020) show that in homogeneous networks, even in the absence of explicit regularization, if loss becomes low enough then the weights will tend in direction to a KKT point of the max margin optimization objective. This implies margin maximization in deep linear networks, although it is not necessarily the global max margin (Vardi et al., 2022). Chizat & Bach (2020) prove that infinite-width 2-homogeneous networks with mean field initialization will converge to the global max margin solution. In a different setting, Lyu et al. (2021) and Frei et al. (2022b) show that the margin is maximized when training leaky-ReLU one hidden layer networks with gradient flow on linearly separable data, given certain assumptions on the input (eg. presence of symmetries, near-orthogonality). For more on studying inductive biases in neural networks, refer to Vardi (2023).

Numerous other works do not focus on neural network dynamics and instead analyze properties of solutions with good margins (Bartlett, 1996). For instance, Frei et al. (2023) show that the maximum margin KKT points have "benign overfitting" properties. The works by Lyu et al. (2021), Morwani et al. (2023a) and Frei et al. (2023) show that max margin implies linear decision boundary for solutions. Gunasekar et al. (2018) show that under certain assumptions, gradient descent on depth-two linear convolutional networks (with weight-sharing in first layer) converges not to the

standard $L_2$ max margin, but to the global max margin with respect to the $L_1$ norm of the Fourier transform of the predictor. Our work follows a similar vein, in which we characterize max margin features in our setting and relate this to trained networks via results from Wei et al. (2019a).

**Training on algebraic tasks and mechanistic interpretability.** Studying neural networks trained on algebraic tasks has offered insights into their training dynamics and inductive biases, with the simpler setting lending a greater ease of understanding. One such example is the task of modular addition, which was studied in Power et al. (2021) in their study of grokking, leading to multiple follow-up works (Liu et al., 2022b, 2023). Another example is the problem of learning parities for neural networks, which has been investigated in numerous works (Daniely & Malach, 2020; Zhenmei et al., 2022; Frei et al., 2022a; Barak et al., 2022; Edelman et al., 2024a). Other mathematical tasks like learning addition have been used to investigate whether models possess algorithmic reasoning capabilities (Saxton et al., 2018; Hendrycks et al., 2021; Lewkowycz et al., 2022).

The area of mechanistic interpretability aims to understand the internal representations of individual neural networks by analyzing its weights. This form of analysis has been applied to understand the motifs and features of neurons in *circuits*— particular subsets of a neural network— in computer vision models (Olah et al., 2020; Cammarata et al., 2020) and more recently in language models (Elhage et al., 2021; Olsson et al., 2022). However, the ability to fully reverse engineer a neural network is extremely difficult for most tasks and architectures. Some work in this area has shifted towards finding small, toy models that are easier to interpret, and employing labor intensive approaches to reverse-engineering specific features and circuits in detail(Elhage et al., 2022). In Nanda et al. (2023), the authors manage to fully interpret how one-layer transformers implement modular addition and use this knowledge to define progress measures that precede the grokking phase transition which was previously observed to occur for this task (Power et al., 2021). Chughtai et al. (2023) extends this analysis to learning composition for various finite groups, and identifies analogous results and progress measures. In this work, we show that these empirical findings can be

analytically explained via max margin analysis, due to the implicit bias of gradient descent towards margin maximization.

## D.2 Experimental details

In this section, we will provide the hyperparameter settings for various experiments in the paper.

### D.2.1 Cyclic Group

We train a 1-hidden layer network with $m = 500$, using gradient descent on the task of learning modular addition for $p = 71$ for 40000 steps. The initial learning rate of the network is 0.05, which is doubled on the steps - $[1e3, 2e3, 3e3, 4e3, 5e3, 6e3, 7e3, 8e3, 9e3, 10e3]$. Thus, the final learning rate of the network is 51.2. This is done to speed up the training of the network towards the end, as the gradient of the loss goes down exponentially. For quadratic network, we use a $L_{2,3}$ regularization of $1e-4$. For ReLU network, we use a $L_2$ regularization of $1e-4$.

### D.2.2 Sparse parity

We train a 1-hidden layer quadratic network with $m = 40$ on $(10, 4)-$sparse parity task. It is trained by gradient descent for 30000 steps with a learning rate of 0.1 and $L_{2,5}$ regularization of $1e-3$.

### D.2.3 General Groups

The hyperparameters for various groups $S_3$, $S_4$, and $S_5$ are provided in subsections below.

### S3

We train a 1-hidden layer quadratic network with $m = 30$, using gradient descent for 50000 steps, with a $L_{2,3}$ regularization of $1e-7$. The initial learning rate is 0.05, which is doubled on the steps - $[200, 400, 600, 800, 1000, 1200, 1400, 1600, 1800, 2000, 2200, 2400, 2600, 5000, 10000]$. Thus, the final learning rate is 1638.4. This is done to speed up the training of the network towards the end, as the gradient of the loss goes down exponentially.

### S4

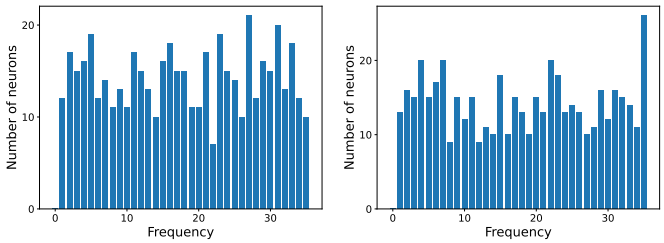
We train a 1-hidden layer quadratic network with $m = 200$, using gradient descent for 50000 steps, with a $L_{2,3}$ regularization of $1e-7$. The initial learning rate is 0.05, which is doubled on the steps - $[200, 400, 600, 800, 1000, 1200, 1400, 1600, 1800, 2000, 2200, 2400, 2600, 5000, 10000]$. Thus, the final learning rate is 1638.4. This is done to speed up the training of the network towards the end, as the gradient of the loss goes down exponentially.

### S5

We train a 1-hidden layer quadratic network with $m = 2000$, using stochastic gradient descent for 75000 steps, with a batch size of 1000 and $L_{2,3}$ regularization of $1e-5$. The initial learning rate is 0.05, which is doubled on the steps - $[3000, 6000, 9000, 12000, 15000, 18000, 21000, 24000]$. Thus, the final learning rate is 12.8. This is done to speed up the training of the network towards the end, as the gradient of the loss goes down exponentially.

## D.3 Additional Experiments

The distribution of neurons of a particular frequency for the modular addition case is shown in Figure D.1. As can be seen, for both ReLU and quadratic activation, the distribution is close to




**(a)** ReLU activation **(b)** Quadratic activation

**Figure D.1:** Final distribution of the neurons corresponding to a particular frequency in (a) ReLU network trained with $L_2$ regularization and (b) Quadratic network trained with $L_{2,3}$ regularization. Similar to our construction, the final distribution across frequencies is close to uniform.

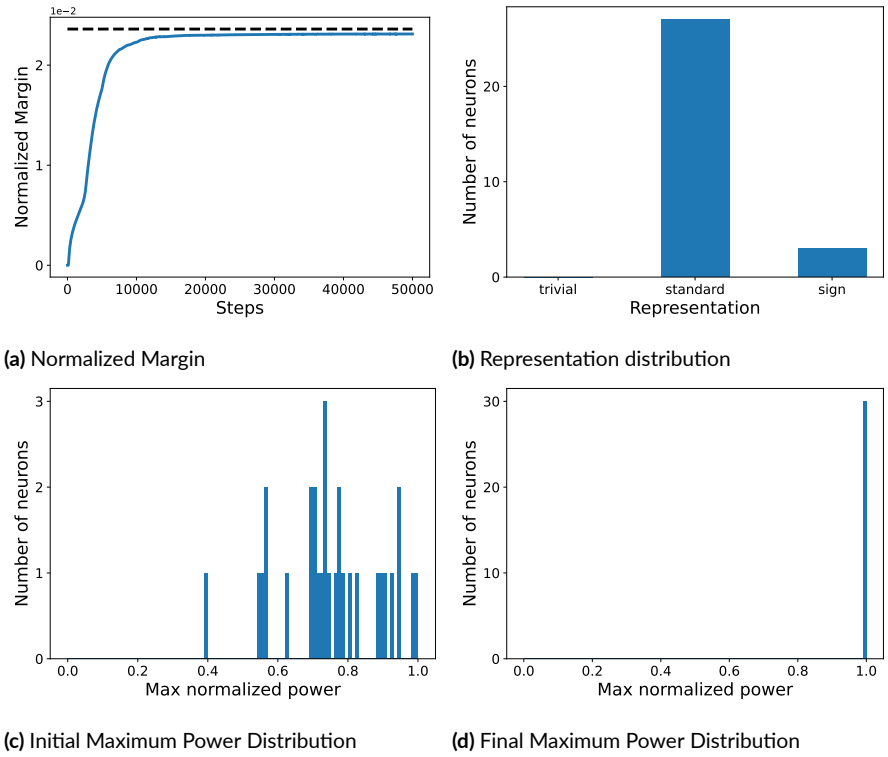
uniform.

Experimental results for other symmetric groups $S_3$ and $S_4$ in Figures D.2 and D.3 respectively. We observe the same max margin features as stated in Theorem 7.9 and the $L_{2,3}$ margin approaches the theoretical max margin that we have predicted.

## D.4 Alternative construction

To argue why the problem of finding correctly classifying networks is overdetermined, we present an alternative construction (which applies to general groups) that does not have an "interesting" Fourier spectrum or any behavioral similarity to the solutions reached by standard training.

For any function $r : [n]^2 \to [n]$, there exists a neural network parameterized by $\theta$ of the form considered in Sections 7.4 and 7.6 with $2p^2$ neurons such that $f(\theta, (a, b))[c] = \mathbf{1}_{c=r(a,b)}$ and that is "dense" in the Fourier spectrum. For each pair $(a, b)$ we use two neurons given by $\{u, v, w\}$ and $\{u', v', w'\}$, where $u_i = u'_i = \mathbf{1}_{i=a}$, $v_i = \mathbf{1}_{i=b}$, $v'_i = -1_{i=b}$, $w_i = \mathbf{1}_{i=r(a,b)}/4$and $w'_i = -\mathbf{1}_{i=r(a,b)}/4$. When adding together the outputs for these two neurons, for an input of $(i, j)$ we get $k^{\text{th}}$ logit equal




**(a)** Normalized Margin

**(b)** Representation distribution

**(c)** Initial Maximum Power Distribution

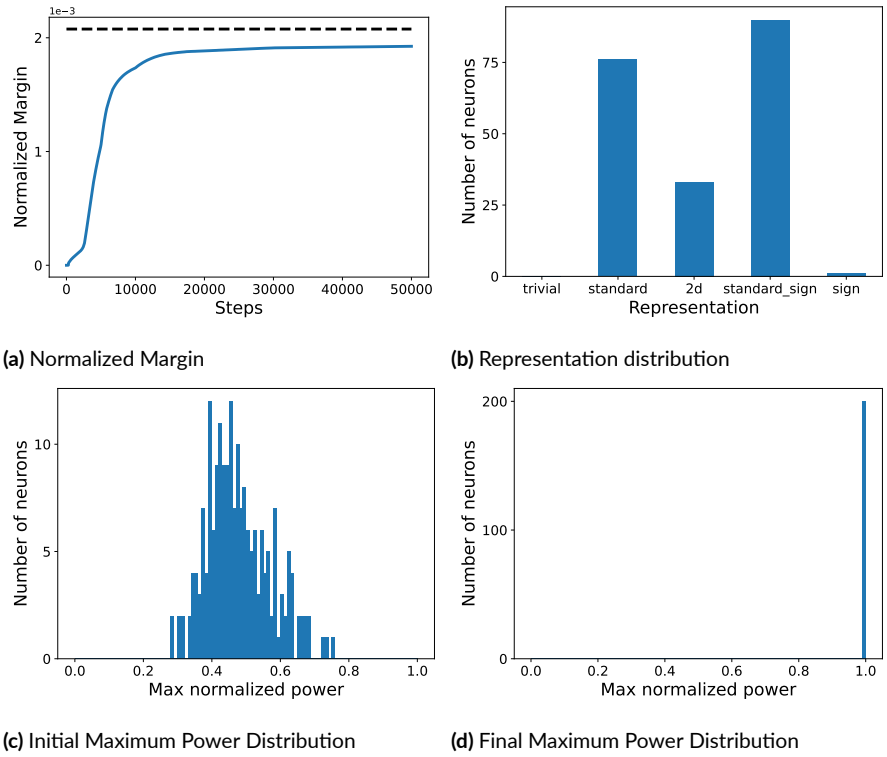
**(d)** Final Maximum Power Distribution

**Figure D.2:** This figure demonstrates the training of a 1-hidden layer quadratic network on the symmetric group $S3$ with $L_{2,3}$ regularization. (a) Evolution of the normalized $L_{2,3}$ margin of the network with training. It approaches the theoretical maximum margin that we predict. (b) Distribution of neurons spanned by a given representation. Higher dimensional representations have more neurons as given by our construction. (c) and (d) Maximum normalized power is given by $\frac{\max \hat{u}[i]^2}{\sum_j \hat{u}[j]^2}$ where $\hat{u}[i]$ refers to the component of weight $u$ along $i^{th}$ representation. Initially, it's random, but towards the end of training, all neurons are concentrated in a single representation, as predicted by maximum margin.

to:

$$\frac{1}{4}\left((\mathbf{1}_{i=a}+\mathbf{1}_{j=b})^2\mathbf{1}_{k=r(i,j)}-(\mathbf{1}_{i=a}-\mathbf{1}_{j=b})^2\mathbf{1}_{k=r(i,j)}\right)=\mathbf{1}_{i=a}\mathbf{1}_{j=b}\mathbf{1}_{k=r(a,b)}$$

Hence, these two norms help "memorize" the output for $(a,b)$ while not influencing the output for any other input, so when summing together all these neurons we get an $f$ with the aforementioned property. Note that all the vectors used are (up to sign) one-hot encodings and thus have an uniform norm in the Fourier spectrum. This is to show that Fourier sparsity is not present in any correct




**(a)** Normalized Margin

**(b)** Representation distribution

**(c)** Initial Maximum Power Distribution

**(d)** Final Maximum Power Distribution

**Figure D.3:** This figure demonstrates the training of a 1-hidden layer quadratic network on the symmetric group $S4$ with $L_{2,3}$ regularization. (a) Evolution of the normalized $L_{2,3}$ margin of the network with training. It approaches the theoretical maximum margin that we predict. (b) Distribution of neurons spanned by a given representation. Higher dimensional representations have more neurons as given by our construction. (c) and (d) Maximum normalized power is given by $\frac{\max \hat{u}[i]^2}{\sum_j \hat{u}[j]^2}$ where $\hat{u}[i]$ refers to the component of weight $u$ along $i^{th}$ representation. Initially, it is random, but towards the end of training, all neurons are concentrated on a single representation, as predicted by the maximum margin analysis.

classifier.

## D.5 Proofs for the Theoretical Approach

For ease of the reader, we will first restate Equations 7.1 and 7.2.

$$q^* \in \arg\min_{q \in \mathcal{P}(D)} \mathop{\mathbb{E}}_{(x,y)\sim q} [g(\theta^*, x, y)]$$

$$\theta^* \in \arg\max_{\theta \in \Theta} \mathop{\mathbb{E}}_{(x,y)\sim q^*} [g(\theta, x, y)]$$

We will first provide the proof of Lemma 7.2.

*Lemma.* If a pair $(\theta^*, q^*)$ satisfies Equations 7.1 and 7.2, then

$$\theta^* \in \arg\max_{\theta \in \Theta} \min_{(x,y)\in D} g(\theta, x, y)$$

*Proof.* First, using max-min inequality, we have:

$$\max_{\theta\in\Theta} \min_{(x,y)\in D} g(\theta, x, y) = \max_{\theta\in\Theta} \min_{q\in\mathcal{P}(D)} \mathop{\mathbb{E}}_{(x,y)\sim q} [g(\theta, x, y)] \leq$$

$$\min_{q\in\mathcal{P}(D)} \max_{\theta\in\Theta} \mathop{\mathbb{E}}_{(x,y)\sim q} [g(\theta, x, y)]$$

On the other hand, it also holds that:

$$\min_{q\in\mathcal{P}(D)} \max_{\theta\in\Theta} \mathop{\mathbb{E}}_{(x,y)\sim q} [g(\theta, x, y)] \leq \max_{\theta\in\Theta} \mathop{\mathbb{E}}_{(x,y)\sim q^*} [g(\theta, x, y)] =$$

$$\mathop{\mathbb{E}}_{(x,y)\sim q^*} [g(\theta^*, x, y)] = \min_{q\in\mathcal{P}(D)} \mathop{\mathbb{E}}_{(x,y)\sim q} [g(\theta^*, x, y)] \leq$$

$$\max_{\theta\in\Theta} \min_{q\in\mathcal{P}(D)} \mathop{\mathbb{E}}_{(x,y)\sim q} [g(\theta, x, y)]$$

where the first equality follows from Equation 7.2 and the second follows from Equation 7.1. Putting these inequalities together it follows that all of the above terms are equal (and, thus we get a minimax theorem). In particular, $\theta^* \in \arg\max_{\theta\in\Theta} \min_{(x,y)\in D} g(\theta, x, y)$ as desired. □

### D.5.1 Binary Classification

Now, we will provide the proof of Lemma 7.3.

*Lemma.* Let $\Theta = \{\theta : \|\theta\|_{a,b} \leq 1\}$ and $\Theta^*_q = \arg\max_{\theta\in\Theta} \mathbb{E}_{(x,y)\sim q} [g(\theta, x, y)]$. Similarly, let

$\Omega = \{\omega : \|\omega\|_a \leq 1\}$ and $\Omega_q^* = \arg\max_{\omega\in\Omega} \mathbb{E}_{(x,y)\sim q}[\psi(\omega, x, y)]$. Then, for binary classification, the following holds:

- **Single neuron optimization**: Any $\theta \in \Theta_q^*$ has directional support only on $\Omega_q^*$.
- **Using multiple neurons**: If $b = \nu$ and $\omega_1^*, \ldots, \omega_m^* \in \Omega_q^*$, then $\theta = \{\lambda_i \omega_i^*\}_{i=1}^m$ with $\sum \lambda_i^\nu = 1, \lambda_i \geq 0$ belongs to $\Theta_q^*$.

*Proof.* Let $\gamma = \max_{\omega\in\Omega} \mathbb{E}_{(x,y)\sim q^*}[\psi(\omega, x, y)]$ and take any $\theta = \{\omega_i\}_{i=1}^m$. Then:

$$\mathbb{E}_{(x,y)\sim q^*}[g(\theta, x, y)] = \mathbb{E}_{(x,y)\sim q^*}\left[\sum_{i=1}^m \psi(\omega_i)\right] = \sum_{i=1}^m \|\omega_i\|_a^\nu \mathbb{E}_{(x,y)\sim q^*}\left[\psi\left(\frac{\omega_i}{\|\omega_i\|_a}\right)\right] \leq$$
$$\gamma \sum_{i=1}^m \|\omega_i\|_a^\nu \leq \gamma \max_{\substack{w\in\mathbb{R}^m \\ \|w\|_b\leq 1}} \|w\|_\nu^\nu$$

with equality when $\frac{\omega_i}{\|\omega_i\|_a} \in \arg\max_{\omega\in\Omega} \mathbb{E}_{(x,y)\sim q^*}[\psi(\omega, x, y)]$ for all $i$ with $\omega_i \neq 0$ and the $L_a$ norms of $\omega$s respect $\{\|\omega_i\|_a\}_{i=1}^m \in \arg\max_{\|w\|_b\leq 1} \|w\|_\nu^\nu$. Since there exists equality for this upper bound, these two criteria define precisely $\arg\max_{\theta\in\Theta} \mathbb{E}_{(x,y)\sim q^*}[g(\theta, x, y)]$. Hence, we proved the first part of the statement by first criterion. For the second, note that when $b = \nu$, one can choose any vector of norms for $\omega$ with $L_b$ norm of 1 (since $\|w\|_\nu^\nu = \|w\|_b^b \leq 1$), such as $\lambda$ - this concludes the proof of the second part. □

*Remark.* Note that the analysis in above proof can be used to compute optimal norms for $b \neq \nu$ as well - however, for any such $b$ we would not get the same flexibility to build a $\theta^*$ satisfying Equation 7.1. This is the reason behind choosing $b = \nu$.

Now, we will provide the proof of Lemma 7.4.

*Lemma.* Let $\Theta = \{\theta : \|\theta\|_{a,b} \leq 1\}$ and $\Theta_q^* = \arg\max_{\theta\in\Theta} \mathbb{E}_{(x,y)\sim q}[g(\theta, x, y)]$. Similarly, let $\Omega = \{\omega : \|\omega\|_a \leq 1\}$ and $\Omega_q^* = \arg\max_{\omega\in\Omega} \mathbb{E}_{(x,y)\sim q}[\psi(\omega, x, y)]$. For the task of binary classification, if

there exists $\{\theta^*, q^*\}$ satisfying Equation 7.1 and 7.2, then any $\hat{\theta} \in \arg\max_{\theta\in\Theta} \min_{(x,y)\in D} g(\theta, x, y)$ satisfies the following:

- $\hat{\theta}$ has directional support only on $\Omega^*_{q^*}$.
- For any $(x_1, y_1) \in \text{spt}(q^*), f(\hat{\theta}, x_1, y_1) - f(\hat{\theta}, x_1, y_1') = \gamma^*$, where $y_1' \neq y_1$, i.e, all points in the support of $q^*$ are on the margin for any maximum margin solution.

*Proof.* Let $\gamma^* = \max_{\theta\in\Theta} \min_{(x,y)\in D} g(\theta, x, y)$. Then, by Lemma 7.2, $\gamma^* = \mathbb{E}_{(x,y)\sim q^*} g(\theta^*, x, y)$.

Consider any $\hat{\theta} \in \arg\max_{\theta\in\Theta} \min_{(x,y)\in D} g(\theta, x, y)$. This means, that $\min_{(x,y)\in D} g(\hat{\theta}, x, y) = \gamma^*$. This implies that $\mathbb{E}_{(x,y)\sim q^*} g(\hat{\theta}, x, y) \geq \gamma^*$. However, by Equation 7.2, $\max_{\theta\in\Theta} \mathbb{E}_{(x,y)\sim q^*} g(\theta, x, y) = \gamma^*$. This implies that $\mathbb{E}_{(x,y)\sim q^*} g(\hat{\theta}, x, y) = \gamma^*$. Thus, $\hat{\theta}$ is also a maximizer of $\mathbb{E}_{(x,y)\sim q^*} g(\theta, x, y)$, and thus by Lemma 7.3, it only has directional support on $\Omega^*_{q^*}$.

Moreover, as $\mathbb{E}_{(x,y)\sim q^*} g(\hat{\theta}, x, y) = \gamma^*$, thus, for any $(x_1, y_1) \in \text{spt}(q^*), f(\hat{\theta}, x_1, y_1) - f(\hat{\theta}, x_1, y_1') = \gamma^*$, where $y_1' \neq y_1$. □

### D.5.2 Multi-Class Classification

We will first provide the proof of Lemma 7.5.

*Lemma.* Let $\Theta = \{\theta : \|\theta\|_{a,b} \leq 1\}$ and $\Theta'^*_q = \arg\max_{\theta\in\Theta} \mathbb{E}_{(x,y)\sim q} [g'(\theta, x, y)]$. Similarly, let $\Omega = \{\omega : \|\omega\|_a \leq 1\}$ and $\Omega'^*_q = \arg\max_{\omega\in\Omega} \mathbb{E}_{(x,y)\sim q} [\psi'(\omega, x, y)]$. Then:

- **Single neuron optimization**: Any $\theta \in \Theta'^*_q$ has directional support only on $\Omega'^*_q$.
- **Using multiple neurons**: If $b = \nu$ and $\omega_1^*, \ldots, \omega_m^* \in \Omega'^*_q$, then $\theta = \{\lambda_i \omega_i^*\}_{i=1}^m$ with $\sum \lambda_i^\nu = 1, \lambda_i \geq 0$ belongs to $\Theta'^*_q$.

*Proof.* The proof follows the same strategy as the proof of Lemma 7.3 (Section D.5.1), following the linearity of $g'$. □

Now, for ease of the reader, we will first restate Equation 7.3 and condition **C.1**.

$$\theta^* \in \arg\max_{\theta \in \Theta} \mathop{\mathbb{E}}_{(x,y)\sim q^*} \left[ g'(\theta, x, y) \right].$$

**C.1** For any $(x, y) \in \text{spt}(q^*)$, it holds that $g'(\theta^*, x, y) = g(\theta^*, x, y)$. This translates to any label with non-zero weight being one of the incorrect labels where $f$ is maximized: $\{\ell \in \mathcal{Y} \setminus \{y\} : \tau(x, y)[\ell] > 0\} \subseteq \arg\max_{\ell \in \mathcal{Y}\setminus\{y\}} f(\theta^*, x)[\ell]$.

We will now the provide the proof of Lemma 7.6.

*Lemma.* Let $\Theta = \{\theta : \|\theta\|_{a,b} \leq 1\}$ and $\Theta'^*_q = \arg\max_{\theta\in\Theta} \mathbb{E}_{(x,y)\sim q} [g'(\theta, x, y)]$. Similarly, let $\Omega = \{\omega : \|\omega\|_a \leq 1\}$ and $\Omega'^*_q = \arg\max_{\omega\in\Omega} \mathbb{E}_{(x,y)\sim q} [\psi'(\omega, x, y)]$. If $\exists\{\theta^*, q^*\}$ satisfying Equations 7.1 and 7.3, and **C.1** holds, then:

- $\theta^* \in \arg\max_{\theta\in\Theta} g(\theta, x, y)$
- Any $\hat{\theta} \in \arg\max_{\theta\in\Theta} \min_{(x,y)\in D} g(\theta, x, y)$ satisfies the following:
  - $\hat{\theta}$ has directional support only on $\Omega'^*_{q^*}$.
  - For any $(x_1, y_1) \in \text{spt}(q^*), f(\hat{\theta}, x_1, y_1) - \max_{y'\in\mathcal{Y}\setminus\{y_1\}} f(\hat{\theta}, x_1, y'_1) = \gamma^*$, i.e, all points in the support of $q^*$ are on the margin for any maximum margin solution.

*Proof.* For the first part, we will show that $\{\theta^*, q^*\}$ satisfy Equations 7.1 and 7.2, and then it follows from Lemma 7.2. As we have already assumed these satisfy Equation 7.1, we will show that they satisfy Equation 7.2.

Note that $g'(\theta, x, y) \geq g(\theta, x, y)$. Thus,

$$\begin{aligned}
\mathop{\mathbb{E}}_{(x,y)\sim q^*}[g(\theta^*, x, y)] &\leq \max_{\theta\in\Theta} \mathop{\mathbb{E}}_{(x,y)\sim q^*}[g(\theta, x, y)] \\
&\leq \max_{\theta\in\Theta} \mathop{\mathbb{E}}_{(x,y)\sim q^*}[g'(\theta, x, y)] \\
&= \mathop{\mathbb{E}}_{(x,y)\sim q^*}[g'(\theta^*, x, y)]
\end{aligned}$$

where the second inequality follows as $g' \geq g$ and the last equality follows as $\theta^*$ satisfies Equation 7.3. Now, as the pair also satisfies **C.1**, therefore $\mathbb{E}_{(x,y)\sim q^*}[g(\theta^*, x, y)] = \mathbb{E}_{(x,y)\sim q^*}[g'(\theta^*, x, y)]$. This means, that all inequalities in the above chain must be equality. Thus,

$$\theta^* \in \arg\max_{\theta\in\Theta} \mathop{\mathbb{E}}_{(x,y)\sim q^*}[g(\theta, x, y)].$$

Thus, the pair $\{\theta^*, q^*\}$ satisfies Equation 7.1 and 7.2, and thus by Lemma 7.2,

$$\theta^* \in \arg\max_{\theta\in\Theta} g(\theta, x, y).$$

Let $\gamma^* = \max_{\theta\in\Theta} \min_{(x,y)\in D} g(\theta, x, y)$. Then, $\gamma^* = \mathbb{E}_{(x,y)\sim q^*} g(\theta^*, x, y)$. Consider any $\hat{\theta} \in \arg\max_{\theta\in\Theta} \min_{(x,y)\in D} g(\theta, x, y)$. This means, that $\min_{(x,y)\in D} g(\hat{\theta}, x, y) = \gamma^*$. This implies that $\mathbb{E}_{(x,y)\sim q^*} g(\hat{\theta}, x, y) \geq \gamma^*$. Since $g' \geq g$, it then folllows that $\mathbb{E}_{(x,y)\sim q^*} g'(\hat{\theta}, x, y) \geq \gamma^*$.

However, by Equation 7.3 and **C.1**, $\max_{\theta\in\Theta} \mathbb{E}_{(x,y)\sim q^*} g'(\theta, x, y) = \mathbb{E}_{(x,y)\sim q^*} g'(\theta^*, x, y) = \mathbb{E}_{(x,y)\sim q^*} g(\theta^*, x, y) = \gamma^*$. This implies that $\mathbb{E}_{(x,y)\sim q^*} g'(\hat{\theta}, x, y) = \gamma^*$. Thus, $\hat{\theta}$ is also a maximizer of $\mathbb{E}_{(x,y)\sim q^*} g'(\theta, x, y)$, and thus by Lemma 7.5, it only has directional support on $\Omega'^*_{q^*}$.

Moreover, as $\min_{(x,y)\in D} g(\hat{\theta}, x, y) = \gamma^*$, therefore, $\mathbb{E}_{(x,y)\sim q^*} g(\hat{\theta}, x, y) \geq \gamma^*$. However, as $g' \geq g$, therefore, $\mathbb{E}_{(x,y)\sim q^*} g(\hat{\theta}, x, y) \leq \mathbb{E}_{(x,y)\sim q^*} g'(\hat{\theta}, x, y) = \gamma^*$, as shown above. Thus, $\mathbb{E}_{(x,y)\sim q^*} g(\hat{\theta}, x, y) = \gamma^*$. Thus, we have $f(\hat{\theta}, x_1, y_1) - \max_{y'\in\mathcal{Y}\setminus\{y_1\}} f(\hat{\theta}, x_1, y'_1) = g(\hat{\theta}, x_1, y_1) = \gamma^*$

for any $(x_1, y_1) \in \text{spt}(q^*)$. □

## D.6 Proofs for cyclic groups(Theorem 7.7)

### D.6.1 Proof that Equation 7.3 is satisfied

*Proof.* Let

$$\eta_{u,v,w}(\delta) := \mathop{\mathbb{E}}_{a,b}\left[(u(a)+v(b))^2 w(a+b-\delta)\right].$$

We wish to find the solution to the following mean margin maximization problem:

$$\mathop{\arg\max}_{u,v,w:\|u\|^2+\|v\|^2+\|w\|^2\le 1}\left(\eta_{u,v,w}(0) - \mathop{\mathbb{E}}_{\delta\neq 0}\left[\eta_{u,v,w}(\delta)\right]\right) = \frac{p}{p-1}\left(\eta_{u,v,w}(0) - \mathop{\mathbb{E}}_{\delta}\left[\eta_{u,v,w}(\delta)\right]\right). \tag{D.1}$$

First, note that $\mathbb{E}_c[w(c)] = 0$, because shifting the mean of $w$ does not affect the margin. It follows that

$$\mathop{\mathbb{E}}_{a,b}\left[(u(a)^2 w(a+b-\delta)\right] = \mathop{\mathbb{E}}_{a}\left[u(a)^2 \mathop{\mathbb{E}}_{b}[w(a+b-\delta)]\right] = \mathop{\mathbb{E}}_{a}\left[u(a)^2 \mathop{\mathbb{E}}_{b}[w(b)]\right] = 0,$$

and similarly for the $v(b)^2$ component of $\eta$, so we can rewrite (D.1) as

$$\mathop{\arg\max}_{u,v,w:\|u\|^2+\|v\|^2+\|w\|^2\le 1} \frac{2p}{p-1}\left(\tilde{\eta}_{u,v,w}(0) - \mathop{\mathbb{E}}_{\delta}\left[\tilde{\eta}_{u,v,w}(\delta)\right]\right),$$

where

$$\tilde{\eta}_{u,v,w}(\delta) := \mathop{\mathbb{E}}_{a,b}\left[u(a)v(b)w(a+b-\delta)\right].$$

Let $\rho := e^{2\pi i/p}$, and let $\hat{u}, \hat{v}, \hat{w}$ be the discrete Fourier transforms of $u$, $v$, and $w$ respectively. Then

we have:

$$\begin{aligned}
\tilde{\eta}_{u,v,w}(\delta) &= \mathop{\mathbb{E}}_{a,b}\left[\left(\frac{1}{p}\sum_{j=0}^{p-1}\hat{u}(j)\rho^{ja}\right)\left(\frac{1}{p}\sum_{k=0}^{p-1}\hat{v}(k)\rho^{kb}\right)\left(\frac{1}{p}\sum_{\ell=0}^{p-1}\hat{w}(\ell)\rho^{\ell(a+b-\delta)}\right)\right] \\
&= \frac{1}{p^3}\sum_{j,k,\ell}\hat{u}(j)\hat{v}(k)\hat{w}(\ell)\rho^{-\ell\delta}\left(\mathop{\mathbb{E}}_{a}\rho^{(j+\ell)a}\right)\left(\mathop{\mathbb{E}}_{b}\rho^{(k+\ell)b}\right) \\
&= \frac{1}{p^3}\sum_{j}\hat{u}(j)\hat{v}(j)\hat{w}(-j)\rho^{j\delta} \qquad \text{(only terms where } j+\ell = k+\ell = 0 \text{ survive)}
\end{aligned}$$

Hence, we need to maximize

$$\frac{2p}{p-1}(\tilde{\eta}_{u,v,w}(0) - \mathop{\mathbb{E}}_{\delta}\left[\tilde{\eta}_{u,v,w}(\delta)\right]) \tag{D.2}$$

$$= \frac{2p}{p-1}\left(\frac{1}{p^3}\sum_{j}\hat{u}(j)\hat{v}(j)\hat{w}(-j) - \frac{1}{p^3}\sum_{j}\hat{u}(j)\hat{v}(j)\hat{w}(-j)(\mathop{\mathbb{E}}_{\delta}\rho^{j\delta})\right)$$

$$= \frac{2}{(p-1)p^2}\sum_{j\neq 0}\hat{u}(j)\hat{v}(j)\hat{w}(-j). \tag{D.3}$$

We have arrived at the crux of why any max margin solution must be sparse in the Fourier domain: in order to maximize expression D.3, we must concentrate the mass of $\hat{u}$, $\hat{v}$, and $\hat{w}$ on the same frequencies, the fewer the better. We will now work this out carefully. Since $u, v, w$ are real-valued, we have

$$\hat{u}(-j) = \overline{\hat{u}(j)}, \hat{v}(-j) = \overline{\hat{v}(j)}, \hat{w}(-j) = \overline{\hat{w}(j)}$$

for all $j \in \mathbb{Z}_p$. Let $\theta_u, \theta_v, \theta_w \in [0, 2\pi)^p$ be the phase components of $u, v, w$ respectively; so, e.g., for $\hat{u}$:

$$\hat{u}(j) = |\hat{u}(j)|\exp(i\theta_u(j)).$$

Then, for odd $p$, expression D.3 becomes:

$$\begin{aligned}
&\frac{2}{(p-1)p^2}\sum_{j=1}^{(p-1)/2}\left[\hat{u}(j)\hat{v}(j)\overline{\hat{w}(j)}+\overline{\hat{u}(j)\hat{v}(j)}\hat{w}(j)\right]\\
&=\frac{2}{(p-1)p^2}\sum_{j=1}^{(p-1)/2}|\hat{u}(j)||\hat{v}(j)||\hat{w}(j)|\left[\exp(i(\theta_u(j)+\theta_v(j)-\theta_w(j))+\exp(i(-\theta_u(j)-\theta_v(j)+\theta_w(j))\right]\\
&=\frac{4}{(p-1)p^2}\sum_{j=1}^{(p-1)/2}|\hat{u}(j)||\hat{v}(j)||\hat{w}(j)|\cos(\theta_u(j)+\theta_v(j)-\theta_w(j)).
\end{aligned}$$

Thus, we need to optimize:

$$\max_{u,v,w:\|u\|^2+\|v\|^2+\|w\|^2\le 1}\frac{4}{(p-1)p^2}\sum_{j=1}^{(p-1)/2}|\hat{u}(j)||\hat{v}(j)||\hat{w}(j)|\cos(\theta_u(j)+\theta_v(j)-\theta_w(j)). \tag{D.4}$$

By Plancherel's theorem, the norm constraint is equivalent to

$$\|\hat{u}\|^2+\|\hat{v}\|^2+\|\hat{w}\|^2\le p,$$

so the choice of $\theta_u(j),\theta_v(j),\theta_w(j)$ is unconstrained. Therefore, we can (and must) choose them to satisfy $\theta_u(j)+\theta_v(j)=\theta_w(j)$, so that $\cos(\theta_u(j)+\theta_v(j)-\theta_w(j))=1$ is maximized for each $j$ (unless the amplitude part of the $j$th term is 0, in which case the phase doesn't matter). The problem is thus further reduced to:

$$\max_{|\hat{u}|,|\hat{v}|,|\hat{w}|:\|\hat{u}\|^2+\|\hat{v}\|^2+\|\hat{w}\|^2\le p}\frac{4}{(p-1)p^2}\sum_{j=1}^{(p-1)/2}|\hat{u}(j)||\hat{v}(j)||\hat{w}(j)|. \tag{D.5}$$

By the inequality of quadratic and geometric means,

$$|\hat{u}(j)||\hat{v}(j)||\hat{w}(j)|\le\left(\frac{|\hat{u}(j)|^2+|\hat{v}(j)|^2+|\hat{w}(j)|^2}{3}\right)^{3/2}. \tag{D.6}$$

Let $z : \{1, \dots, \frac{p-1}{2}\} \to \mathbb{R}$ be defined as $z(j) := |\hat{u}(j)|^2 + |\hat{v}(j)|^2 + |\hat{w}(j)|^2$. Then, since we must have $\hat{u}(0) = \hat{v}(0) = \hat{w}(0) = 0$ in the optimization above, we can upper-bound expression D.5 by

$$\begin{aligned}
&\frac{4}{(p-1)p^2} \cdot \max_{\|z\|_1 \le \frac{p}{2}} \sum_{j=1}^{(p-1)/2} \left(\frac{z(j)}{3}\right)^{3/2} \\
&\le \frac{4}{3^{3/2}(p-1)p^2} \cdot \max_{\|z\|_1 \le \frac{p}{2}} \left(\sum_{j=1}^{(p-1)/2} z(j)^2\right)^{1/2} \cdot \left(\sum_{j=1}^{(p-1)/2} z(j)\right)^{1/2} \qquad \text{(Cauchy-Schwartz)} \\
&= \frac{2^{3/2}}{3^{3/2}(p-1)p^{3/2}} \cdot \max_{\|z\|_1 \le \frac{p}{2}} \|z\|_2 \\
&\le \frac{2^{3/2}}{3^{3/2}(p-1)p^{3/2}} \cdot \frac{p}{2} = \sqrt{\frac{2}{27}} \cdot \frac{1}{p^{1/2}(p-1)}.
\end{aligned}$$

The only way to turn inequality D.6 into an equality is to set $|\hat{u}(j)| = |\hat{v}(j)| = |\hat{w}(j)|$, and the only way to achieve $\|z\|_2 = \frac{p}{2}$ is to place all the mass on a single frequency, so the only possible way to achieve the upper bound is to set

$$|\hat{u}(j)| = |\hat{v}(j)| = |\hat{w}(j)| = \begin{cases} \sqrt{p/6} & \text{if } j = \pm\zeta \\ 0 & \text{otherwise} \end{cases}.$$

for some frequency $\zeta \in \{1, \dots, \frac{p-1}{2}\}$. In this case, we indeed match the upper bound:

$$\frac{4}{(p-1)p^2} \cdot \left(\frac{p}{6}\right)^{3/2} = \sqrt{\frac{2}{27}} \cdot \frac{1}{p^{1/2}(p-1)}.$$

so this is the maximum margin.

Putting it all together, and abusing notation by letting $\theta_u^* := \theta_u(\zeta)$, we obtain that all neurons

maximizing the expected class-weighted margin are of the form (up to scaling):

$$\begin{aligned} u(a) &= \frac{1}{p}\sum_{j=0}^{p-1} \hat{u}(j)\rho^{ja} \\ &= \frac{1}{p}\left[\hat{u}(\zeta)\rho^{\zeta a} + \hat{u}(-\zeta)\rho^{-\zeta a}\right] \\ &= \frac{1}{p}\left[\sqrt{\frac{p}{6}}\exp(i\theta_u^*)\rho^{\zeta a} + \sqrt{\frac{p}{6}}\exp(-i\theta_u^*)\rho^{-\zeta a}\right] \\ &= \sqrt{\frac{2}{3p}}\cos(\theta_u^* + 2\pi\zeta a/p) \end{aligned}$$

and

$$\begin{aligned} v(b) &= \sqrt{\frac{2}{3p}}\cos(\theta_v^* + 2\pi\zeta b/p) \\ w(c) &= \sqrt{\frac{2}{3p}}\cos(\theta_w^* + 2\pi\zeta c/p) \end{aligned}$$

for some phase offsets $\theta_u^*, \theta_v^*, \theta_w^* \in \mathbb{R}$ satisfying $\theta_u^* + \theta_v^* = \theta_w^*$ and some $\zeta \in \mathbb{Z}_p \setminus \{0\}$ (where $\zeta$ is the same for $u$, $v$, and $w$). □

It remains to construct a network $\theta^*$ which uses neurons of the above form and satisfies condition C.1 and Equation 7.1 with respect to $q = \text{unif}(\mathbb{Z}_p)$.

### D.6.2 Proof that condition C.1 and Equation 7.1 are satisfied

*Proof.* Our $\theta^*$ will consist of $4(p-1)$ neurons: 8 neurons for each of the frequencies $1, \ldots, \frac{p-1}{2}$. Consider a given frequency $\zeta$. For brevity, let $\cos_\zeta(x)$ denote $\cos(2\pi\zeta x/p)$, and similarly for $\sin_\zeta(x)$.

First, we observe:

$$\begin{aligned}\cos_\zeta(a+b-c) &= \cos_\zeta(a+b)\cos_\zeta(c) + \sin_\zeta(a+b)\sin_\zeta(c) \\ &= \cos_\zeta(a)\cos_\zeta(b)\cos_\zeta(c) - \sin_\zeta(a)\sin_\zeta(b)\cos_\zeta(c) \\ &\quad + \sin_\zeta(a)\cos_\zeta(b)\sin_\zeta(c) + \cos_\zeta(a)\sin_\zeta(b)\sin_\zeta(c)\end{aligned}$$

Each of these four terms can be implemented by a pair of neurons $\varphi_1, \varphi_2$. Consider the first term, $\cos_\zeta(a)\cos_\zeta(b)\cos_\zeta(c)$. For the first neuron $\varphi_1$, set $u_1(\cdot), v_1(\cdot), w_1(\cdot) := \cos_\zeta(\cdot)$, and for $\varphi_2$, set $u_2(\cdot) := \cos_\zeta(\cdot)$ and $v_2(\cdot), w_2(\cdot) := -\cos_\zeta(\cdot)$. These can be implemented in the form we derived by setting $(\theta_u^*, \theta_v^*, \theta_w^*)$ to $(0,0,0)$ for the first neuron and $(0,\pi,\pi)$ for the second.

Adding these two neurons, we obtain:

$$\begin{aligned}\varphi_1(a,b) + \varphi_2(a,b) &= (\cos_\zeta(a) + \cos_\zeta(a))^2\cos_\zeta(c) + (\cos_\zeta(a) - \cos_\zeta(a))^2(-\cos_\zeta(c)) \\ &= 4\cos_\zeta(a)\cos_\zeta(b)\cos_\zeta(c)\end{aligned}$$

Similarly, each of the other three terms can be implemented by pairs of neurons, by setting the phase offsets $(\theta_u^*, \theta_v^*, \theta_w^*)$ to

1. $(\frac{\pi}{2}, -\frac{\pi}{2}, 0)$ and $(\frac{\pi}{2}, \frac{\pi}{2}, \pi)$
2. $(-\frac{\pi}{2}, 0, -\frac{\pi}{2})$ and $(-\frac{\pi}{2}, \pi, \frac{\pi}{2})$
3. $(0, -\frac{\pi}{2}, -\frac{\pi}{2})$ and $(0, \frac{\pi}{2}, \frac{\pi}{2})$

If we include such a collection of 8 neurons for every frequency $\zeta \in \{1, \ldots, \frac{p-1}{2}\}$, the resulting

network will compute the function

$$\begin{aligned} f(a,b) &= \sum_{\zeta=1}^{(p-1)/2} \cos_\zeta(a+b-c) \\ &= \sum_{\zeta=1}^{p-1} \frac{1}{2} \cdot \exp(2\pi i\zeta(a+b-c)/p) \\ &= \begin{cases} \frac{p-1}{2} & \text{if } a+b=c \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

The scaling constant $\lambda$ for each neuron can be chosen so that the network has $L_{2,3}$-norm 1. For this network, every datapoint is on the margin, so $q = \text{unif}(\mathbb{Z}_p)$ is trivially supported on points on the margin, satisfying Equation 7.1. And for each input $(a,b)$, $f$ takes the same value on all incorrect labels $c'$, satisfying C.1. □

### D.6.3 Proof that all frequencies are used

*Proof.* For this proof, we need to introduce the multidimensional discrete Fourier transform. For a function $f : \mathbb{Z}_p^3 \to \mathbb{C}$, the multidimensional DFT of $f$ is defined as:

$$\hat{f}(j,k,\ell) := \sum_{a\in\mathbb{Z}_p} e^{-2\pi i\cdot ja/p} \left( \sum_{b\in\mathbb{Z}_p} e^{-2\pi i\cdot jb/p} \left( \sum_{c\in\mathbb{Z}_p} e^{-2\pi i\cdot jc/p} f(a,b,c) \right) \right)$$

for all $j,k,\ell \in \mathbb{Z}$.

To simplify the notation, let $\theta_u = \theta_u^* \cdot \frac{p}{2\pi}$, so

$$u(a) = \sqrt{\frac{2}{3p}} \cos_p(\theta_u + \zeta a).$$

Let

$$\begin{aligned} f(a,b,c) &= \sum_{h=1}^{H} \varphi_h(a,b,c) \\ &= \sum_{h=1}^{H} \left(u_h(a) + v_h(b)\right)^2 w_h(c) \\ &= \left(\frac{2}{3p}\right)^{3/2} \sum_{h=1}^{H} \left(\cos_p(\theta_{u_h} + \zeta_h a) + \cos_p(\theta_{v_h} + \zeta_h b)\right)^2 \cos_p(\theta_{w_h} + \zeta_h c) \end{aligned}$$

be the function computed by an arbitrary margin-maximizing network of width $H$, where each neuron is of the form derived earlier.

Each neuron $\varphi$ can be split into three terms:

$$\varphi(a,b,c) = \varphi^{(1)}(a,b,c) + \varphi^{(2)}(a,b,c) + \varphi^{(3)}(a,b,c) := u(a)^2 w(c) + v(b)^2 w(c) + 2u(a)v(b)w(c)$$

$\widehat{\varphi^{(1)}}(j,k,\ell)$ is nonzero only for $k = 0$, and $\widehat{\varphi^{(2)}}(j,k,\ell)$ is nonzero only for $j = 0$. For the third term, we have

$$\widehat{\varphi^{(3)}}(j,k,\ell) = 2 \sum_{a,b,c \in \mathbb{Z}_p} u(a)v(b)w(c)\rho^{-(ja+kb+\ell c)} = 2\hat{u}(j)\hat{v}(k)\hat{w}(\ell).$$

In particular,

$$\begin{aligned}
\hat{u}(j) &= \sum_{a\in\mathbb{Z}_p} \sqrt{\frac{2}{3p}} \cos_p(\theta_u + \zeta a)\rho^{-ja} \\
&= (6p)^{-1/2} \sum_{a\in\mathbb{Z}_p} \left(\rho^{\theta_u+\zeta a} + \rho^{-(\theta_u+\zeta a)}\right)\rho^{-ja} \\
&= (6p)^{-1/2} \left(\rho^{\theta_u} \sum_{a\in\mathbb{Z}_p} \rho^{(\zeta-j)a} + \rho^{-\theta_u} \sum_{a\in\mathbb{Z}_p} \rho^{-(\zeta+j)a}\right) \\
&= \begin{cases} \sqrt{p/6}\cdot\rho^{\theta_u} & \text{if } j = \zeta \\ \sqrt{p/6}\cdot\rho^{-\theta_u} & \text{if } j = -\zeta \\ 0 & \text{otherwise} \end{cases}
\end{aligned}$$

and similarly for $\hat{v}$ and $\hat{w}$. $\zeta$ was defined to be nonzero, so the $\zeta = 0$ case is ignored. Thus, $\hat{\varphi^{(3)}}(j,k,\ell)$ is nonzero only when $j,k,\ell$ are all $\pm\zeta$. We can conclude that $\hat{\varphi}(j,k,\ell)$ can only be nonzero if one of the following conditions holds:

1. $j = 0$
2. $k = 0$
3. $j,k,\ell = \pm\zeta$.

Independent of the above considerations, we know by Lemma 7.6 that the function $f$ implemented by the network has equal margin across different inputs and across different classes for the same input. In other words, $f$ can be decomposed as

$$f(a,b,c) = f_1(a,b,c) + f_2(a,b,c)$$

where

$$f_1(a,b,c) = F(a,b)$$

for some $F : \mathbb{Z}_p \times \mathbb{Z}_p \to \mathbb{R}$, and

$$f_2(a,b,c) = \lambda \cdot \mathbf{1}_{a+b=c}$$

where $\lambda > 0$ is the margin of $f$.

The Fourier transforms of $f_1$ and $f_2$ are

$$\hat{f}_1(j,k,l) = \begin{cases} \hat{F}(j,k) & \text{if } \ell = 0 \\ 0 & \text{otherwise} \end{cases}$$

and

$$\hat{f}_2(j,k,l) = \begin{cases} \lambda p^2 & \text{if } j = k = -\ell \\ 0 & \text{otherwise} \end{cases}.$$

Hence, when $j = k = -\ell \neq 0$, we must have $\hat{f}(j,k,\ell) > 0$. But then, from the conditions under which each neuron's DFT $\hat{\varphi}$ is nonzero, it must follow that there is at least one neuron for each frequency. □

## D.7 Proofs for Sparse parity

*Theorem.* Consider a single hidden layer neural network of width $m$ with the activation function given by $x^k$, i.e, $f(x) = \sum_{i=1}^{m} (u_i^\top x)^k w_i$, where $u_i \in \mathbb{R}^n$ and $w_i \in \mathbb{R}^2$, trained on the $(n,k)-$sparse parity task. Without loss of generality, assume that the first coordinate of $w_i$ corresponds to the output for class $y = +1$. Denote the vector $[1,-1]$ by $\boldsymbol{b}$. Provided $m \geq 2^{k-1}$, the $L_{2,k+1}$ maximum

margin is:

$$k!\sqrt{2(k+1)^{-(k+1)}}.$$

Any network achieving this margin satisfies the following conditions:

1. For every $i$ having $\|u_i\| > 0$, $\text{spt}(u_i) = S$, $w_i$ lies in the span of $\boldsymbol{b}$ and $\forall j \in S, |u_i[j]| = \|w_i\|$.
2. For every $i$, $\left(\Pi_{j\in S} u_i[j]\right)\left(w_i^\top \boldsymbol{b}\right) \geq 0$.

*Proof.* We will consider $q^*$ to be equally distributed on the dataset and optimize the class-weighted margin as defined in Equation 7.3. We will consider the weight $\tau(x,y)[y'] = 1$ for $y' \neq y$. Also, let $\boldsymbol{a}$ denote the vector $[1,1]$ and $\boldsymbol{b}$ denote the vector $[1,-1]$. Then, any $w_i \in \mathbb{R}^2$ can be written as $w_i = \frac{1}{\sqrt{2}}\left[\alpha_i \boldsymbol{a} + \beta_i \boldsymbol{b}\right]$ for some $\alpha_i, \beta_i \in \mathbb{R}$.

First, using lemma 7.5, we can say that one neuron maximizers of class-weighted margin are given by

$$\underset{\|[u,w]\|_2 \leq 1}{\arg\max} \underset{(x,y)\sim D}{\mathbb{E}} \left[\varphi(\{u,w\},x)[y] - \varphi(\{u,w\},x)[y']\right]$$

where $y' = -y$, $\varphi(\{u,w\},x) = (u^\top x)^k w$ and $\|[u,w]\|_2$ represents the 2-norm of the concatenation of $u$ and $w$.

Considering that $y \in \{\pm 1\}$ and $w = \frac{1}{\sqrt{2}}\left[\alpha \boldsymbol{a} + \beta \boldsymbol{b}\right]$, we can say $\varphi(\{u,w\},x)[y] = \frac{1}{\sqrt{2}}(u^\top x)^k[\alpha + y\beta]$. Thus, we can say

$$\begin{aligned}
\underset{(x,y)\sim D}{\mathbb{E}} \left[\varphi(\{u,w\},x)[y] - \varphi(\{u,w\},x)[y']\right] &= \sqrt{2} \underset{(x,y)\sim D}{\mathbb{E}} \left[(u^\top x)^k \beta y\right] \\
&= \sqrt{2} \underset{(x,y)\sim D}{\mathbb{E}} \left[(u^\top x)^k \beta \Pi_{i\in S} x_i\right] \\
&= \sqrt{2}k! \left(\Pi_{i\in S} u_i\right)\beta
\end{aligned}$$

where in the last step, all other terms are zero by symmetry of the dataset.

Clearly, under the constraint $\|u\|^2 + \alpha^2 + \beta^2 \leq 1$ (where $\|w\|^2 = \alpha^2 + \beta^2$), this is maximized when $u_i = 0$ for $i \notin S, \alpha = 0, u_i = \pm\frac{1}{\sqrt{k+1}}$ and $\beta = \pm\frac{1}{\sqrt{k+1}}$, with $(\Pi_{i\in S} u_i)\,\beta > 0$.

Now, using Lemma 7.5, we will create a network using these optimal neurons such that it satisfies **C.1**, and Equations 7.1 and 7.3, thus concluding by Lemma 7.6. **C.1** holds trivially as this is a binary classification task, so $g' = g$.

Consider a maximal subset $A \subset \{\pm 1\}^k$ such that if $\in A$, then $- \notin A$ and for any $\in A$, $_1 = 1$. Now, consider a neural network having $2^{k-1}$ neurons given by

$$f(\theta, x) = \frac{1}{2^{k-1}} \sum_{\in A} \left( \sum_{i=1}^{k} \frac{\sigma_i}{\sqrt{k+1}} x_{S_i} \right)^k \frac{\left(\Pi_{i=1}^{k} \sigma_i\right)}{\sqrt{k+1}} \frac{1}{\sqrt{2}} \boldsymbol{b} = \frac{1}{\sqrt{2}} k! (k+1)^{-(k+1)/2} \left(\Pi_{i\in S} x_i\right) \boldsymbol{b}$$

By Lemma 7.5, the above neural network also maximizes the class-weighted mean margin. Moreover, it also satisfies Equation 7.1, as every term other than $\Pi x_{S_i}$ cancels out in the sum.

Consider any monomial $T$ which depends only on $S' \subset S$. Consider any one of the terms in $f(x)$ and let the coefficient of $T$ in the term given by $c_T$. Consider another term in $f(x)$, where, for some $i \in S \setminus S'$ and $j = k+1$, $\sigma_i$ and $\sigma_j$ are flipped. For this term, the coefficient of $T$ will be $-c_T$, as for all $i \in S'$, $\sigma_i$ is the same, but $\sigma_{k+1}$ is different. Thus, for any such monomial, its coefficient in expanded $f(x)$ will be 0 as terms will always exist in these pairs.

Thus, $f(\theta, x)$ satisfies **C.1**, Equation 7.1 and 7.3, hence, by Lemma 7.6, any maximum margin solution satisfies the properties stated in Theorem 7.8. □

### D.8 Additional Group Representation Theory Preliminaries

In this section we properly define relevant results from group representation theory used in the proof of Theorem 7.9. We also refer the reader to Kosmann-Schwarzbach et al. (2010), one of many good references for representation theory.

**Definition D.1.** A linear representation of a group $G$ is a finite dimensional complex vector space $V$ and a group homomorphism $R : G \to GL(V)$. We denote such a representation by $(R, V)$ or simply just $R$. The dimension of the representation $R$, denoted $d_R$, equals the dimension of the vector space $V$.

In our case we are only concerned with finite groups with real representations, i.e. $V = \mathbb{R}^d$ and each representation $R$ maps group elements to real invertible $d \times d$ matrices. Furthermore, we are only concerned with *unitary* representations $R$, i.e. $R(g)$ is unitary for every $g$. It is a known fact that every representation of a finite group can be made unitary, in the following sense:

**Theorem D.2** (Kosmann-Schwarzbach et al. (2010), Theorem 1.5.)**.** *Every representation of a finite group $(R, V)$ is unitarizable, i.e. there is a scalar product on $V$ such that $R$ is unitary.*

Also of particular interest are irreducible representations.

**Definition D.3.** A representation $(R, V)$ of $G$ is irreducible if $V \neq \{0\}$ and the only vector subspaces of $V$ invariant under $R$ are $\{0\}$ or $V$ itself.

A well-known result is Maschke's Theorem, which states that every finite-dimensional representation of a finite group is completely reducible; thus it suffices to consider a fundamental set of irreducible unitary representations in our analysis.

**Theorem D.4** (Maschke's Theorem.)**.** *Every finite-dimensional representation of a finite group is a direct sum of irreducible representations.*

**Theorem D.5** (Kosmann-Schwarzbach et al. (2010), Theorem 3.4.)**.** *Let $G$ be a finite group. If $R_1, \ldots, R_K$ denote the irreducible representations of $G$, then $|G| = \sum_{n=1}^{K} d_{R_n}^2$, where $d_{R_n}$ represents the dimensionality of $R_n$.*

The theory about characters of representations and orthogonality relations are essential for our max margin analysis. This is a rich area of results, and we only list those that are directly used in our proofs.

**Definition D.6.** Let $(R, V)$ be a representation of $G$. the character of $R$ is the function $\chi_R : G \to \mathbb{R}$ defined as $\chi_R(g) = \mathrm{tr}(R(g))$ for each $g \in G$.

For each conjugacy class of $G$, the character of $R$ is constant (this can easily be verified via properties of the matrix trace). More generally, functions which are constant for each conjugacy class are called *class functions* on $G$. Given the characters across inequivalent irreducible representations, one can construct a "*character table*" for a group $G$ in which the columns correspond to the conjugacy classes of a group, and whose rows correspond to inequivalent irreducible representations of a group. The entries of the character table correspond to the character for the representation at that given row, evaluated on the conjugacy class at that given column.

Characters of inequivalent irreducible representations are in fact orthogonal, which follow from the orthogonality relations of representation matrix elements. For a unitary irreducible representation $R$, define the vector $R_{(i,j)} = (R(g)_{(i,j)})_{g \in G}$ with entries being the $(i,j)$th entry of the matrix output for each $g \in G$ under $R$. We have the following result.

**Proposition D.7** (Kosmann-Schwarzbach et al. (2010), Corollary 2.10.)**.** *Let $(R_1, V_1)$ and $(R_2, V_2)$ be unitary irreducible representations of $G$. Choosing two orthonormal bases in $V_1$ and $V_2$, the following holds:*

1. *If $R_1$ and $R_2$ are inequivalent, then for every $i, j, k, l$, we have $\langle R_{1(i,j)}, R_{2(k,l)} \rangle = 0$.*
2. *If $R_1 = R_2 = R$ and $V_1 = V_2 = V$, then for every $i, j, k, l$, we have $\langle R_{(i,j)}, R_{(k,l)} \rangle = \frac{1}{d_R} \delta_{ik} \delta_{jl}$, where $\delta_{ik} = 1[i = k]$.*

**Theorem D.8** (Kosmann-Schwarzbach et al. (2010), Theorem 2.11.). *Let $G$ be a finite group. If $R_1$ and $R_2$ are inequivalent irreducible representations of $G$, then $\langle \chi_{R_1}, \chi_{R_2} \rangle = 0$. If $R$ is an irreducible representation of $G$, then $\langle \chi_R, \chi_R \rangle = 1$.*

A fundamental result about characters is that the irreducible characters of $G$ form an orthonormal set in $L^2(G)$ (Kosmann-Schwarzbach et al. (2010), Theorem 2.12.). This implies the following result, which states that the irreducible characters form an orthonormal basis in the vector space of class functions on $G$ taking values in $\mathbb{R}$. Since this vector space has dimension equal to the number of conjugacy classes of $G$, it also follows that the number of equivalence classes of irreducible representations is the number of conjugacy classes. In other words, the character table is square for every finite group.

**Theorem D.9** (Kosmann-Schwarzbach et al. (2010), Theorem 3.6.). *The irreducible characters form an orthonormal basis of the vector space of character functions.*

In section D.9 of the Appendix, we rigorously define the basis vectors for network weights based on the representation matrix elements defined in Proposition D.7, and establish the properties they satisfy, which are key to our analysis.

### D.8.1 A Concrete Example: Symmetric Group

The symmetric group $S_n$ consists of the permutations over a set of cardinality $n$. The order of the group is $n!$. It is a fact that every permutation can be written as a product of *transpositions*— a permutation which swaps two elements. We can associate with each permutation the parity of the number of transpositions needed, which is independent of the choice of decomposition.

We will provide a concrete description of the representation theory for $S_5$, which is a central

| class | $e$ | $(1\,2)$ | $(1\,2)(3\,4)$ | $(1\,2\,3)$ | $(1\,2\,3\,4)$ | $(1\,2\,3\,4\,5)$ | $(1\,2)(3\,4\,5)$ |
|---|---|---|---|---|---|---|---|
| size | 1 | 10 | 15 | 20 | 30 | 24 | 20 |
| $R_1$ | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| $R_2$(sign) | 1 | $-1$ | 1 | 1 | $-1$ | 1 | $-1$ |
| $R_3$(standard) | 4 | $-2$ | 0 | 1 | 0 | $-1$ | 1 |
| $R_4$(standard $\otimes$ sign) | 4 | 2 | 0 | 1 | 0 | $-1$ | $-1$ |
| $R_5$(5d_a) | 5 | 1 | 1 | $-1$ | $-1$ | 0 | 1 |
| $R_6$(5d_b) | 5 | $-1$ | 1 | $-1$ | 1 | 0 | $-1$ |
| $R_7$(6d) | 6 | 0 | $-2$ | 0 | 0 | 1 | 0 |

**Table D.1:** Character table of $S_5$.

group of study in this paper. It has 7 conjugacy classes, which we denote as

$$\{e, (1\,2), (1\,2)(3\,4), (1\,2\,3), (1\,2\,3\,4), (1\,2\,3\,4\,5), (1\,2)(3\,4\,5)\}$$

(selecting one representative from each conjugacy class). It also has 7 irreducible representations. Apart from the trivial representation, it has another 1-dimensional *sign* representation representing the parity of a permutation.

The symmetric group also has an $n$-dimensional representation which is the *natural permutation representation*, mapping permutations to *permutation matrices* which shuffle the $n$ coordinates. It turns out that this is in fact reducible, since this has the trivial subrepresentation consisting of vectors whose coordinates are all equal. Decomposing this representation into irreducible representations results in the trivial representation and what is called the *standard* representation of dimension $n-1$. It has another $n-1$-dimensional representation, which is the *product of sign and standard* representations.

The final three representations of $S_5$ are higher-dimensional, with dimensions 5, 5, and 6. We denote them as 5d_a, 5d_b, and 6d. We give the character table of $S_5$ in Table D.1, which will be useful for calculating the value of the max margin which we theoretically derive.

## D.9 Proofs for finite groups with real representations

In this section we prove that for finite groups with real representations, all max margin solutions have neurons which only use a single irreducible representation.

*Theorem.* Consider a single hidden layer neural network of width $m$ with quadratic activation trained on learning group composition for $G$ with real irreducible representations. Provided $m \geq 2\sum_{n=2}^{K} {d_{R_n}}^3$ and $\sum_{n=2}^{K} {d_{R_n}}^{1.5}\chi_{R_n}(C) < 0$ for every non-trivial conjugacy class $C$, the $L_{2,3}$ maximum margin is:

$$\gamma^* = \frac{2}{3\sqrt{3|G|}} \frac{1}{\left(\sum_{n=2}^{K} d_{R_n}^{2.5}\right)}.$$

Any network achieving this margin satisfies the following conditions:

1. For every neuron, there exists a non-trivial representation such that the input and output weight vectors are spanned only by that representation.

2. There exists at least one neuron spanned by each representation (except for the trivial representation) in the network.

Let $R_1, \ldots, R_K$ be the unitary irreducible representations and let $C_1, \ldots, C_K$ be the conjugacy classes of a finite group $G$ with real representations. We fix $R_1$ to be the trivial one-dimensional representation mapping $R_1(g) = 1$ for all $g \in G$ and $C_1$ to be the trivial conjugacy class $C_1 = \{e\}$. For each of these representations $(V, R)$ of the group $G$, where $R : G \to V$, we will consider the $|G|$-dimensional vectors by fixing one position in the matrix $R(g)$ for all $g \in G$, i.e. vectors $(R(g)_{i,j})_{g \in G}$ for some $i, j \in [d_V]$. These form a set of $|G|$ vectors which we will denote $\rho_1, \ldots, \rho_{|G|}$ ($\rho_1$ is always the vector corresponding to the trivial representation). These vectors in fact form an orthogonal basis, and satisfy additional properties established in the following lemma.

**Lemma D.10.** *The set of vectors $\rho_1, \ldots, \rho_{|G|}$ satisfy the following properties:*

1. $\sum_{a \in G} \rho_i(a)\rho_j(a) = 0$ *for* $i \neq j$. *(Orthogonality)*

2. $\sum_{a \in G} \rho_i(a)^2 = |G|/d_V$ *for all* $i$, *where* $d_V$ *is the dimensionality of the vector space* $V$ *corresponding to the representation that* $\rho_i$ *belongs to.*

3. *For all the* $\rho_j$ *which correspond to off-diagonal entries of a representation,* $\sum_{a \in C_i} \rho_j[a] = 0$, *i.e, the sum of elements within the same conjugacy class is 0.*

4. *If* $\rho_j$ *and* $\rho_k$ *correspond to different diagonal entries within the same representation, then* $\sum_{a \in C_i} \rho_j[a] = \sum_{a \in C_i} \rho_k[a]$, *i.e, for the diagonal entries, the sum for a given conjugacy class is invariant with the position of the diagonal element.*

*Proof.* The first two properties are the orthogonality relations of unitary representation matrix elements (Proposition D.7), and the last two points follow additionally from Proposition 2.7 and Proposition 2.8 in Kosmann-Schwarzbach et al. (2010). □

Since this set of $|G|$-dimensional vectors are orthogonal to each other, each set of weights for a neuron in our architecture can be expressed as a linear combination of these basis vectors

$$u = \sum_{i \in [|G|]} \alpha_i \rho_i, \quad v = \sum_{i \in [|G|]} \beta_i \rho_i, \quad w = \sum_{i \in [|G|]} \gamma_i \rho_i.$$

It will also be useful to define the matrices $\alpha_{R_i}, \beta_{R_i}, \gamma_{R_i}$ for each irreducible representation $R_i$ of $G$ which consist of the coefficients for $u$, $v$, and $w$ corresponding to each entry in the representation matrix.

Let $h_{u,v,w}(c) := \mathbb{E}_{a,b}\left[(u(a) + v(b))^2 w(a \circ b \circ c)\right]$. Recall we seek solutions for the following *weighted* margin maximization problem

$$h_{u,v,w}(e) - \sum_{c \neq e} \tau_c h_{u,v,w}(c), \text{ where } \sum_{c \neq e} \tau_c = 1. \tag{D.7}$$

Note that if we substitute the weights $u, v, w$ in terms of the basis vectors in the definition of $h_{u,v,w}$

$$h_{u,v,w}(c) = \mathop{\mathbb{E}}_{a,b}\left[\left(\sum \alpha_i \rho_i(a) + \sum \beta_i \rho_i(b)\right)^2 \left(\sum \gamma_i \rho_i(a \circ b \circ c)\right)\right]$$

and we expand this summation, all terms involving the trivial representation vector $\rho_1$ will equal zero since it is constant on all group elements. Furthermore, for terms of the form

$$\mathop{\mathbb{E}}_{a,b}[\rho_i(a)^2 \rho_k(a \circ b \circ c)] = \mathop{\mathbb{E}}_{a}[\rho_i(a)^2 \mathop{\mathbb{E}}_{b}[\rho_k(a \circ b \circ c)]] = 0$$

due to $\mathbb{E}_b[\rho_k(a \circ b \circ c)] = 0$ by orthogonality to the trivial representation vector.

Thus as was the case for the cyclic group, we study the term $\tilde{h}_{u,v,w}(c) := \mathbb{E}_{a,b}\left[u(a)v(b)w(a \circ b \circ c)\right]$ and derive an expression for the weighted margin in the following lemma.

**Lemma D.11.** *Suppose the weights $\tau_c$ in the expression for the weighted margin in D.7 were constant over conjugacy classes, i.e. we have $\tau_c = \tau_{C_i}$ for all $c \in C_i$ and $i \in [K]$. Then the weighted margin can be simplified as*

$$\sum_{m=2}^{K}\left(1 - \sum_{n=2}^{K} \frac{\tau_{C_n}|C_n|\chi_{R_m}(C_n)}{d_{R_m}}\right)\frac{\mathrm{tr}(\alpha_{R_m}\beta_{R_m}\gamma_{R_m}{}^T)}{{d_{R_m}}^2}.$$

*Proof.* Consider one term $\mathbb{E}_{a,b}[\alpha_i\beta_j\gamma_k\rho_i(a)\rho_j(b)\rho_k(a \circ b \circ c)]$ in the expansion of the product in $\tilde{h}_{u,v,w}(c)$. Note that $\rho_k(a \circ b \circ c)$ is one *entry* in the matrix of some irreducible representation evaluated at $a \circ b \circ c$; this can be expanded in terms of the same irreducible representation *matrix* evaluated at $a$, $b$, and $c$ using matrix multiplication. This results in terms of the form

$$\rho_i(a)\rho_j(b)\rho_{i'}(a)\rho_{j'}(b)\rho_{k'}(c)$$

in the expectation, where $\rho_{i'}$ and $\rho_{j'}$ correspond to entries of matrices from the same representation as $\rho_{k'}$. Thus if either $\rho_i$ or $\rho_j$ correspond to vectors from a *different* representation than $\rho_k$, the

expectation of this term will be zero, by orthogonality of the basis vectors.

Hence we can assume that $\rho_i, \rho_j, \rho_k$ correspond to entries from the same representation $(V, R)$. Let $d = d_V$. Let us write $i = (i_1, i_2), j = (j_1, j_2), k = (k_1, k_2)$, the matrix indices for this representation. We can expand the term $\rho_k(a \circ b \circ c)$ as described above.

$$\begin{aligned}\rho_i(a)\rho_j(b)\rho_k(a \circ b \circ c) &= \rho_i(a)\rho_j(b) \sum_{m=1}^{d} \rho_{(k_1,m)}(a \circ b)\rho_{(m,k_2)}(c) \\ &= \sum_{\ell=1}^{d}\sum_{m=1}^{d} \rho_{(i_1,i_2)}(a)\rho_{(k_1,\ell)}(a)\rho_{(j_1,j_2)}(b)\rho_{(\ell,m)}(b)\rho_{(m,k_2)}(c).\end{aligned}$$

From this it is clear that when taking the expectation over choosing $a, b$ uniformly, the only non-zero terms are when $(i_1, i_2) = (k_1, \ell)$ and $(j_1, j_2) = (\ell, m)$, once again by orthogonality of the basis vectors. Thus we have

$$\alpha_i\beta_j\gamma_k\rho_i(a)\rho_j(b)\rho_k(a \circ b \circ c) = \alpha_{(i_1,j_1)}\beta_{(j_1,j_2)}\gamma_{(i_1,k_2)}\rho^2_{(i_1,j_1)}(a)\rho^2_{(j_1,j_2)}(b)\rho_{(j_2,k_2)}(c).$$

Moreover, we know $\mathbb{E}[\rho_i(a)^2] = 1/d$, where $d$ is the dimensionality of the representation. Now, for a particular $c$, we will evaluate group all terms containing $\rho_{(j_2,k_2)}(c)$ and take the expectation over $a, b$, which yields

$$\frac{1}{d^2}\sum_{i_1=1}^{d}\sum_{j_1=1}^{d} \alpha_{(i_1,j_1)}\beta_{(j_1,j_2)}\gamma_{(i_1,k_2)}\rho_{(j_2,k_2)}(c). \tag{D.8}$$

From the third property of Lemma D.10, for every conjugacy class $C_n$ for $n \in [K]$, we have

$$\sum_{c \in C_n} \frac{1}{d^2}\sum_{i_1=1}^{d}\sum_{j_1=1}^{d} \alpha_{(i_1,j_1)}\beta_{(j_1,j_2)}\gamma_{(i_1,k_2)}\rho_{(j_2,k_2)}(c) = 0$$

for $j_2 \neq k_2$. Thus, we can focus on diagonal entries $\rho_{(k,k)}(c)$ (i.e. where $j_2 = k_2$ in the expression

D.8 above). In this case, following directly from D.8 grouping all terms containing $\rho_{(k,k)}(c)$ we get

$$\frac{1}{d^2}\sum_{i_1=1}^{d}\sum_{j_1=1}^{d}\alpha_{(i_1,j_1)}\beta_{(j_1,k)}\gamma_{(i_1,k)}\rho_{(k,k)}(c). \tag{D.9}$$

Note that this coefficient in front of $\rho_{(k,k)}(c)$ is the sum of the entries of the $k^{th}$ column of the matrix $(\alpha_R\beta_R)\odot\gamma_R$ divided by $d^2$ (with $\alpha_R\beta_R$ interpreted as matrix product and $\odot$ being the Hadamard product). Recall that $i_1, j_1$, and $k$ are indices from the *same* representation $R$. By summing over all diagonal entries $(k,k)$ in $R$, we evaluate the expression

$$\begin{aligned}\sum_{i_1,j_1,k\in[d]}\frac{\alpha_{(i_1,j_1)}\beta_{(j_1,k)}\gamma_{(i_1,k)}}{d^2}\left[\rho_{k,k}(e)-\sum_{c\neq e}\tau_c\rho_{(k,k)}(c)\right]\\ =\frac{\mathrm{tr}(\alpha_R\beta_R{\gamma_R}^T)}{d^2}\left[1-\sum_{n=2}^{K}\tau_{C_n}\sum_{c\in C_n}\rho_{(k,k)}(c)\right]\end{aligned}$$

where we have replaced $\tau_c$ with the same weight $\tau_{C_n}$ for each non-trivial conjugacy class $C_2,\ldots,C_K$ and the term $\sum_{c\in C_n}\rho_{(k,k)}(c)$ is independent of the choice of $k$ (by property 4 of Lemma D.10). Thus after summing over all $k\in[d]$ the coefficient in equation D.9 is the sum of all entries of the matrix $(\alpha_R\beta_R)\odot\gamma_R$ (which equals $\mathrm{tr}(\alpha_R\beta_R{\gamma_R}^T)$). Furthermore,

$$|C_n|\chi_R(C_n)=\sum_{c\in C_n}\sum_{k\in[d]}\rho_{(k,k)}(c)=\sum_{k\in[d]}\sum_{c\in C_n}\rho_{(k,k)}(c)=d\sum_{c\in C_n}\rho_{(k,k)}(c)$$

where the first equality follows from the definition of the character of the representation $R$ which is constant over elements in the same conjugacy class, and the last equality follows again from property 4 of Lemma D.10). Thus $\sum_{c\in C_n}\rho_{(k,k)}(c)=|C_n|\chi_R(C_n)/d$ for all $k$.

Now we can evaluate our result for the weighted margin. The expression in D.10 is the contribution of one representation $R$ to the total weighted margin. Thus by summing over all non-trivial

representations of $G$, we get the final result.

$$\tilde{h}_{u,v,w}(e) - \sum_{c \neq e} w_c \tilde{h}_{u,v,w}(c) = \tilde{h}_{u,v,w}(e) - \sum_{n=2}^{K} \tau_{C_n} \sum_{c \in C_n} \tilde{h}_{u,v,w}(c) \tag{D.10}$$

$$= \sum_{m=2}^{K} \sum_{i_1,j_1,k \in [d_{R_m}]} \frac{\alpha_{(i_1,j_1)} \beta_{(j_1,k)} \gamma_{(i_1,k)}}{d^2} \left[ \rho_{k,k}(e) - \sum_{c \neq e} \tau_c \rho_{(k,k)}(c) \right] \tag{D.11}$$

$$= \sum_{m=2}^{K} \frac{\mathrm{tr}(\alpha_{R_m} \beta_{R_m} \gamma_{R_m}{}^T)}{d^2} \left[ 1 - \sum_{n=2}^{K} \tau_{C_n} \sum_{c \in C_n} \rho_{(k,k)}(c) \right] \tag{D.12}$$

$$= \sum_{m=2}^{K} \left[ 1 - \sum_{n=2}^{K} \frac{\tau_{C_n} |C_n| \chi_{R_m}(C_n)}{d_{R_m}} \right] \frac{\mathrm{tr}(\alpha_{R_m} \beta_{R_m} \gamma_{R_m}{}^T)}{{d_{R_m}}^2}. \tag{D.13}$$

□

We have simplified the weighted margin expression for any set of weights on the conjugacy classes. Recall that we wish to optimize this weighted margin across individual neurons and then scale them appropriately to define the network $\theta^*$ satisfying **C.1** and Equation 7.1 to find the max margin solution.

The next lemma establishes the original $L_2$ norm restraint over neurons on the weighted margin problem in terms of the coefficients with respect to each representation.

**Lemma D.12.** *The $L_2$ norm of $u$, $v$ and $w$ are related to the Frobenius norm of $\alpha$, $\beta$ and $\gamma$ as follows:*

$$\|u\|^2 + \|v\|^2 + \|w\|^2 = \sum_{m=1}^{K} \frac{|G|}{d_{R_m}} \left( \|\alpha_{R_m}\|_F^2 + \|\beta_{R_m}\|_F^2 + \|\gamma_{R_m}\|_F^2 \right)$$

*Proof.* The proof follows from 1st and 2nd point of Lemma D.10. □

By the above two lemmas, we want to maximize the weighted margin with respect to the norm constraint

$$\sum_{m=1}^{K} \frac{|G|}{d_{R_m}} \left( \|\alpha_{R_m}\|_F^2 + \|\beta_{R_m}\|_F^2 + \|\gamma_{R_m}\|_F^2 \right) \leq 1. \tag{D.14}$$

Under this constraint, the following lemma provides the maximum value for the weighted margin, which occurs only when the weights $u, v, w$ are spanned by a single representation $R$.

**Lemma D.13.** *Consider the set of representations $\mathcal{R}$ given by*

$$\mathcal{R} := \underset{m=2,..,K}{\arg\max} \frac{1}{\sqrt{d_{R_m}}} \left[1 - \sum_{n=2}^{K} \frac{\tau_{C_n} |C_n| \chi_{R_m}(C_n)}{d_{R_m}}\right]$$

*The weighted margin in Lemma D.11 is maximized under the norm constraint in (D.14) only when the weights $u, v, w$ are spanned by a single representation belonging to the set $\mathcal{R}$; that is, $\alpha_R, \beta_R, \gamma_R \not\equiv 0$ for only one non-trivial representation $R \in \mathcal{R}$, and are 0 otherwise. In this case, the maximum value attained is*

$$\frac{1}{3\sqrt{3}|G|^{3/2}} \max_{m=2,..,K} \frac{1}{\sqrt{d_{R_m}}} \left[1 - \sum_{n=2}^{K} \frac{\tau_{C_n} |C_n| \chi_{R_m}(C_n)}{d_{R_m}}\right].$$

*Proof.* First we consider the case where $u, v, w$ are spanned by only one representation. Then it suffices to evaluate

$$\max_{\alpha_R, \beta_R, \gamma_R} \frac{\mathrm{tr}(\alpha_R \beta_R {\gamma_R}^T)}{d^2} \text{ s.t. } \left(\|\alpha_R\|_F^2 + \|\beta_R\|_F^2 + \|\gamma_R\|_F^2\right) \leq \frac{d}{|G|}.$$

Here let's denote the columns of $\alpha_R$ (resp. $\beta_R$) as $\vec{\alpha_j} = (\alpha_{j,1}, \ldots, \alpha_{j,d})$ for $1 \leq j \leq d$ (resp. $\vec{\beta_j}$). Thus $\mathrm{tr}(\alpha_R \beta_R {\gamma_R}^T) = \sum_{j,k} (\vec{\alpha_j} \cdot \vec{\beta_k}) \gamma_{(j,k)}$. This can be viewed as the dot product of the linearizations of $\alpha_R \beta_R$ and $\gamma_R$, and thus by Cauchy-Schwarz it follows that

$$\sum_{j,k} (\vec{\alpha_j} \cdot \vec{\beta_k}) \gamma_{(j,k)} \leq \sqrt{\sum_{j,k} (\vec{\alpha_j} \cdot \vec{\beta_k})^2} \sqrt{\|\gamma_R\|_F^2}$$

with equality when $\gamma_{(j,k)}$ is proportional to $\vec{\alpha_j} \cdot \vec{\beta_k}$. We can apply Cauchy-Schwarz once again to

the first term on the right hand side above and obtain

$$\sqrt{\sum_{j,k}(\vec{\alpha_j}\cdot\vec{\beta_k})^2} \leq \sqrt{\sum_{j,k}\|\vec{\alpha_j}\|_2^2\|\vec{\beta_k}\|_2^2} = \sqrt{\|\alpha_R\|_F^2\|\beta_R\|_F^2}$$

once again with equality when all $\vec{\alpha_j}, \vec{\beta_k}$ are proportional to each other. Combining these together, we want to maximize $\sqrt{\|\alpha_R\|_F^2\|\beta_R\|_F^2\|\gamma_R\|_F^2}$ subject to $\left(\|\alpha_R\|_F^2 + \|\beta_R\|_F^2 + \|\gamma_R\|_F^2\right) \leq \frac{d}{|G|}$. By the AM-GM inequality, we have

$$\sqrt{\|\alpha_R\|_F^2\|\beta_R\|_F^2\|\gamma_R\|_F^2} \leq \left(\frac{\|\alpha_R\|_F^2 + \|\beta_R\|_F^2 + \|\gamma_R\|_F^2}{3}\right)^{3/2}$$

with equality when $\|\alpha_R\|_F^2 = \|\beta_R\|_F^2 = \|\gamma_R\|_F^2 = \frac{d}{3|G|}$. Thus the maximum value attained is $\frac{1}{(|G|^{3/2}3\sqrt{3d})}\left[1 - \sum_{n=2}^{K}\frac{\tau_{C_n}|C_n|\chi_R(C_n)}{d_R}\right]$.

Now consider the general case when $u, v, w$ were spanned by the representations $R_2, \ldots, R_K$ (as $R_1$ does not appear in Equation D.13). The norm constraint now becomes

$$\sum_{m=2}^{K}\frac{|G|}{d_{R_m}}\left(\|\alpha_{R_m}\|_F^2 + \|\beta_{R_m}\|_F^2 + \|\gamma_{R_m}\|_F^2\right) \leq 1.$$

This can be equivalently written as

$$\|\alpha_{R_m}\|_F^2 + \|\beta_{R_m}\|_F^2 + \|\gamma_{R_m}\|_F^2 \leq \frac{d_{R_m}\varepsilon_m}{|G|} \quad \forall m \in \{2, \ldots, K\}$$

$$\varepsilon_m \geq 0 \quad \forall m \in \{2, \ldots, K\}$$

$$\sum_{m=2}^{K}\varepsilon_m \leq 1$$

Repeating the calculation above, we get that for a given $\varepsilon_2, \ldots, \varepsilon_K$, the maximum margin is given by

$$\sum_{m=2}^{K} \frac{\varepsilon_m^{3/2}}{3\sqrt{3}|G|^{3/2}} \frac{1}{\sqrt{d_{R_m}}} \left[1 - \sum_{n=2}^{K} \frac{\tau_{C_n}|C_n|\chi_{R_m}(C_n)}{d_{R_m}}\right]$$

We want to maximize the expression above under the constraint that $\varepsilon_m \geq 0 \quad \forall m \in \{2, \ldots, K\}$ and $\sum \varepsilon_m \leq 1$.

Clearly, this is maximized only when one of the $\varepsilon_i = 1$ and everything else is 0, with

$$i \in \arg\max_{m=2,..,K} \frac{1}{\sqrt{d_{R_m}}} \left[1 - \sum_{n=2}^{K} \frac{\tau_{C_n}|C_n|\chi_{R_m}(C_n)}{d_{R_m}}\right].$$

□

Up until this point, we have kept our weighted margin problem generic without setting the $\tau_{C_n}$. If we naively chose $\tau_{C_n}$ to weigh the conjugacy classes uniformly, then the maximizers for this specific weighted margin would be only neuron weights spanned by the sign representation (of dimension 1). However, we cannot hope to correctly classify all pairs $a, b \in G$ using only the sign representation for our network $\theta^*$ and thus the maximizers for this weighted margin cannot be the maximizers for the original max margin problem. The next lemma establishes an appropriate assignment for each $\tau_{C_n}$ such that the expression in Lemma D.13 is equal for all non-trivial representations $R$, provided some conditions pertaining to the group are satisfied. Since the function $g \mapsto \tau_C$ (where $C$ is the conjugacy class containing $g$) is a class function, each $\tau_{C_n}$ can be expressed as a linear combination of characters $\chi_R(C_n)$.

**Lemma D.14.** *For the group $G$ if we have $\sum_R d_R^{1.5}\chi_R(C) < 0$ for every non-trivial conjugacy class $C$, then the weights $\tau_{C_n}$ can be set as*

$$\tau_{C_n} = \sum_R z_R \chi_R(C_n)$$

*where* $z_{R_{triv}} = 0$ *and* $z_R = \frac{d_R^{1.5}}{\sum_{m=2}^{K} d_{R_m}^{2.5}}$ *otherwise, such that the maximum value from Lemma D.13 is equal for all non-trivial representations* $R$.

*Proof.* Define the vectors $\tau$ and $\text{char}(R)$ as

$$\text{char}(R) = [\chi_R(C_1), \underbrace{\chi_R(C_2), \ldots, \chi_R(C_2)}_{|C_2| \text{ times}}, \ldots \underbrace{\chi_R(C_K), \ldots, \chi_R(C_K)}_{|C_K| \text{ times}}],$$

$$\tau = [1, \underbrace{-\tau_{C_2}, \ldots, -\tau_{C_2}}_{|C_2| \text{ times}}, \ldots \underbrace{-\tau_{C_K}, \ldots, -\tau_{C_K}}_{|C_K| \text{ times}}].$$

Then we can rewrite the max value of the weighted margin in Lemma D.13 as

$$\frac{1}{|G|^{3/2} 3\sqrt{3 d_R}} \left[ \frac{1}{d_R} \text{char}(R)^T \tau \right] \tag{D.15}$$

for each non-trivial representation $R$. Since $\tau$ is a class function (viewed as a function on $G$), we can express $\tau$ as a linear combination $\tau = \sum_{n=1}^{K} z_{R_n} \text{char}(R_n)$ of character vectors for each representation. By orthogonality, the inner product $\text{char}(R)^T \tau = z_R$. Thus for the expression (D.15) to be equal for every non-trivial representation $R$, we require

$$z_R = d_R^{3/2} z_{R_{\text{sign}}}.$$

Furthermore, since $1 - \sum_{n=2}^{K} \tau_{C_n} = 0$ and $\text{char}(R_{\text{triv}})$ is a vector with strictly positive values that is orthogonal to all other character vectors, we must have $z_{R_{\text{triv}}} = 0$. To solve for $z_{R_{\text{sign}}}$, since the first component of $\tau$ equals 1 and $\chi_R(C_1) = d_R$ for all $R$, we have

$$\sum_{m=2}^{K} z_{R_m} d_{R_m} = \sum_{m=2}^{K} d_{R_m}^{2.5} z_{R_{\text{sign}}} = 1 \implies z_{R_{\text{sign}}} = \sum_{m=2}^{K} d_{R_m}^{2.5}.$$

To conclude the proof, note that we need the weights $\tau_{C_n}$ to be positive; this is guaranteed as long as

for each conjugacy class $C$, we have $\sum_{n=2}^{K} \chi_{R_n}(C) {d_{R_n}}^{3/2} < 0$ (recall the entries of $\tau$ being $-\tau_{C_n}$).

□

Up until now, we have established a weighted margin problem and proven that the neurons which maximize this are spanned by only one representation out of any of the non-trivial representations. Now we give a precise construction of the neuron weights $u, v, w$ such that they implement $\mathrm{tr}(R(a)R(b)R(c)^{-1})$ for all inputs $a, b \in G$ and outputs $c \in G$ for a given representation $R$. These neuron weights expressed in terms of the basis vectors will have coefficients that also maximize $\mathrm{tr}(\alpha_R \beta_R {\gamma_R}^T)$.

**Lemma D.15.** *For every non-trivial representation $R$, there exists a construction of the network weights such that given inputs $a, b \in G$, the output at $c$ is $\mathrm{tr}(R(a)R(b)R(c)^{-1})$ using $2{d_R}^3$ neurons and the corresponding coefficients $\alpha_R, \beta_R, \gamma_R$ for each neuron achieve the maximum value $\mathrm{tr}(\alpha_R \beta_R {\gamma_R}^T) = (d_R/3|G|)^{3/2}$.*

*Proof.* Since the representations are unitary, we have

$$\mathrm{tr}(R(a)R(b)R(c)^{-1}) = \mathrm{tr}(R(a)R(b)R(c)^T) = \sum_{i,j,k} R(a)_{(i,j)} R(b)_{(j,k)} R(c)_{(i,k)}$$

and thus it suffices to show how to obtain $R(a)_{(i,j)} R(b)_{(j,k)} R(c)_{(i,k)}$ with a combination of neurons. For this, set one neuron's coefficients to equal $\alpha_{(i,j)} = \beta_{(j,k)} = \gamma_{(i,j)} = 1/\sqrt{3|G|}$ and 0 otherwise. Then the output given $(a, b)$ at $c$ is

$$\frac{(R(a)_{(i,j)} + R(b)_{(j,k)})^2 R(c)_{(i,k)}}{(3|G|)^{3/2}}.$$

Set another neuron's coefficients to equal $\alpha_{(i,j)} = 1/\sqrt{3|G|}, \beta_{(j,k)} = \gamma_{(i,k)} = -1/\sqrt{3|G|}$. Then

the sum of the outputs of these two neurons at $c$ is precisely

$$\frac{R(a)_{(i,j)}R(b)_{(j,k)}R(c)_{(i,k)}}{(3|G|)^{3/2}}.$$

Thus we need $2d_R^3$ neurons to create the summand for each $i, j, k$ to implement $\mathrm{tr}(R(a)R(b)R(c)^{-1})$. This construction also satisfies $\mathrm{tr}(\alpha_R \beta_R {\gamma_R}^T) = (d_R/3|G|)^{3/2}$ for every neuron. □

Once we have defined these neuron constructions, it only remains to scale these optimal neurons appropriately as given in Lemma 7.5 such that we can construct our final network $\theta^*$ satisfying condition **C.1** and Equation 7.1.

**Lemma D.16.** *Given the network given in Lemma D.15, for every neuron spanned by non-trivial representation $R$ we scale the weights $u, v, w$ by $d_R^{1/3}/\Delta$, where $\Delta$ is a constant normalization term such that the norm constraints of the max margin problem still hold. Then the expected output of any element contained in any non-trivial conjugacy class $C$ for inputs $a, b$ is $-1/\Delta^3$, i.e. the output is equal for all conjugacy classes.*

*Proof.* For a given neuron spanned by a non-trivial representation $R$, we know that its output for at $c$ for each input pair $(a, b)$ is $\chi_R(abc^{-1}) = \chi_R(C)$ where $C$ is the conjugacy class containing $abc^{-1}$. After scaling each weight by $d_R^{1/3}/\Delta$, the corresponding output is scaled by $d_R/\Delta^3$. Due to column orthogonality of the characters with the trivial conjugacy class (i.e. $\sum_{n=1}^K \chi_{R_n}(e)\chi_{R_n}(C) = 0$ for for all non-trivial conjugacy classes $C$), this output simplifies to

$$\sum_{n=2}^{K} \frac{d_{R_n}\chi_R(C)}{\Delta^3} = -\frac{1}{\Delta^3}\sum_{n=2}^{K} \frac{d_{R_n}\chi_R(C)}{\Delta^3} = -\frac{1}{\Delta^3}, \tag{D.16}$$

which is constant for all non-trivial conjugacy classes $C$.

□

With this lemma, we define the network $\theta^*$ according to this scaling and guarantee that it satisfies C.1 and Equation 7.1. Applying Lemma 7.5 gives us our final result that the solutions for the max margin problem have the desired properties in Theorem 7.9.

### D.9.1 Proof that all representations are used

This proof follows exactly the same argument as for the modular addition case (Section D.6.3).

For this proof, we will introduce the multidimensional Fourier transform for groups. For a function $f : G^3 \to \mathbb{R}$, this is defined as

$$\hat{f}(j,k,l) = \sum_{a \in |G|} \rho_j(a) \sum_{b \in |G|} \rho_k(b) \sum_{c \in |G|} \rho_l(c) f(a,b,c)$$

Similar to the modular addition case, for a single margin maximizing neuron, we know it uses only one of the representations for input and output neurons, let's say $R_m$. Then, considering just the basis vectors with respect to $R_m$, we can say, that the output of this neuron is given by

$$f(a,b,c) = \left[ \sum_{i \in d_{R_m}} \sum_{j \in d_{R_m}} \alpha_{(i,j)} \rho_{(i,j)}[a] + \beta_{(i,j)} \rho_{(i,j)}[b] \right]^2 \left( \sum_{k \in d_{R_m}} \sum_{l \in d_{R_m}} \gamma_{(k,l)} \rho_{(k,l)}[c] \right)$$

Now, for the squared terms, these are either dependent on $a, c$ or $b, c$. These have non-zero fourier coefficients only if $j = 0$ or $k = 0$.

For the cross terms, by orthogonality of the representatons, we can say, if either $j, k$ or $l$ does not belong to $R_m$, then $\hat{f}(j,k,l) = 0$.

Thus, for a single neuron, $\hat{f}(j,k,l)$ is only non-zero if $j = 0, k = 0$ or if $j, k$ and $l$ belong to the same representation.

Independent of the above considerations, we know by Lemma 7.6 that the function $f$ implemented by the network has equal margin across different inputs and across different classes for the

same input. In other words, $f$ can be decomposed as

$$f(a,b,c) = f_1(a,b,c) + f_2(a,b,c)$$

where

$$f_1(a,b,c) = F(a,b)$$

for some $F : G \times G \to \mathbb{R}$, and

$$f_2(a,b,c) = \lambda \cdot \mathbf{1}_{a \circ b = c}$$

where $\lambda > 0$ is the margin of $f$.

The Fourier transform of $f_1$ is

$$\hat{f_1}(j,k,l) = \begin{cases} \hat{F}(j,k) & \text{if } \ell = 0 \\ 0 & \text{otherwise} \end{cases}$$

For $f_2$, consider the expression of the fourier transform:

$$\hat{f_2}(j,k,l) = \lambda \sum_{a \in |G|} \rho_j(a) \sum_{b \in |G|} \rho_k(b) \rho_l(a \circ b)$$

Now, $\rho_l(a \circ b) = \sum \rho_{l'}(a) \rho_{k'}(b)$ for some $j', k'$ given by the relation that $R(a \circ b) = R(a)R(b)$, where $R(a)R(b)$ denoted the matrix product of $R(a)$ and $R(b)$. Now, clearly if $j, k$ and $l$ belong to different representations, then $\hat{f_2}(j,k,l)$ is 0. For $j, k, l$ belonging to the same representation, $\hat{f_2}(j,k,l)$ will be non-zero whenever $j = j'$ and $k = k'$ (or $j = k'$ and $k = j'$), and the value will be given by $\lambda |G|^2 / d_{R_m}^2$. Thus, $f = f_1 + f_2$ has support on all the representations.

But, this is only possible if there is atleast one neuron for each representation, as a single neuron places non-zero fourier mass only on one of the representation.

### D.9.2 A General Theorem for Finite Groups

As mentioned in section 7.6, Theorem 7.9 does not hold for all groups because of the required condition that $\sum_{n=2}^{K} d_{R_n}^{1.5} \chi_{R_n}(C) < 0$ for every non-trivial conjugacy class. Recall that in the previous section, we had to define an appropriate weighting over *all* conjugacy classes such that the margin of a neuron did not scale down with the dimension of the neuron's spanning representation. We also had to define an appropriate scaling over *all* representations so we could use the neuron maximizers of the weighted margin to construct a network $\theta^*$ to invoke Lemma 7.5. This is akin to selecting the entire character table for our margin analysis; in this section, we show how our analysis is amenable to selecting a *subset* of the character table for the margin analysis of a general finite group $G$, which can lead to a max margin solution in the same way as above. This will occur upon solving a system of two linear equations, as long as these solutions satisfy some conditions.

Namely, let $\kappa_R, \kappa_C \subset [K] \setminus \{1\}$ be subsets indicating which representations and which conjugacy classes will be considered in the scaling and weighting respectively, with $|\kappa_R| = |\kappa_C|$. If we view the character table as a matrix and consider the square submatrix pertaining to only the representations indexed by elements in $\kappa_R$ and the conjugacy classes indexed by elements in $\kappa_C$, the rows are $\chi_{R_m}$ for fixed $m \in \kappa_R$ and the columns are $[\chi_{R_m}(C_n)]_{m \in \kappa_R}$ for fixed $n \in \kappa_C$.

Instead of requiring expression (D.15) to be equal for all representations in the proof of Lemma D.14, we can instead require that they are equal across representations in $\kappa_R$. To be precise, consider the following set of equations over variables $\tau_{C_n}, n \in \kappa_C$:

$$\left(1 - \sum_{n \in \kappa_C} \frac{\tau_{C_n} |C_n| \chi_{R_m}}{d_{R_m}}\right) = \sqrt{\frac{d_{R_m}}{d_{R_{m'}}}} \left(1 - \sum_{n \in \kappa_C} \frac{\tau_{C_n} |C_n| \chi_{R_{m'}}}{d_{R_{m'}}}\right) \quad \forall\, m, m' \in \kappa_R,$$

$$\sum_{n \in \kappa_C} \tau_{C_n} = 1.$$

This gives a system of $|\kappa_C|$ linear equations in $|\kappa_C|$ variables. Let the solution be denoted as $\tau_{C_n}^*$ for

each $n \in \kappa_C$.

Furthermore, just as we established in equation D.16, we can identify a scaling dependent on each representation such that the output remains constant for all conjugacy classes in $\kappa_C$ and such that if we had used this scaling for neurons maximizing the weighted margin, the $L_{2,3}$ norm constraint is maintained. This can be represented using the following set of equations with variables $\lambda_{R_m}$:

$$\sum_{m \in \kappa_R} \lambda_{R_m} \chi_{R_m}(C_n) = \sum_{m \in \kappa_R} \lambda_{R_m} \chi_{R_m}(C_{n'}) \; \forall \; n, n' \in \kappa_C$$

$$\sum_{m \in \kappa_R} \lambda_{R_m} = 1.$$

This again forms a system of $|\kappa_R|$ linear equations in $|\kappa_R|$ variables. Let the solution be denoted as $\lambda^*_{R_m}$. Suppose the following conditions are satisfied:

1. The weighting and scaling are positive: $\lambda^*_{R_m}, \tau^*_{C_n} \geq 0$ for all $m \in \kappa_R, n \in \kappa_C$.

2. For any $m \in \kappa_R$ and $m' \notin \kappa_R$, we have

$$\left(1 - \sum_{n \in \kappa_C} \frac{\tau_{C_n} |C_n| \chi_{R_m}}{d_{R_m}}\right) \geq \sqrt{\frac{d_{R_m}}{d_{R_{m'}}}} \left(1 - \sum_{n \in \kappa_C} \frac{\tau_{C_n} |C_n| \chi_{R_{m'}}}{d_{R_{m'}}}\right).$$

3. For any $n \in \kappa_C$ and $n' \notin \kappa_C$, we have

$$\sum_{m \in \kappa_R} \lambda_{R_m} \chi_{R_m}(C_n) \geq \sum_{m \in \kappa_R} \lambda_{R_m} \chi_{R_m}(C_{n'}).$$

The second condition ensures that the representations in $\kappa_R$ indeed maximize the weighted margin, and no other representations maximize it. The third condition above ensures that the conjugacy classes in $\kappa_C$ are on the margin, and no other conjugacy class can be on the margin. Then it follows

that neurons spanned by the representations in $\kappa_R$ will maximize the weighted margin defined using $\tau^*$ with all conjugacy classes in $\kappa_C$ on the margin, and thus scaling these neurons by $\lambda^*$, we have a network $\theta^*$ that is a max margin solution for the group $G$.

# E

# Induction Heads

## E.1 Proofs

In this section, we present our theoretical results on in-context learning Markov Chains with the minimal model of Section 8.2.3.

Setup and notation Our data consists of sequences of length $t$, $\boldsymbol{x} = (x_1, \ldots, x_t)$, drawn from a Markov Chain with state space $S = \{1, \ldots, k\}$ (i.e., $x_j \in \{1, \ldots, k\}$ for all $j \in [t]$), and a

random transition matrix $\mathcal{P}$. Each row of the matrix is sampled from a Dirichlet distribution with concentration parameter $\alpha$, i.e. $\mathcal{P}_{i,:} \sim \text{Dir}(\alpha)$. Unless stated otherwise, we set $\alpha = (1, \ldots, 1)^\top$, corresponding drawing the row from a uniform distribution over the simplex. Let $e_{x_p} \in \{0,1\}^k$ denote the one-hot embedding of the state at position $p \in [t]$ and let $e \in \mathbb{R}^{t \times k}$ be the embedding matrix. We assume there are 2 states in the Markov Chain, i.e. $k = 2$. In that case, the transition matrix can be parameterized as:

$$\mathcal{P} = \begin{pmatrix} a & 1-a \\ 1-b & b \end{pmatrix}, \tag{E.1}$$

where $a, b \sim Unif(0,1)$.

Model We define our model as a simplified sequence to sequence linear transformer $f : \mathbb{R}^{t \times k} \to \mathbb{R}^{t \times k}$ with $f(e) = \text{mask}\left(eW_k(Me)^T\right)e$. It is $W_k \in \mathbb{R}^{k \times k}$ and $M = \begin{pmatrix} v_1 & 0 & \ldots & 0 \\ v_2 & v_1 & \ldots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ v_t & v_{t-1} & \ldots & v_1 \end{pmatrix}$, where $v = [v_1, v_2, \ldots, v_t] \in \mathbb{R}^t$. Equivalently, we can express the $i$-th logit for the $p$-th position as:

$$f(e)_{p,i} = \sum_{t'=1}^{p} 1\{x_{t'} = i\} \sum_{s=1}^{t'} v_{t'-s+1} e_{x_p}^\top W_k e_{x_s}. \tag{E.2}$$

The model can represent the unigrams and bigrams solutions as following:

- Construction for bigrams: $v = (0,1,0,\ldots,0)^\top$ and $W_k = I_{k\times k}$, then $f(e)_{p,i} = \sum_{t'=2}^{p} 1\{x_{t'} = i\}\, 1\{x_{t'-1} = x_p\}$.
- Construction for unigrams: $v = (1,0,0,\ldots,0)^\top$ and $W_k = 11^T$ (all ones), then $f(e)_{p,i} = \sum_{t'=1}^{p} 1\{x_{t'} = i\}$.

TRAINING We analyze stochastic gradient descent training with the margin loss $l_M(f(e)_{p,:}, x_{p+1}) = \frac{1}{k}\sum_{i=1, i\neq x_{p+1}}^{k} \max\left\{0, \Delta + f(e)_{p,i} - f(e)_{p,x_{p+1}}\right\}$. The total loss (sum of the losses across all positions) is given by

$$L(f(e), e) = \left[\frac{1}{t}\sum_{p=1}^{t} l_M(f(e)_{p,:}, x_{p+1})\right], \tag{E.3}$$

and the population loss:

$$\mathcal{L} = \mathop{\mathbb{E}}_{\substack{\boldsymbol{x}\sim\mathcal{P} \\ \mathcal{P}\sim\mathrm{Dir}(\alpha)^{\otimes k}}} L(f(e), e), \tag{E.4}$$

where recall $e$ depends on $\boldsymbol{x}$.

We next compute gradients of the loss for the first two steps.

**Lemma E.1.** *Let the model defined as in eq.* (E.2) *and initialized with* $W_k = c11^T, v = c1^T$. *Then, after one step of gradient descent on the population loss* (E.4) *we have:*

$$\begin{aligned} W_k^{(1)} &= \begin{pmatrix} c & c \\ c & c \end{pmatrix} + c\eta \left[ O(t^2) \begin{pmatrix} B & A \\ A & B \end{pmatrix} + O(t) \right] \\ v_j^{(1)} &= c + \frac{c\eta}{t}\left[\frac{(t-j+1)(t-j+2)}{2} D + O(t)\right], j \in [t], \end{aligned} \tag{E.5}$$

*where* $A, B, D > 0$ *with* $B \approx 4A$ *(diagonal bias) and* $\eta$ *is the learning rate. After the second step,* $v_2^{(2)}$ *becomes dominant, i.e.* $v_2^{(2)} > v_j^{(2)}, j = 1, 3, 4, \ldots, t$.

*Proof.* **First step**. We analyze the first step of stochastic gradient descent. The function at initialization is

$$\begin{aligned} f^0(e)_{p,i} &= c^2 \sum_{t'=1}^{p} 1\{x_{t'} = i\} \sum_{s=1}^{t'} e_{x_p}^\top 11^T e_{x_s} \\ &= c^2 \sum_{t'=1}^{p} 1\{x_{t'} = i\}\, t' \quad \in [0, c^2\frac{p(p+1)}{2}]. \end{aligned} \tag{E.6}$$

By choosing $\Delta \geq c^2 \frac{t(t+1)}{2}$, we ensure that $\Delta + f^0(e)_{p,i} - f^0(e)_{p,x_{p+1}} \geq 0$, for all $p \in [t]$. From the total law of expectation it is:

$$\mathop{\mathbb{E}}_{\substack{\boldsymbol{x}\sim\mathcal{P}\\ \mathcal{P}\sim\mathrm{Dir}(\alpha)^{\otimes k}}}[L] = \mathbb{E}_{\mathcal{P}\sim\mathrm{Dir}(\alpha)^{\otimes k}}\left[\mathbb{E}_{\boldsymbol{x}\sim\mathcal{P}}\left[L|\mathcal{P}\right]\right]. \tag{E.7}$$

We will first focus on the inner conditional expectation and at the end of the calculations we will take expectation with respect the transition matrix. In what follows, unless otherwise stated, $\mathbb{E}$ and $\mathbb{P}$ will be with respect to the randomness of $\boldsymbol{x}$ conditioned on $\mathcal{P}$.

Then the gradient of the loss with respect to $W_k$ is:

$$\begin{aligned}
\nabla_{W_k}\mathbb{E}_{\boldsymbol{x}\sim\mathcal{P}}\left[L|\mathcal{P}\right] &= \frac{1}{t}\sum_{p=1}^{t}\frac{1}{k}\sum_{i=1,i\neq x_{p+1}}^{k}\mathbb{E}\left[1\left\{\Delta + f(e)_{p,i} - f(e)_{p,x_{p+1}} \geq 0\right\}\left(\nabla_{W_k}f(e)_{p,i} - \nabla_{W_k}f(e)_{p,x_{p+1}}\right)\right] \\
&= \frac{1}{t}\sum_{p=1}^{t}\frac{1}{k}\sum_{i=1,i\neq x_{p+1}}^{k}\mathbb{E}\left[\left(\nabla_{W_k}f(e)_{p,i} - \nabla_{W_k}f(e)_{p,x_{p+1}}\right)\right] \\
&= \frac{1}{t}\sum_{p=1}^{t}\frac{1}{k}\sum_{i=1}^{k}\mathbb{E}\left[\left(\nabla_{W_k}f(e)_{p,i} - \nabla_{W_k}f(e)_{p,x_{p+1}}\right)\right] \\
&= \frac{1}{t}\sum_{p=1}^{t}\left[\left(\frac{1}{k}\sum_{i=1}^{k}\mathbb{E}\left[\nabla_{W_k}f(e)_{p,i}\right]\right) - \mathbb{E}\left[\nabla_{W_k}f(e)_{p,x_{p+1}}\right]\right].
\end{aligned} \tag{E.8}$$

From equation (E.2), we have for the gradient of logit $i$:

$$\nabla_{W_k}f(e)_{p,i} = c\sum_{t'=1}^{p}1\left\{x_{t'} = i\right\}\sum_{s=1}^{t'}e_{x_p}e_{x_s}^T, \tag{E.9}$$

or, equivalently, its elements are:

$$\left(\nabla_{W_k} f(e)_{p,i}\right)_{m,l} = c \sum_{t'=1}^{p} 1\left\{x_{t'} = i\right\} \sum_{s=1}^{t'} 1\{x_p = m\} 1\{x_s = l\}, \tag{E.10}$$

and their expectation (with respect to $\boldsymbol{x}$ conditioned on $\mathcal{P}$) is:

$$\mathbb{E}\left[\left(\nabla_{W_k} f(e)_{p,i}\right)_{m,l}\right] = c \sum_{t'=1}^{p} \sum_{s=1}^{t'} \mathbb{P}\left[x_s = l, x_{t'} = i, x_p = m\right]. \tag{E.11}$$

Similarly, for the ground truth logit:

$$\mathbb{E}\left[\left(\nabla_{W_k} f(e)_{p,x_{p+1}}\right)_{m,l}\right] = c \sum_{t'=1}^{p} \sum_{s=1}^{t'} \mathbb{P}\left[x_s = l, x_{t'} = x_{p+1}, x_p = m\right]. \tag{E.12}$$

Thus, by substituting back to eq. (E.8) we have :

$$\begin{aligned}
-\nabla_{W_k} \mathbb{E}_{\boldsymbol{x}\sim\mathcal{P}}\left[L|\mathcal{P}\right] &= \frac{c}{t} \sum_{p=1}^{t} \sum_{t'=1}^{p} \sum_{s=1}^{t'} \mathbb{P}\left[x_s = l, x_{t'} = x_{p+1}, x_p = m\right] - \frac{c}{tk} \sum_{i=1}^{k} \sum_{p=1}^{t} \sum_{t'=1}^{p} \sum_{s=1}^{t'} \mathbb{P}\left[x_s = l, x_{t'} = i, x_p = m\right] \\
&= \frac{c}{t} \sum_{i=1}^{k} \sum_{p=1}^{t} \sum_{t'=1}^{p} \sum_{s=1}^{t'} \mathbb{P}\left[x_{p+1} = i, x_s = l, x_{t'} = i, x_p = m\right] - \frac{c}{tk} \sum_{i=1}^{k} \sum_{p=1}^{t} \sum_{t'=1}^{p} \sum_{s=1}^{t'} \mathbb{P}\left[x_s = l, x_{t'} = i, x_p = \right. \\
&= \frac{c}{t} \sum_{i=1}^{k} \left(\mathcal{P}_{mi} - \frac{1}{k}\right) \sum_{p=1}^{t} \sum_{t'=1}^{p} \sum_{s=1}^{t'} \mathbb{P}\left[x_s = l, x_{t'} = i, x_p = m\right] \\
&= \frac{c}{t} \sum_{i=1}^{k} \left(\mathcal{P}_{mi} - \frac{1}{k}\right) \sum_{p=1}^{t} \sum_{t'=1}^{p} \sum_{s=1}^{t'} \mathbb{P}\left[x_p = m \mid x_{t'} = i\right] \mathbb{P}\left[x_{t'} = i \mid x_s = l\right] \mathbb{P}\left[x_s = l\right] \\
&= \frac{c}{t} \sum_{i=1}^{k} \left(\mathcal{P}_{mi} - \frac{1}{k}\right) \sum_{p=1}^{t} \sum_{t'=1}^{p} \sum_{s=1}^{t'} \left(\mathcal{P}^{p-t'}\right)_{im} \left(\mathcal{P}^{t'-s}\right)_{li} \pi_l,
\end{aligned} \tag{E.13}$$

where we used the Markov property and the assumption that the chain has reached its stationary

distribution $\pi \in \mathbb{R}^{1\times k}$, so $\mathbb{P}[x_t = i] = \pi_i$ for all $t > 0, i \in [k]$. Recall that, in the case of $k = 2$, the (random) transition matrix can be parameterized as:

$$\mathcal{P} = \begin{pmatrix} a & 1-a \\ 1-b & b \end{pmatrix}, \tag{E.14}$$

where $a, b \sim Unif([0,1])$. $\mathcal{P}$ is almost surely diagonalizable, and using its eigendecomposition, we can calculate its $n$-th powers

$$\begin{aligned} \mathcal{P}^n &= \frac{1}{a+b-2} \begin{pmatrix} 1 & \frac{1-a}{b-1} \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & a+b-1 \end{pmatrix}^n \begin{pmatrix} b-1 & a-1 \\ 1-b & b-1 \end{pmatrix} \\ &= \frac{1}{a+b-2} \begin{pmatrix} (a-1)(a+b-1)^n + b - 1 & (1-a)(a+b-1)^n + a - 1 \\ (1-b)(a+b-1)^n + b - 1 & (b-1)(a+b-1)^n + a - 1 \end{pmatrix}. \end{aligned} \tag{E.15}$$

We observe that every element of the matrix, $(\mathcal{P}^n)_{ij}$, is of the form $\frac{\beta_{ij}\lambda^n + \gamma_{ij}}{\lambda - 1}$, with $\lambda = a + b - 1$. It is also $\pi = \frac{1}{\lambda-1}\begin{pmatrix} b-1 & a-1 \end{pmatrix}$ (the normalized eigenvector that corresponds to $\lambda$). Thus, eq. (E.13) can be written as

$$\begin{aligned} -\nabla_{W_k} \mathbb{E}_{\boldsymbol{x}\sim\mathcal{P}}[L|\mathcal{P}] &= \frac{c}{t(\lambda-1)^2} \sum_{i=1}^{2} \left(\mathcal{P}_{mi} - \frac{1}{2}\right) \sum_{p=1}^{t} \sum_{t'=1}^{p} \sum_{s=1}^{t'} \left(\beta_{im}\lambda^{p-t'} + \gamma_{im}\right)\left(\beta_{li}\lambda^{t'-s} + \gamma_{li}\right)\pi_l \\ &= \frac{c\pi_l}{t(\lambda-1)^2} \sum_{i=1}^{2} \left(\mathcal{P}_{mi} - \frac{1}{2}\right) \sum_{p=1}^{t} \sum_{t'=1}^{p} \sum_{s=1}^{t'} \left( \underbrace{\beta_{im}\beta_{li}\lambda^{p-s}}_{(A)} + \underbrace{\beta_{im}\gamma_{li}\lambda^{p-t'}}_{(B)} + \underbrace{\beta_{li}\gamma_{im}\lambda^{t'-s}}_{(C)} + \underbrace{\gamma_{im}\gamma_{li}}_{(D)} \right). \end{aligned} \tag{E.16}$$

We calculate the four terms separately:

$$
\begin{aligned}
(A) &= \sum_{p=1}^{t}\sum_{t'=1}^{p}\sum_{s=1}^{t'}\beta_{im}\beta_{li}\lambda^{p-s} \\
&= \beta_{im}\beta_{li}\sum_{p=1}^{t}\sum_{t'=1}^{p}\sum_{s=1}^{t'}\lambda^{p-s} \\
&= \beta_{im}\beta_{li}\sum_{p=1}^{t}\sum_{t'=1}^{p}\frac{\lambda^{p-1}-\lambda^{p-t'-1}}{1-\lambda^{-1}} \\
&= \frac{\beta_{im}\beta_{li}}{1-\lambda^{-1}}\sum_{p=1}^{t}\left(p\lambda^{p-1}-\frac{\lambda^{p-2}-\lambda^{-2}}{1-\lambda^{-1}}\right) \\
&= \frac{\beta_{im}\beta_{li}}{1-\lambda^{-1}}\left(\frac{1-(t+1)\lambda^{t}+t\lambda^{t+1}}{(1-\lambda)^2}-\frac{\lambda^{-1}-\lambda^{t-1}}{\left(1-\lambda^{-1}\right)(1-\lambda)}+t\frac{\lambda^{-2}}{1-\lambda^{-1}}\right) \\
&= \frac{\beta_{im}\beta_{li}}{\lambda-1}\left(\frac{\lambda-(t+1)\lambda^{t+1}+t\lambda^{t+2}}{(1-\lambda)^2}-\frac{1-\lambda^{t}}{\left(1-\lambda^{-1}\right)(1-\lambda)}+t\frac{1}{\lambda-1}\right). \\
(B) &= \sum_{p=1}^{t}\sum_{t'=1}^{p}\sum_{s=1}^{t'}\beta_{im}\gamma_{li}\lambda^{p-t'} \\
&= \beta_{im}\gamma_{li}\sum_{p=1}^{t}\sum_{t'=1}^{p}\sum_{s=1}^{t'}\lambda^{p-t'} \\
&= \beta_{im}\gamma_{li}\sum_{p=1}^{t}\sum_{t'=1}^{p}t'\lambda^{p-t'} \\
&= \beta_{im}\gamma_{li}\sum_{p=1}^{t}\lambda^{p-1}\frac{1-(p+1)\lambda^{-p}+p\lambda^{-p-1}}{(1-\lambda^{-1})^2} \\
&= \beta_{im}\gamma_{li}\sum_{p=1}^{t}\frac{\lambda^{p-1}-(p+1)\lambda^{-1}+p\lambda^{-2}}{(1-\lambda^{-1})^2} \\
&= \frac{\beta_{im}\gamma_{li}}{(1-\lambda^{-1})^2}\left(\frac{1-\lambda^{t}}{1-\lambda}-\lambda^{-1}\frac{t(t+3)}{2}+\lambda^{-2}\frac{t(t+1)}{2}\right) \\
&= \frac{\beta_{im}\gamma_{li}}{(\lambda-1)^2}\left(\lambda^{2}\frac{1-\lambda^{t}}{1-\lambda}-\lambda\frac{t(t+3)}{2}+\frac{t(t+1)}{2}\right). \\
(C) &= \sum_{p=1}^{t}\sum_{t'=1}^{p}\sum_{s=1}^{t'}\beta_{li}\gamma_{im}\lambda^{t'-s} \\
&= \beta_{li}\gamma_{im}\sum_{p=1}^{t}\sum_{t'=1}^{p}\sum_{s=1}^{t'}\lambda^{t'-s} \\
&= \beta_{li}\gamma_{im}\sum_{p=1}^{t}\sum_{t'=1}^{p}\frac{\lambda^{t'-1}-\lambda^{-1}}{1-\lambda^{-1}} \\
&= \frac{\beta_{li}\gamma_{im}}{1-\lambda^{-1}}\sum_{p=1}^{t}\left(\frac{1-\lambda^{p}}{1-\lambda}-\lambda^{-1}p\right) \\
&= \frac{\beta_{li}\gamma_{im}}{1-\lambda^{-1}}\left(\frac{t-(t+1)\lambda^{t+1}}{(1-\lambda)^2}\right)
\end{aligned}
$$

$$(D) = \sum_{p=1}^{t}\sum_{t'=1}^{p}\sum_{s=1}^{t'} \gamma_{im}\gamma_{li} = \gamma_{im}\gamma_{li}\frac{t(t+1)(t+2)}{6}.$$

Plugging these expressions back into eq. (E.16), we get:

$$\begin{aligned}-\nabla_{W_k}\mathbb{E}_{\boldsymbol{x}\sim\mathcal{P}}\left[L|\mathcal{P}\right] = \frac{c\pi_l\left(\mathcal{P}_{m1}-\frac{1}{2}\right)}{t(\lambda-1)^2}\Bigg[&\frac{\beta_{1m}\beta_{l1}-\beta_{2m}\beta_{l2}}{\lambda-1}\underbrace{\left(\frac{\lambda-(t+1)\lambda^{t+1}+t\lambda^{t+2}}{(1-\lambda)^2}-\frac{1-\lambda^t}{\left(1-\lambda^{-1}\right)(1-\lambda)}+t\frac{1}{\lambda-1}\right)}_{(I)}\\&+\frac{\beta_{1m}\gamma_{l1}-\beta_{2m}\gamma_{l2}}{(\lambda-1)^2}\underbrace{\left(\lambda^2\frac{1-\lambda^t}{1-\lambda}-\lambda\frac{t(t+3)}{2}+\frac{t(t+1)}{2}\right)}_{(II)}\\&+\frac{\beta_{l1}\gamma_{1m}-\beta_{l2}\gamma_{2m}}{\lambda-1}\underbrace{\left(t\frac{\lambda}{1-\lambda}-\frac{\lambda^2-\lambda^{t+2}}{(1-\lambda)^2}-\frac{t(t+1)}{2}\right)}_{(III)}\\&+\left(\gamma_{1m}\gamma_{l1}-\gamma_{2m}\gamma_{l2}\right)\underbrace{\frac{t(t+1)(t+2)}{6}}_{(IV)}\Bigg].\end{aligned} \tag{E.17}$$

Finally, by taking expectation over $a, b$ (the randomness of the transition matrix), we get for the update of $W_k$ with learning rate $\eta$:

$$\left(W_k^{(1)}\right)_{m,l} = \left(W_k^{(0)}\right)_{m,l} - \eta\nabla_{W_k}\mathbb{E}_{a,b}\left[\mathbb{E}_{\boldsymbol{x}\sim\mathcal{P}}\left[L|\mathcal{P}\right]\right]. \tag{E.18}$$

We consider the 4 cases separately. Case $m, l = 1, 1$:

$$\begin{aligned}\left(W_k^{(1)}\right)_{1,1} = c + \frac{\eta c}{t}\mathbb{E}_{a,b}\Bigg[&\frac{(b-1)(a-\frac{1}{2})}{(\lambda-1)^3}\Bigg[\frac{(a-1)(a-b)}{\lambda-1}(I)+\frac{2(a-1)(b-1)}{(\lambda-1)^2}(II)\\&+\frac{2(a-1)(b-1)}{\lambda-1}(III)+(b-1)(b-a)(IV)\Bigg]\Bigg]\end{aligned} \tag{E.19}$$

Case $m, l = 1, 2$:

$$\left(W_k^{(1)}\right)_{1,2} = c + \frac{\eta c}{t}\mathbb{E}_{a,b}\left[\frac{(a-1)(a-\frac{1}{2})}{(\lambda-1)^3}\left[\frac{(1-b)(a-b)}{\lambda-1}(I) + \frac{2(a-1)(b-1)}{(\lambda-1)^2}(II) - \frac{2(b-1)^2}{\lambda-1}(III) + (b-1)(b-a)(IV)\right]\right] \tag{E.20}$$

Case $m, l = 2, 1$:

$$\left(W_k^{(1)}\right)_{2,1} = c + \frac{\eta c}{t}\mathbb{E}_{a,b}\left[\frac{(b-1)(\frac{1}{2}-b)}{(\lambda-1)^3}\left[\frac{(1-a)(a-b)}{\lambda-1}(I) - \frac{2(a-1)(b-1)}{(\lambda-1)^2}(II) + \frac{2(a-1)^2}{\lambda-1}(III) + (a-1)(b-a)(IV)\right]\right] \tag{E.21}$$

Case $m, l = 2, 2$:

$$\left(W_k^{(1)}\right)_{2,2} = c + \frac{\eta c}{t}\mathbb{E}_{a,b}\left[\frac{(a-1)(\frac{1}{2}-b)}{(\lambda-1)^3}\left[\frac{(b-1)(a-b)}{\lambda-1}(I) - \frac{2(a-1)(b-1)}{(\lambda-1)^2}(II) - \frac{2(a-1)(b-1)}{\lambda-1}(III) + (a-1)(b-a)(IV)\right]\right] \tag{E.22}$$

See Figure E.1 (left) for an empirical estimation for some choice of hyperparameters. We observe that the gradient is symmetric, so it is going to be $\left(W_k^{(1)}\right)_{2,1} = \left(W_k^{(1)}\right)_{1,2}$, and the value in the diagonal is the same, i.e. $\left(W_k^{(1)}\right)_{1,1} = \left(W_k^{(1)}\right)_{2,2}$.

By only focusing on the leading order terms, $(IV)$, we have:

$$\left(W_k^{(1)}\right)_{m,l} = c + c\eta\mathbb{E}_{a,b}\left[\frac{\pi_m\pi_l}{(\lambda-1)}(b-a)\left(\mathcal{P}_{m1} - \frac{1}{2}\right)\right]\frac{(t+1)(t+2)}{6} + O(t). \tag{E.23}$$

A calculation then shows that

$$\mathbb{E}_{a,b}\left[\frac{\pi_1\pi_1}{(\lambda-1)}(b-a)\left(\mathcal{P}_{11}-\frac{1}{2}\right)\right]=\mathbb{E}_{a,b}\left[\frac{\pi_2\pi_2}{(\lambda-1)}(b-a)\left(\mathcal{P}_{21}-\frac{1}{2}\right)\right]$$
$$=\int_0^1\int_0^1\frac{(b-1)^2(b-a)(a-\frac{1}{2})}{(a+b-2)^3}dadb=\frac{\ln 256-5}{12}\approx 0.045, \tag{E.24}$$

and

$$\mathbb{E}_{a,b}\left[\frac{\pi_1\pi_2}{(\lambda-1)}(b-a)\left(\mathcal{P}_{11}-\frac{1}{2}\right)\right]=\mathbb{E}_{a,b}\left[\frac{\pi_1\pi_2}{(\lambda-1)}(b-a)\left(\mathcal{P}_{21}-\frac{1}{2}\right)\right]$$
$$=\int_0^1\int_0^1\frac{(b-1)(a-1)(b-a)(a-\frac{1}{2})}{(a+b-2)^3}dadb=\frac{7-10\ln 2}{6}\approx 0.011, \tag{E.25}$$

hence the diagonal grows a constant ($\approx 4$) times more than the off-diagonal.

Similarly, for the expected gradient of $v$ (with respect to $\boldsymbol{x}$ conditioned on $\mathcal{P}$), we have:

$$\begin{aligned}\mathbb{E}\left[\frac{\partial f(e)_{p,i}}{\partial v_j}\right]&=c\mathbb{E}\left[\sum_{t'=1}^{p}1\left\{x_{t'}=i\right\}\sum_{s=1}^{t'}\delta_{j(t'-s+1)}1\left\{j\le p\right\}\right]\\&=c\mathbb{E}\left[\sum_{t'=1}^{p}1\left\{x_{t'}=i\right\}1\left\{j\le t'\right\}1\left\{j\le p\right\}\right]\\&=c\sum_{t'=j}^{p}\mathbb{P}\left[x_{t'}=i\right]1\left\{j\le p\right\}\\&=c\pi_i\left(p-j+1\right)1\left\{j\le p\right\},\end{aligned} \tag{E.26}$$

and

$$\begin{aligned}
\mathbb{E}\left[\frac{\partial f(e)_{p,x_{p+1}}}{\partial v_j}\right] &= c\sum_{t'=j}^{p}\mathbb{P}\left[x_{t'}=x_{p+1}\right]1\left\{j\le p\right\}\\
&= c\sum_{t'=j}^{p}\sum_{i=1}^{k}\mathbb{P}\left[x_{t'}=x_{p+1}=i\right]1\left\{j\le p\right\}\\
&= c\sum_{i=1}^{k}\sum_{t'=j}^{p}\mathbb{P}\left[x_{p+1}=i\mid x_{t'}=i\right]\mathbb{P}\left[x_{t'}=i\right]1\left\{j\le p\right\}\\
&= c\sum_{i=1}^{k}\pi_i\sum_{t'=j}^{p}\left(\mathcal{P}^{p+1-t'}\right)_{ii}1\left\{j\le p\right\}\\
&= c\sum_{i=1}^{k}\frac{\pi_i}{\lambda-1}\sum_{t'=j}^{p}\left(\beta_{ii}\lambda^{p+1-t'}+\gamma_{ii}\right)1\left\{j\le p\right\}\\
&= c\left(\frac{b-1}{(\lambda-1)^2}\sum_{t'=j}^{p}\left((a-1)\lambda^{p+1-t'}+(b-1)\right)+\frac{a-1}{(\lambda-1)^2}\sum_{t'=j}^{p}\left((b-1)\lambda^{p+1-t'}+(a-1)\right)\right)1\left\{j\le p\right.\\
&= c\left(2\frac{(a-1)(b-1)}{(\lambda-1)^2}\frac{\lambda^{p+1-j}-1}{1-\lambda^{-1}}+\frac{(a-1)^2+(b-1)^2}{(\lambda-1)^2}(p-j+1)\right)1\left\{j\le p\right\}.
\end{aligned}\tag{E.27}$$

Thus, the update would be:

$$\begin{aligned}
v_j^{(1)} &= v_j^{(0)}-\eta\frac{\partial\mathcal{L}}{\partial v_j}\\
&= v_j^{(0)}+\frac{\eta}{t}\sum_{p=1}^{t}\left(\mathbb{E}_{a,b}\left[\mathbb{E}_{\boldsymbol{x}\sim\mathcal{P}}\left[\frac{\partial f(e)_{p,x_{p+1}}}{\partial v_j}\middle|\mathcal{P}\right]\right]-\frac{1}{k}\sum_{i=1}^{k}\mathbb{E}_{a,b}\left[\mathbb{E}_{\boldsymbol{x}\sim\mathcal{P}}\left[\frac{\partial f(e)_{p,x_i}}{\partial v_j}\middle|\mathcal{P}\right]\right]\right)\\
&= c+\frac{c\eta}{t}\sum_{p=j}^{t}\mathbb{E}_{a,b}\left[\frac{1}{(\lambda-1)^2}\left(\left((a-1)^2+(b-1)^2\right)(p-j+1)+2(a-1)(b-1)\frac{\lambda^{p-j+1}-1}{1-\lambda^{-1}}\right)-\frac{1}{2}(p-j+1)\right]\\
&= c+\frac{c\eta}{t}\mathbb{E}_{a,b}\left[\frac{(t-j+1)(t-j+2)}{2}\left(\frac{(a-1)^2+(b-1)^2}{(\lambda-1)^2}-\frac{1}{2}\right)+2\lambda\frac{(a-1)(b-1)}{(\lambda-1)^3}\left(\frac{\lambda-\lambda^{t-j+2}}{1-\lambda}-t+j-1\right)\right]
\end{aligned}\tag{E.28}$$

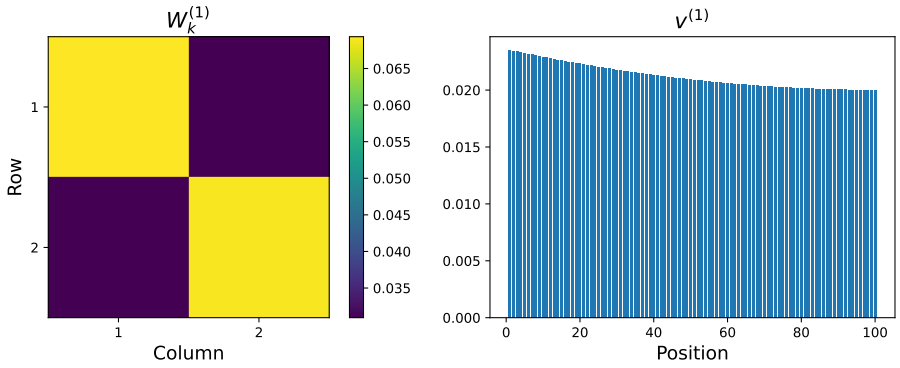



**Figure E.1:** Values of $W_k$ and $v$ after one step of stochastic gradient descent from eqs. (E.17),(E.28). Hyperparameters: initialization $c = 0.02$, learning rate $\eta = 0.03$, sequence length $t = 100$. The outer expectations are approximated using 10,000 samples.

See Figure E.1 (right) for an empirical estimation for some choice of hyperparameters. As we did in the case of the gradient of $W_k$, we focus on the leading order terms and we have:

$$v_j^{(1)} = c + \frac{c\eta}{t}\frac{(t-j+1)(t-j+2)}{2}\mathbb{E}_{a,b}\left[\frac{(a-1)^2+(b-1)^2}{(\lambda-1)^2} - \frac{1}{2}\right] + O(1), \tag{E.29}$$

and, since $\mathbb{E}_{a,b}\left[\frac{(a-1)^2+(b-1)^2}{(\lambda-1)^2}\right] = 2 - \ln 4 > \frac{1}{2}$, we get a positive bias with quadratic dependence.

**Second step.** We saw that we get a bias towards a diagonal $W_k$ after one update (E.23). Let $W_k = C' + (\rho - c')I$, where $C' = c'11^T$ and $\rho > c'$, with $c'$ denoting the off-diagonal term and $\rho$ the on-diagonal one. As in the first step, we can tune the margin parameter of the loss, $\Delta$ to be

large enough, so that we operate on the linear part of the loss. We have for the gradient of $v$:

$$\begin{aligned}
\mathbb{E}\left[\frac{\partial f(e)_{p,i}}{\partial v_j}\right] &= \mathbb{E}\left[\sum_{t'=1}^{p} 1\left\{x_{t'}=i\right\} \sum_{s=1}^{t'} \delta_{j(t'-s+1)} \left(c' + (\rho - c') e_{x_p}^T e_{x_s}\right) 1\left\{j \le p\right\}\right] \\
&= \mathbb{E}\left[\sum_{t'=1}^{p} 1\left\{x_{t'}=i\right\} \left(c' + \left(\rho - c'\right) e_{x_p}^T e_{x_{t'-j+1}}\right) 1\left\{j \le t'\right\} 1\left\{j \le p\right\}\right] \\
&= \mathbb{E}\left[\sum_{t'=j}^{p} 1\left\{x_{t'}=i\right\} \left(c' + \left(\rho - c'\right) 1\left\{x_{t'-j+1} = x_p\right\}\right) 1\left\{j \le p\right\}\right] \\
&= c'\pi_i \left(p - j + 1\right) 1\left\{j \le p\right\} + \left(\rho - c'\right) \sum_{t'=j}^{p} \mathbb{P}\left[x_{t'} = i, x_{t'-j+1} = x_p\right] 1\left\{j \le p\right\}.
\end{aligned} \tag{E.30}$$

We split the cases of $j = 1$ and $j \ge 2$. For $j = 1$:

$$\begin{aligned}
\mathbb{E}\left[\frac{\partial f(e)_{p,i}}{\partial v_1}\right] &= c'\pi_i p + \left(\rho - c'\right) \sum_{t'=1}^{p} \mathbb{P}\left[x_{t'} = i = x_p\right] \\
&= c'\pi_i p + \left(\rho - c'\right) \sum_{t'=1}^{p} \left(\mathcal{P}^{p-t'}\right)_{ii} \pi_i \\
&= c'\pi_i p + \frac{\left(\rho - c'\right)\pi_i}{\lambda - 1} \sum_{t'=1}^{p} (\beta_{ii}\lambda^{p-t'} + \gamma_{ii}) \\
&= c'\pi_i p + \frac{\left(\rho - c'\right)\pi_i}{\lambda - 1} \left(p\gamma_{ii} + \beta_{ii}\frac{\lambda^p - 1}{\lambda - 1}\right).
\end{aligned} \tag{E.31}$$

$$
\begin{aligned}
\mathbb{E}\left[\frac{\partial f(e)_{p,i}}{\partial v_1}\right] &= c'\pi_i\,(p-j+1)\,1\{j\le p\} + (\rho-c')\sum_{t'=j}^{p}\sum_{l=1}^{2}\mathbb{P}\left[x_{t'}=i, x_{t'-j+1}=x_p=l\right]1\{j\le p\} \\
&= c'\pi_i\,(p-j+1)\,1\{j\le p\} + (\rho-c')\sum_{l=1}^{2}\sum_{t'=j}^{p}\mathbb{P}\left[x_p=l\mid x_{t'}=i\right]\mathbb{P}\left[x_{t'}=i\mid x_{t'-j+1}=l\right]\mathbb{P}\left[x_{t'-j+1}=l\right]1 \\
&= c'\pi_i\,(p-j+1)\,1\{j\le p\} + (\rho-c')\sum_{l=1}^{2}\pi_l\sum_{t'=j}^{p}\frac{\beta_{il}\lambda^{p-t'}+\gamma_{il}}{\lambda-1}\frac{\beta_{li}\lambda^{j-1}+\gamma_{li}}{\lambda-1}1\{j\le p\} \\
&= \left(c'\pi_i\,(p-j+1) + (\rho-c')\sum_{l=1}^{2}\frac{\pi_l}{(\lambda-1)^2}\sum_{t'=j}^{p}\left(\beta_{il}\beta_{li}\lambda^{p-t'+j-1}+\beta_{il}\gamma_{li}\lambda^{p-t'}+\gamma_{il}\beta_{li}\lambda^{j-1}+\gamma_{il}\gamma_{li}\right)\right)1\{j\le \\
&= c'\pi_i\,(p-j+1)\,1\{j\le p\} \\
&\quad + \frac{(\rho-c')}{(\lambda-1)^2}\sum_{l=1}^{2}\pi_l\left(\beta_{il}\beta_{li}\frac{\lambda^p-\lambda^{j-1}}{\lambda-1}+\beta_{il}\gamma_{li}\frac{\lambda^{p-j+1}-1}{\lambda-1}+(p-j+1)\left(\gamma_{il}\beta_{li}\lambda^{j-1}+\gamma_{il}\gamma_{li}\right)\right)1\{j\le p\}\,.
\end{aligned}
\tag{E.32}
$$

and, similarly, for the derivative of the ground truth logit:

$$
\mathbb{E}\left[\frac{\partial f(e)_{p,x_{p+1}}}{\partial v_j}\right] = c'\sum_{t'=j}^{p}\mathbb{P}\left[x_{t'}=x_{p+1}\right]1\{j\le p\} + (\rho-c')\sum_{t'=j}^{p}\mathbb{P}\left[x_{t'}=x_{p+1}, x_{t'-j+1}=x_p\right]1\{j\le p\}\,.
\tag{E.33}
$$

The first term corresponds to the gradient of the first step and can be found in eq. (E.27). For the second term, we have for $j=1$:

$$
\begin{aligned}
\sum_{t'=1}^{p}\mathbb{P}\left[x_{t'}=x_{p+1}=x_p\right] &= \sum_{i=1}^{2}\sum_{t'=1}^{p}\mathbb{P}\left[x_{p+1}=i\mid x_p=i\right]\mathbb{P}\left[x_p=i\mid x_{t'}=i\right]\mathbb{P}\left[x_{t'}=i\right] \\
&= \sum_{i=1}^{2}\frac{\pi_i\mathcal{P}_{ii}}{\lambda-1}\left(\beta_{ii}\frac{\lambda^p-1}{\lambda-1}+\gamma_{ii}p\right),
\end{aligned}
\tag{E.34}
$$

and for $j \geq 2$:

$$\sum_{t'=j}^{p} \mathbb{P}\left[x_{t'} = x_{p+1}, x_{t'-j+1} = x_p\right] = \sum_{i=1}^{2}\sum_{l=1}^{2}\sum_{t'=j}^{p} \mathcal{P}_{li}\mathbb{P}\left[x_p = l \mid x_{t'} = i\right]\mathbb{P}\left[x_{t'} = i \mid x_{t'-j+1} = l\right]\mathbb{P}\left[x_{t'-j+1} = l\right], \tag{E.35}$$

which, by using the calculations of eq. (E.32), amounts to:

$$\frac{1}{(\lambda-1)^2}\sum_{i=1}^{2}\sum_{l=1}^{2}\pi_l\mathcal{P}_{li}\left(\beta_{il}\beta_{li}\frac{\lambda^p-\lambda^{j-1}}{\lambda-1} + \beta_{il}\gamma_{li}\frac{\lambda^{p-j+1}-1}{\lambda-1} + (p-j+1)\left(\gamma_{il}\beta_{li}\lambda^{j-1} + \gamma_{il}\gamma_{li}\right)\right)1\left\{j \leq p\right\}. \tag{E.36}$$

Thus, the updates will have a gradient contribution that is the same as the first step, $\frac{\partial \mathcal{L}^0}{\partial v_j}$, and another one that comes from the diagonal bias of $W_k$.

For $j = 1$:

$$\begin{aligned}
v_1^{(2)} &= v_1^{(1)} - \eta\frac{\partial \mathcal{L}}{\partial v_1} \\
&= v_1^{(1)} - \eta c'\frac{\partial \mathcal{L}^0}{\partial v_1} + \frac{\eta\left(\rho - c'\right)}{t}\sum_{p=1}^{t}\sum_{i=1}^{2}\mathbb{E}_{a,b}\left[\left(\mathcal{P}_{ii} - \frac{1}{2}\right)\frac{\pi_i}{\lambda-1}\left(\beta_{ii}\frac{\lambda^p-1}{\lambda-1} + \gamma_{ii}p\right)\right] \\
&= v_1^{(1)} - \eta c'\frac{\partial \mathcal{L}^0}{\partial v_1} + \frac{\eta\left(\rho - c'\right)}{t}\mathbb{E}_{a,b}\left[\left(\frac{\lambda-\lambda^{t+1}}{1-\lambda} - t\right)\frac{(a-1)(b-1)}{(\lambda-1)^2} + \left(\frac{(b-1)^2(a-\frac{1}{2}) + (a-1)^2(b-\frac{1}{2})}{(\lambda-1)^2}\right)\frac{t(t-}{2}\right.
\end{aligned} \tag{E.37}$$

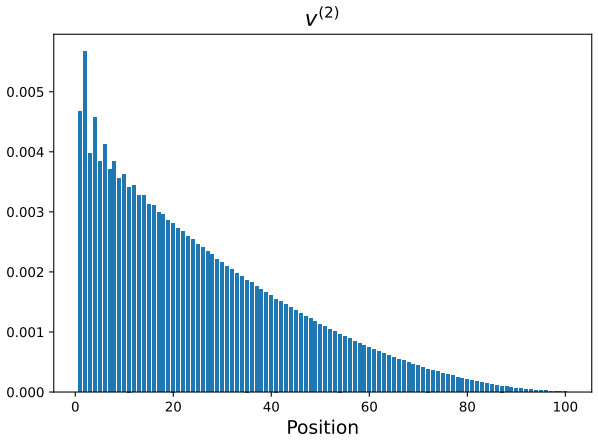



**Figure E.2:** Empirical estimation of the 2nd step gradient of $v$ that comes from the diagonal bias of $W_k$ - see eqs. (E.37),(E.38). Hyperparameters: initialization $c = 0.02$, learning rate $\eta = 0.03$, sequence length $t = 100$. The outer expectations are approximated using 10,000 samples.

For $j \geq 2$:

$$\begin{aligned} v_j^{(2)} &= v_j^{(1)} - \eta \frac{\partial \mathcal{L}}{\partial v_j} \\ &= v_1^{(1)} - \eta c' \frac{\partial \mathcal{L}^0}{\partial v_j} + \frac{\eta(\rho - c')}{t} \sum_{i=1}^{2} \sum_{l=1}^{2} \mathbb{E}_{a,b} \Bigg[ \frac{\pi_l}{(\lambda-1)^2} \left( \mathcal{P}_{li} - \frac{1}{2} \right) \Bigg[ \frac{\beta_{il}\beta_{li}}{\lambda - 1} \left( \frac{\lambda^j - \lambda^{t+1}}{1-\lambda} - (t-j+1)\lambda^{j-1} \right) \\ &\qquad + \frac{\beta_{il}\gamma_{li}}{\lambda - 1} \left( \frac{\lambda - \lambda^{t-j+2}}{\lambda - 1} - t + j - 1 \right) \\ &\qquad + \frac{(t-j+1)(t-j+2)}{2} \left( \gamma_{il}\beta_{li}\lambda^{j-1} + \gamma_{il}\gamma_{li} \right) \Bigg] \Bigg]. \end{aligned} \tag{E.38}$$

See Figure E.2 for an empirical estimation of the diagonal contribution of the gradient for some choice of hyperparameters. In leading order terms, the updates simplify to:

For $j = 1$:

$$v_1^{(2)} = v_1^{(1)} - \eta c' \frac{\partial \mathcal{L}^0}{\partial v_1} + \frac{\eta\,(\rho - c')}{t} \mathbb{E}_{a,b}\left[\frac{(b-1)^2(a-\frac{1}{2}) + (a-1)^2(b-\frac{1}{2})}{(\lambda-1)^2}\right] \frac{t(t+1)}{2} + O(1). \tag{E.39}$$

For $j \geq 2$:

$$v_j^{(2)} = v_j^{(1)} - \eta c' \frac{\partial \mathcal{L}^0}{\partial v_j} + \frac{\eta\,(\rho - c')}{t} \frac{(t-j+1)(t-j+2)}{2} \mathbb{E}_{a,b}\Bigg[2\frac{(a-1)(b-1)\left((b-1)\left(a-\frac{1}{2}\right) + (a-1)\left(b-\frac{1}{2}\right)\right)}{(\lambda-1)^3}\lambda^{j-1} + \frac{(b-a)\left((b-1)^2\left(a-\frac{1}{2}\right) - (a-1)^2\left(b-\frac{1}{2}\right)\right)}{(\lambda-1)^3}\Bigg] + O(1) \tag{E.40}$$

where for all $j$ it is:

$$v_j^{(1)} - \eta c' \frac{\partial \mathcal{L}^0}{\partial v_j} = c + \frac{(c+c')\eta}{t} \frac{(t-j+1)(t-j+2)}{2} \mathbb{E}_{a,b}\left[\frac{(a-1)^2 + (b-1)^2}{(\lambda-1)^2} - \frac{1}{2}\right] + O(1). \tag{E.41}$$

We now show that $v_2^{(2)} > v_1^{(2)}$ in the large $t$ regime. Observe that the term of (E.41) is the same for $j = 1, 2$. We evaluate the expectation of eq. (E.39) that corresponds to $j = 1$:

$$\mathbb{E}_{a,b}\left[\frac{(b-1)^2(a-\frac{1}{2}) + (a-1)^2(b-\frac{1}{2})}{(\lambda-1)^2}\right] = \frac{1-\ln 2}{3} \tag{E.42}$$

and the expectation of (E.40), leveraging symmetry of $a, b$ in the expression, that corresponds to $j = 2$:

$$4\mathbb{E}_{a,b}\left[\frac{(a-1)(b-1)^2(a-\frac{1}{2})}{(a+b-2)^3}(a+b-1)\right] + 2\mathbb{E}_{a,b}\left[\frac{(b-a)(b-1)^2(a-\frac{1}{2})}{(a+b-2)^3}\right] = \frac{7-10\ln 2}{2} + \frac{5-8\ln 2}{6}. \tag{E.43}$$

The latter term is larger, so, when $t \to \infty$, it is $v_2^{(2)} > v_1^{(2)}$. Finally, observe that the expectation $\mathbb{E}_{a,b}\left[\frac{(a-1)(b-1)^2(a-\frac{1}{2})}{(a+b-2)^3}(a+b-1)^{j-1}\right]$ is negative for $j$ odd and decreasing otherwise, which shows

that $v_2^{(2)} > v_j^{(2)}, j > 2$. So, we showed that the 2nd coordinate grows larger than the rest after the 2nd step.

□

With this calculation at hand, we can ask questions about the optimization trajectory and how that would change if we alter the data distribution. In particular, in the following Corollary, we answer the question of how the gradient would change when $a = b$, i.e., when the transition matrix is

$$\mathcal{P} = \begin{pmatrix} a & 1-a \\ 1-a & a \end{pmatrix}, \tag{E.44}$$

with $a \sim Unif([0,1])$. Notice that in the case there is not a unigrams solution (or in other words the unigrams solution is as good as the uniformly at random solution).

**Corollary E.2.** *When $a = b$, it is*

$$\begin{aligned} W_k^{(1)} &= \begin{pmatrix} c & c \\ c & c \end{pmatrix} + c\eta \left[ O(t) \begin{pmatrix} \infty & 0 \\ 0 & \infty \end{pmatrix} + O(1) \right], \\ v_j^{(1)} &= c - \frac{c\eta}{2t}(t - j + 1) + O(1/t), j \in [t]. \end{aligned} \tag{E.45}$$

*Proof.* We first compute the gradient with respect to $W_k$, by setting $a = b$ in eq. (E.19),(E.20). For $m, l = 1, 1$, we have:

$$\left(W_k^{(1)}\right)_{1,1} = c + \frac{\eta c}{t} \mathbb{E}_{a \sim Unif([0,1])} \left[ \frac{(a-1)(a-\frac{1}{2})}{(2a-2)^3} \left[ \frac{2(a-1)^2}{(2a-2)^2}(II) + \frac{2(a-1)^2}{2a-2}(III) \right] \right], \tag{E.46}$$

where $(II)$, $(III)$ are defined in eq. (E.23). Notice that the $(IV)$ with the cubic, $O(t^3)$, dependence disappeared, so the leading order terms are $O(t^2)$. By substituting the leading order terms from

$(II)$, $(III)$, we get:

$$\begin{aligned}\left(W_k^{(1)}\right)_{1,1} &= c + \frac{\eta c}{t}\mathbb{E}_{a\sim Unif([0,1])}\left[\frac{(a-1)(a-\frac{1}{2})}{(2a-2)^3}\left[\frac{2(a-1)^2}{(2a-2)^2}\frac{2-2a}{2} - \frac{2(a-1)^2}{2a-2}\frac{1}{2}\right]\right] O(t^2) \\ &= c + \eta c\mathbb{E}_{a\sim Unif([0,1])}\left[-\frac{(a-\frac{1}{2})}{8(a-1)}\right] O(t) \to \infty.\end{aligned} \tag{E.47}$$

For $m, l = 1, 2$, the $O(t^3)$ terms similarly disappear, and we have:

$$\begin{aligned}\left(W_k^{(1)}\right)_{1,2} &= c + \frac{\eta c}{t}\mathbb{E}_{a\sim Unif([0,1])}\left[\frac{(a-1)(a-\frac{1}{2})}{(2a-2)^3}\left[\frac{2(a-1)^2}{(2a-2)^2}(II) - \frac{2(a-1)^2}{2a-2}(III)\right]\right] \\ &= c + \frac{\eta c}{t}\mathbb{E}_{a\sim Unif([0,1])}\left[\frac{(a-1)(a-\frac{1}{2})}{(2a-2)^3}\left[\frac{2(a-1)^2}{(2a-2)^2}\frac{2-2a}{2} + \frac{2(a-1)^2}{2a-2}\frac{1}{2}\right]\right] O(t^2) \\ &= c.\end{aligned} \tag{E.48}$$

Now, for the $v$ vector, we observe that when we set $a = b$ in eq. (E.28), the leading term disappears and its expression simplifies to:

$$\begin{aligned}v_j^{(1)} &= c - \frac{c\eta}{t}\mathbb{E}_{a\sim Unif([0,1])}\left[2(2a-2)\frac{(a-1)^2}{(2a-2)^3}(t-j+1)\right] + O(1/t) \\ &= c - \frac{c\eta}{2t}(t-j+1) + O(1/t).\end{aligned} \tag{E.49}$$

□

The previous calculations reveal that there is less "signal" towards the unigrams solution when $a = b$.

Finally, we provide our main technical result that demonstrates that two steps of stochastic gradient descent lead the model to the bigrams solution.

**Proposition E.3.** *Suppose we initialize the model of eq. (E.2) with* $W_k^{(0)} = w_{init} 1 1^T$ *and* $v = v_{init} 1^T$. *Consider a two step procedure, where we first train with gradient descent on the original distribution* $a \sim \text{Unif}(0,1), b \sim \text{Unif}(0,1)$, *and then at the second step with* $a = b \sim \text{Unif}([\frac{1}{2} - \varepsilon, \frac{1}{2} + \varepsilon])$, $\varepsilon \in (0, 1/2)$. *Let* $\eta_1 = O\left(\frac{1}{t^2}\right)$ *and* $\eta_2 = O\left(\frac{1}{t}\right)$ *be the learning rates for the 1st and 2nd step of gradient descent, and let* $\lambda_{wd} = 1$ *be the weight decay coefficient used in the second step. Then, after 2 steps of gradient descent, for* $t \to \infty$, *it is:*

$$f(e)_{p,i} = O(\varepsilon^4) \sum_{t'=1}^{p} 1\{x_{t'} = i\} 1\{x_p = x_{t'-1}\} + O(\varepsilon^6). \tag{E.50}$$

*Proof.* For the first step, we have from Lemma 1:

$$W_k^{(1)} = w_{init} 1 1^T + v_{init} \eta_1 \begin{pmatrix} B & A \\ A & B \end{pmatrix} O(t^2) + v_{init} \eta_1 O(t). \tag{E.51}$$

For $\eta_1 = O\left(\frac{1}{t^2}\right)$, then

$$W_k^{(1)} = w_{init} 1 1^T + \begin{pmatrix} B & A \\ A & B \end{pmatrix} O(1) + O(1/t). \tag{E.52}$$

Recall, from Lemma 1, that we have $v_j^{(1)} = v_{init} + \eta_1 w_{init} O(t)$ for all $j \in [t]$, so, for $\eta_1 = O\left(\frac{1}{t^2}\right)$, it will be $v_j^{(1)} = v_{init}$ for all $j \in [t]$.

Assume now that at the second step we train with weight decay $\lambda_{wd} = 1$ and the data distribution changes to $a = b \sim \text{Unif}([\frac{1}{2} - \varepsilon, \frac{1}{2} + \varepsilon])$. Then, from eq. (E.47)* the update of the 2nd layer would

*That equation refers to the first step of gradient descent for $a = b$, but notice that since the model is linear in $W_k$, and in this case $v_j^{(1)} = v_{init}$, it characterizes the 2nd step too.

be:

$$\begin{aligned} W_k^{(2)} &= (1-\lambda_{wd})W_k^{(1)} - \eta_2 \nabla \mathcal{L} \\ &= v_{init}\eta_2 \begin{pmatrix} \int_{\frac{1}{2}-\varepsilon}^{\frac{1}{2}+\varepsilon} \frac{a-\frac{1}{2}}{(a-1)} & 0 \\ 0 & \int_{\frac{1}{2}-\varepsilon}^{\frac{1}{2}+\varepsilon} \frac{a-\frac{1}{2}}{(a-1)} \end{pmatrix} O(t) + O(1) \\ &= v_{init}\eta_2 \begin{pmatrix} 2\varepsilon + \frac{1}{2}\ln\left(\frac{\frac{1}{2}-\varepsilon}{\frac{1}{2}+\varepsilon}\right) & 0 \\ 0 & 2\varepsilon + \frac{1}{2}\ln\left(\frac{\frac{1}{2}-\varepsilon}{\frac{1}{2}+\varepsilon}\right) \end{pmatrix} O(t) + O(1). \end{aligned} \tag{E.53}$$

For the update of $v$ we have:

$$\begin{aligned} v_j^{(2)} &= (1-\lambda_{wd})v_j^{(1)} - \frac{\partial \mathcal{L}}{\partial v_j} \\ &= \eta_2 A \frac{\partial \mathcal{L}_{const}}{\partial v_j} + \eta_2 3A \frac{\partial \mathcal{L}_{diag}}{\partial v_j} \\ &= A\frac{\eta_2 \rho}{t}\left(\frac{(t-j+1)(t-j+2)}{2}\left(\frac{(a-1)^2+(b-1)^2}{(\lambda-1)^2} - \frac{1}{2}\right) + O(t)\right) + 3A\eta_2 \frac{\partial \mathcal{L}_{diag}}{\partial v_j}. \end{aligned} \tag{E.54}$$

For $a = b$, it is only the $O(t)$ part inside the parenthesis that survives from the second term (see eq. (E.49)) and by substituting the diagonal gradient from eqs. (E.39),(E.40) when $a = b$, we get for all $j \in [t]$:

$$v_j = \eta_2 \rho A O(1) + 3A\frac{\eta_2}{t}\frac{(t-j+1)(t-j+2)}{2} 2\mathbb{E}_{a\sim Unif([1/2-\varepsilon,1/2+\varepsilon])}\left[\frac{a-1/2}{4}(2a-1)^{j-1}\right] + 3A\frac{\eta_2}{t}O(t). \tag{E.55}$$

We calculate the expectation:

$$\mathbb{E}_{a\sim Unif([1/2-\varepsilon,1/2+\varepsilon])}\left[\frac{a-1/2}{4}(2a-1)^{j-1}\right] = 0, \tag{E.56}$$

when $j = 1$. For $j \geq 2$:

$$\begin{aligned}\mathbb{E}_{a\sim Unif([1/2-\varepsilon,1/2+\varepsilon])}\left[(a-1/2)(2a-1)^{j-1}\right] &= \int_{1/2-\varepsilon}^{1/2+\varepsilon}(a-1/2)(2a-1)^{j-1} \\ &= \frac{1}{2j}\left[(2a-1)^j(a-\frac{1}{2})\right]_{1/2-\varepsilon}^{1/2+\varepsilon} - \frac{1}{2j}\int_{1/2-\varepsilon}^{1/2+\varepsilon}(2a-1)^j da \\ &= \frac{1}{2j}\varepsilon\left((2\varepsilon)^j + (-2\varepsilon)^j\right) - \frac{1}{4j(j+1)}\left((2\varepsilon)^{j+1} - (-2\varepsilon)^{j+1}\right).\end{aligned} \tag{E.57}$$

For $j = 2k+1$, this equals to 0. For $j = 2k$, it is equal to $\frac{4^k}{2k+1}\varepsilon^{2k+1}$.

Thus, by setting $\eta_2 = O\left(\frac{1}{t}\right)$, we get:

$$W_k^{(2)} = O(\varepsilon)\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix} + O\left(\frac{1}{t}\right). \tag{E.58}$$

$$v_j = v_{init} + O\left(\frac{(t-j+1)(t-j+2)}{t^2}\right)\sum_{s=1,s=2k}^{t}\delta(s-j)\varepsilon^{j+1} + O\left(\frac{1}{t}\right), \tag{E.59}$$

where $\delta$ is Dirac's delta function.

Thus, after 2 steps of gradient descent, as $t \to \infty$, the prediction of the model will be

$$f(e)_{p,i} = O(\varepsilon^4)\sum_{t'=1}^{p} 1\left\{x_{t'} = i\right\} 1\left\{x_p = x_{t'-1}\right\} + O(\varepsilon^6). \tag{E.60}$$

□

An interesting observation that stems from the proof is that the learning rate of the first step is $O(1/t^2)$, where recall $t$ is the sequence length, while for the second step the learning rate is $O(1/t)$ (much larger). The first step in our proof corresponds to the learning of the second layer $W_k$, while the second step "cleans up" $W_k$ and learns the first layer $v$. Interestingly, this is what we also observe

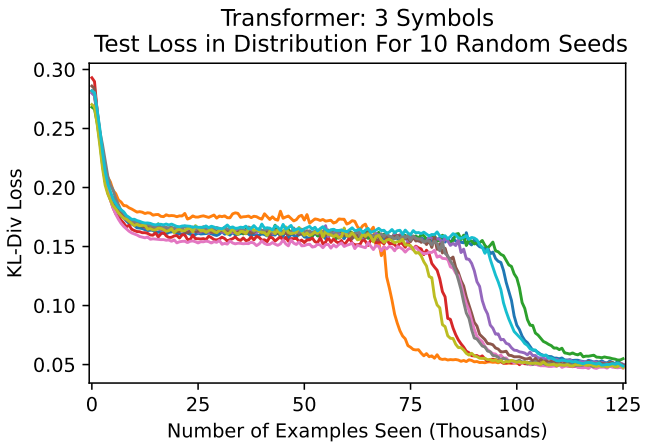



**Figure E.3:** In distribution test loss for 10 two layer attention only transformers, with random seeds $0, 1, \ldots 9$ (randomness affects initialization and the training data). The training dynamics are consistent for each model, though the exact position of the phase transitions.

in the experiments with the transformers and this minimal model: the first drop is immediate for the learning of the 2nd layer, while the second loss drop happens after a long plateau and corresponds to the "grokking" of the positional embeddings.

## E.2 Experimental Details

We train transformers of the form (8.4) with the AdamW optimizer with learning rate $3e-5$, batch size 64, and hidden dimension 16. The sequence length of the examples is 100 tokens. The minimal model was trained with SGD, with batch size 64, and learning rate $2e-3$. For 3-grams, a learning rate of $3e-2$ was used. We use PyTorch 2.1.2. Some of the training and model code was based on minGPT Karpathy (2023).

The experiments all measure the outputs of the models at the last token.

## E.3 Additional Experiments

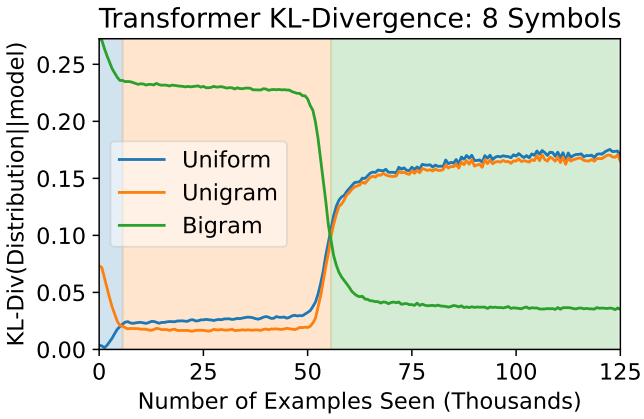



**Figure E.4:** Our results extend to more symbols than $k = 2$ or $k = 3$. The KL-divergence between the transformer and strategies over training. This required a sequence length greater than $100$ ($200$ in this case) for the difference between unigrams and bigrams to be large enough for the unigram phase to be visible (in either case there was a plateau before the final drop in test loss).

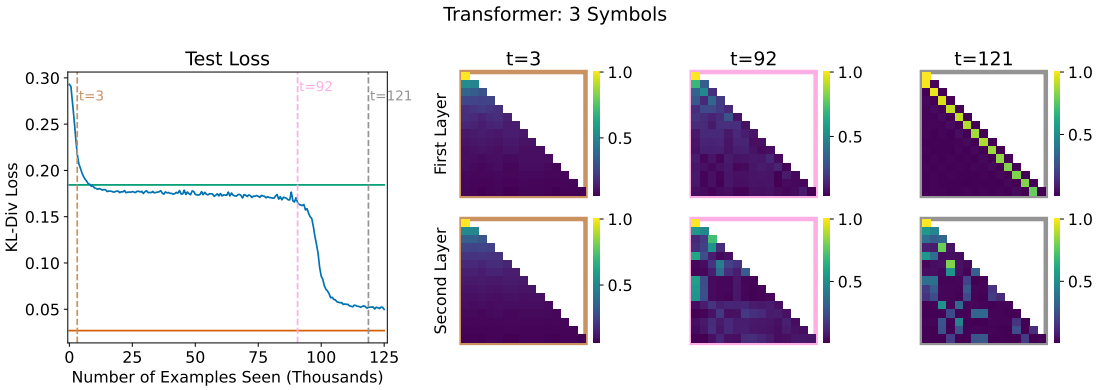



**Figure E.5:** A two layer attention-only transformer was trained with cross entropy loss on ICL-MC. The heatmaps on the right represent part of the attention for the transformer at various time steps, specifically the value of $A$ from (8.5). The top row are showing $A$ from the first layer, and the bottom row from the second layer.

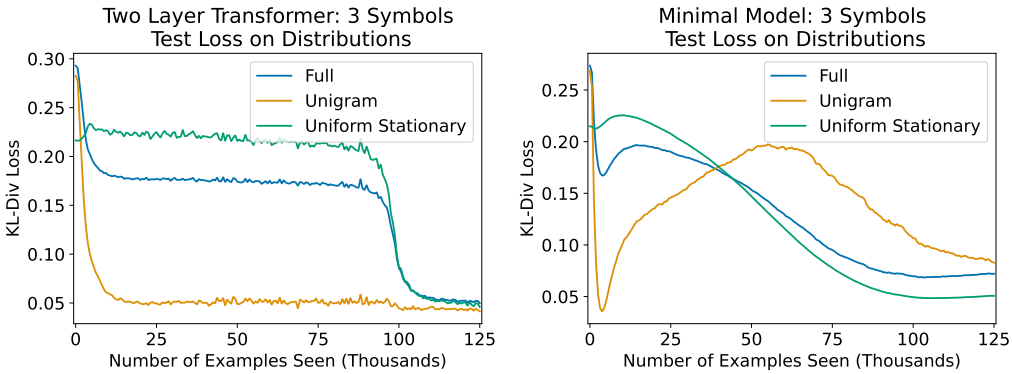



**Figure E.6:** A two layer attention-only transformer (top) and minimal model (8.8) (bottom), trained on the main task with ICL-MC with cross entropy loss, test loss measured by KL-Divergence from the underlying truth (labels based on transition probabilities, not samples). The distributions test loss is measured in are (from left to right) in-distribution, a distribution where each token is sampled iid, and a distribution over uniformly random doubly stochastic transition matrices (equivalently, stationary distribution is identity, or unigram based guesses are as good as guessing uniform probability). For both models, the in distribution test loss quickly drops to the level of the unigram algorithm.

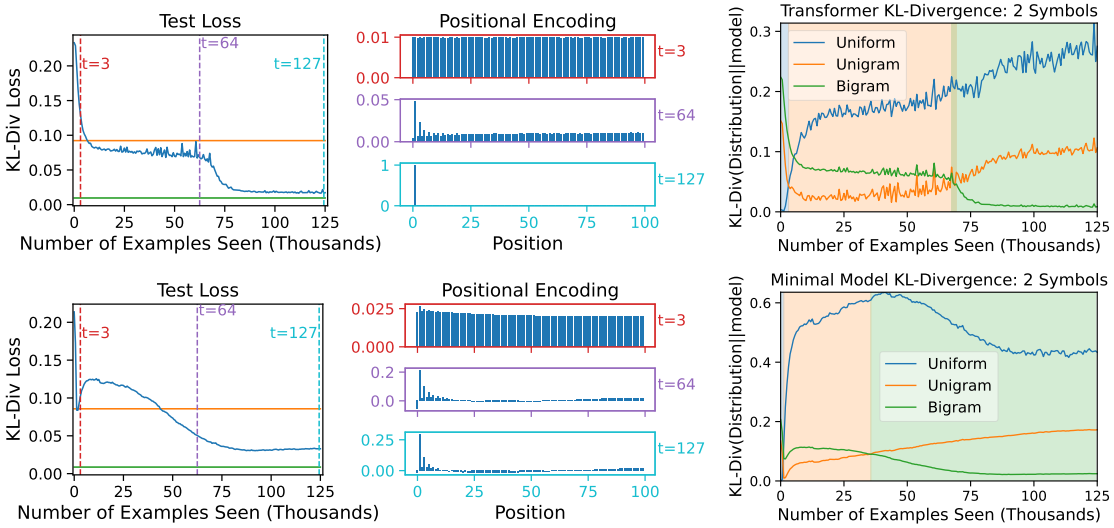



**Figure E.7:** A comparison of the two layer attention only transformer and minimal model for $k = 2$ symbols. Note the alternating pattern in the positional encodings at 64,000 examples seen.

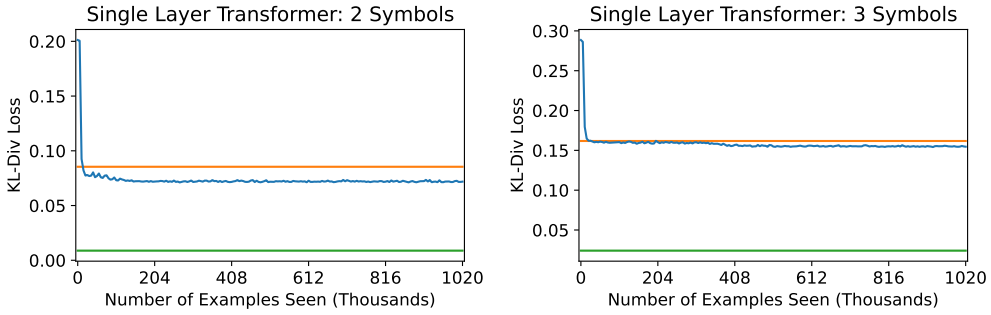



**Figure E.8:** Graphs of test loss showing that a single layer transformer can not achieve good performance on ICL-MC. This result holds for transformers with or without MLPs, and with absolute or relative positional encodings. These graphs show that even trained 8 times longer, there is no notable increase in performance beyond the unigrams strategy (orange line).

# References


Abbe, E., Adsera, E. B., & Misiakiewicz, T. (2022a). The merged-staircase property: a necessary and nearly sufficient condition for sgd learning of sparse functions on two-layer neural networks. In *Conference on Learning Theory* (pp. 4782–4887).: PMLR.

Abbe, E., Adsera, E. B., & Misiakiewicz, T. (2023). Sgd learning on neural networks: leap complexity and saddle-to-saddle dynamics. In *The Thirty Sixth Annual Conference on Learning Theory* (pp. 2552–2623).: PMLR.

Abbe, E., Cornacchia, E., Hązła, J., & Marquis, C. (2022b). An initial alignment between neural network and target is needed for gradient descent to learn. *arXiv preprint arXiv:2202.12846*.

Abbe, E., Kamath, P., Malach, E., Sandon, C., & Srebro, N. (2021). On the power of differentiable learning versus PAC and SQ learning. *Advances in Neural Information Processing Systems*, 34.

Abbe, E. & Sandon, C. (2020). Poly-time universality and limitations of deep learning. *arXiv preprint arXiv:2001.02992*.

Abernethy, J. D., Agarwal, A., Marinov, T. V., & Warmuth, M. K. (2023). A mechanism for sample-efficient in-context learning for sparse retrieval tasks. *CoRR*, abs/2305.17040.

Agarwal, N., Anil, R., Hazan, E., Koren, T., & Zhang, C. (2020). Disentangling adaptive gradient methods from learning rates. *arXiv preprint arXiv:2002.11803*.

Akyürek, E., Schuurmans, D., Andreas, J., Ma, T., & Zhou, D. (2022). What learning algorithm is in-context learning? investigations with linear models. *arXiv preprint arXiv:2211.15661*.

Akyürek, E., Wang, B., Kim, Y., & Andreas, J. (2024). In-context language learning: Architectures and algorithms. *CoRR*, abs/2401.12973.

Akyürek, E., Wang, B., Kim, Y., & Andreas, J. (2024). In-context language learning: Arhitectures and algorithms. *arXiv preprint arXiv:2401.12973*.

Alekhnovich, M. (2003). More on average case vs approximation complexity. In *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.* (pp. 298–307).: IEEE.

Allen-Zhu, Z. & Li, Y. (2019). What can ResNet learn efficiently, going beyond kernels? *Advances in Neural Information Processing Systems*, 32.

Allen-Zhu, Z., Li, Y., & Song, Z. (2019). A convergence theory for deep learning via over-parameterization. In *International Conference on Machine Learning* (pp. 242–252).: PMLR.

Andoni, A., Panigrahy, R., Valiant, G., & Zhang, L. (2014). Learning polynomials with neural networks. In *International Conference on Machine Learning* (pp. 1908–1916).: PMLR.

Anil, C., Wu, Y., Andreassen, A., Lewkowycz, A., Misra, V., Ramasesh, V., Slone, A., Gur-Ari, G., Dyer, E., & Neyshabur, B. (2022). Exploring length generalization in large language models. *Advances in Neural Information Processing Systems*, 35, 38546–38556.

Anthony, M. & Bartlett, P. L. (1999). *Neural network learning: Theoretical foundations*, volume 9. cambridge university press Cambridge.

Applebaum, B., Barak, B., & Wigderson, A. (2010). Public-key cryptography from different assumptions. In *Proceedings of the forty-second ACM symposium on Theory of computing* (pp. 171–180).

Applebaum, B., Cash, D., Peikert, C., & Sahai, A. (2009). Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In *Annual International Cryptology Conference* (pp. 595–618).: Springer.

Arora, S. & Barak, B. (2009). *Computational complexity: a modern approach*. Cambridge University Press.

Arous, G. B., Gheissari, R., & Jagannath, A. (2021). Online stochastic gradient descent on non-convex losses from high-dimensional inference. *J. Mach. Learn. Res.*, 22, 106–1.

Arpit, D., Jastrzebski, S., Ballas, N., Krueger, D., Bengio, E., Kanwal, M. S., Maharaj, T., Fischer, A., Courville, A., Bengio, Y., et al. (2017). A closer look at memorization in deep networks. In *International conference on machine learning* (pp. 233–242).: PMLR.

Ba, J., Erdogdu, M. A., Suzuki, T., Wang, Z., Wu, D., & Yang, G. (2022). High-dimensional asymptotics of feature learning: How one gradient step improves the representation. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, & A. Oh (Eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

Bachmann, G., Anagnostidis, S., & Hofmann, T. (2023). Scaling mlps: A tale of inductive bias. *arXiv preprint arXiv:2306.13575*.

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Bahri, Y., Dyer, E., Kaplan, J., Lee, J., & Sharma, U. (2021). Explaining neural scaling laws. *arXiv preprint arXiv:2102.06701*.

Barak, B., Edelman, B., Goel, S., Kakade, S., Malach, E., & Zhang, C. (2022). Hidden progress in deep learning: Sgd learns parities near the computational limit. *Advances in Neural Information Processing Systems*, 35, 21750–21764.

Bartlett, P. (1996). For valid generalization the size of the weights is more important than the size of the network. *Advances in neural information processing systems*, 9.

Bartlett, P. L., Foster, D. J., & Telgarsky, M. J. (2017). Spectrally-normalized margin bounds for neural networks. *Advances in neural information processing systems*, 30.

Bartlett, P. L. & Mendelson, S. (2002). Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3, 463–482.

Baur, W. & Strassen, V. (1983). The complexity of partial derivatives. *Theoretical computer science*, 22(3), 317–330.

Baxter, R. J. (2016). *Exactly solved models in statistical mechanics*. Elsevier.

Belilovsky, E., Eickenberg, M., & Oyallon, E. (2019). Greedy layerwise learning can scale to imagenet. In *International conference on machine learning* (pp. 583–593).: PMLR.

Belkin, M., Hsu, D., Ma, S., & Mandal, S. (2019). Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32), 15849–15854.

Belkin, M., Ma, S., & Mandal, S. (2018). To understand deep learning we need to understand kernel learning. In *International Conference on Machine Learning* (pp. 541–549).: PMLR.

Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8), 1798–1828.

Bengio, Y., Louradour, J., Collobert, R., & Weston, J. (2009). Curriculum learning. In A. P. Danyluk, L. Bottou, & M. L. Littman (Eds.), *Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, Montreal, Quebec, Canada, June 14-18, 2009*, volume 382 of *ACM International Conference Proceeding Series* (pp. 41–48).: ACM.

Berry, A. C. (1941). The accuracy of the gaussian approximation to the sum of independent variates. *Transactions of the American Mathematical Society*, 49(1), 122–136.

Bhattamishra, S., Ahuja, K., & Goyal, N. (2020a). On the ability and limitations of transformers to recognize formal languages. *arXiv preprint arXiv:2009.11264*.

Bhattamishra, S., Patel, A., & Goyal, N. (2020b). On the computational power of transformers and its implications in sequence modeling. *arXiv preprint arXiv:2006.09286*.

Bietti, A., Bruna, J., Sanford, C., & Song, M. J. (2022). Learning single-index models with shallow neural networks. *Advances in Neural Information Processing Systems*, 35, 9768–9783.

Bietti, A., Cabannes, V., Bouchacourt, D., Jegou, H., & Bottou, L. (2023). Birth of a transformer: A memory viewpoint.

Blum, A., Furst, M., Jackson, J., Kearns, M., Mansour, Y., & Rudich, S. (1994). Weakly learning dnf and characterizing statistical query learning using fourier analysis. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing* (pp. 253–262).

Blum, A., Kalai, A., & Wasserman, H. (2003). Noise-tolerant learning, the parity problem, and the statistical query model. *Journal of the ACM (JACM)*, 50(4), 506–519.

Blum, A. L. & Langley, P. (1997). Selection of relevant features and examples in machine learning. *Artificial intelligence*, 97(1-2), 245–271.

Bogdanov, A., Sabin, M., & Vasudevan, P. N. (2019). Xor codes and sparse learning parity with noise. *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, (pp. 986–1004).

Bogdanov, A., Trevisan, L., et al. (2006). Average-case complexity. *Foundations and Trends® in Theoretical Computer Science*, 2(1), 1–106.

Bordelon, B., Atanasov, A., & Pehlevan, C. (2024). A dynamical model of neural scaling laws. *arXiv preprint arXiv:2402.01092*.

Bordelon, B., Canatar, A., & Pehlevan, C. (2020). Spectrum dependent learning curves in kernel regression and wide neural networks. In *International Conference on Machine Learning* (pp. 1024–1034).: PMLR.

Bordenave, C., Caputo, P., & Chafai, D. (2008). Circular law theorem for random markov matrices. *Probability Theory and Related Fields*, 152.

Borisov, V., Leemann, T., Seßler, K., Haug, J., Pawelczyk, M., & Kasneci, G. (2022). Deep neural networks and tabular data: A survey. *IEEE Transactions on Neural Networks and Learning Systems*.

Boyd, S. & Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

Breiman, L. (2001). Random forests. *Machine learning*, 45, 5–32.

Bronstein, I., Brutzkus, A., & Globerson, A. (2022). On the inductive bias of neural networks for learning read-once dnfs. In *Uncertainty in Artificial Intelligence* (pp. 255–265).: PMLR.

Brown, P. F., deSouza, P. V., Mercer, R. L., Pietra, V. J. D., & Lai, J. C. (1992). Class-based n-gram models of natural language. *Comput. Linguist.*, 18(4), 467–479.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Cammarata, N., Goh, G., Carter, S., Schubert, L., Petrov, M., & Olah, C. (2020). Curve detectors. *Distill*, 5(6), e00024–003.

Cao, Y., Fang, Z., Wu, Y., Zhou, D.-X., & Gu, Q. (2019). Towards understanding the spectral bias of deep learning. *arXiv preprint arXiv:1912.01198*.

Chan, S., Santoro, A., Lampinen, A., Wang, J., Singh, A., Richemond, P., McClelland, J., & Hill, F. (2022). Data distributional properties drive emergent in-context learning in transformers. *Advances in Neural Information Processing Systems*, 35, 18878–18891.

Chen, A., Shwartz-Ziv, R., Cho, K., Leavitt, M. L., & Saphra, N. (2023). Sudden drops in the loss: Syntax acquisition, phase transitions, and simplicity bias in mlms.

Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., & Mordatch, I. (2021a). Decision transformer: Reinforcement learning via sequence modeling. *arXiv preprint arXiv:2106.01345*.

Chen, M., Li, X., & Zhao, T. (2019). On generalization bounds of a family of recurrent neural networks. *arXiv preprint arXiv:1910.12947*.

Chen, M., Tworek, J., Jun, H., Yuan, Q., Ponde, H., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., et al. (2021b). Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Chen, S. & Meka, R. (2020). Learning polynomials in few relevant dimensions. In *Conference on Learning Theory* (pp. 1161–1227).: PMLR.

Chen, T. & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 785–794).

Chizat, L. & Bach, F. (2020). Implicit bias of gradient descent for wide two-layer neural networks trained with the logistic loss.

Chizat, L., Oyallon, E., & Bach, F. (2019). On lazy training in differentiable programming. *Advances in neural information processing systems*, 32.

Chomsky, N. (1956). Three models for the description of language. *IRE Transactions on information theory*, 2(3), 113–124.

Choromanski, K., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J., Mohiuddin, A., Kaiser, L., et al. (2020). Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., et al. (2022). Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

Chughtai, B., Chan, L., & Nanda, N. (2023). A toy model of universality: Reverse engineering how networks learn group operations. *arXiv preprint arXiv:2302.03025*.

Clark, K., Khandelwal, U., Levy, O., & Manning, C. D. (2019). What does bert look at? an analysis of bert's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP* (pp. 276–286).

Corsi, A. K., Wightman, B., & Chalfie, M. (2015). A transparent window into biology: a primer on caenorhabditis elegans. *Genetics*, 200(2), 387–407.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4), 303–314.

Dai, D., Sun, Y., Dong, L., Hao, Y., Sui, Z., & Wei, F. (2022). Why can gpt learn in-context? language models secretly perform gradient descent as meta optimizers. *arXiv preprint arXiv:2212.10559*.

Damian, A., Lee, J., & Soltanolkotabi, M. (2022). Neural networks can learn representations with gradient descent. In *Conference on Learning Theory* (pp. 5413–5452).: PMLR.

Damian, A., Nichani, E., Ge, R., & Lee, J. D. (2023). Smoothing the landscape boosts the signal for sgd: Optimal sample complexity for learning single index models. *arXiv preprint arXiv:2305.10633*.

Daniely, A. & Malach, E. (2020). Learning parities with neural networks. *Advances in Neural Information Processing Systems*, 33.

d'Ascoli, S., Touvron, H., Leavitt, M., Morcos, A., Biroli, G., & Sagun, L. (2021). Convit: Improving vision transformers with soft convolutional inductive biases. *arXiv preprint arXiv:2103.10697*.

Dehghani, M., Gouws, S., Vinyals, O., Uszkoreit, J., & Kaiser, Ł. (2018). Universal transformers. *arXiv preprint arXiv:1807.03819*.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Diakonikolas, I., Goel, S., Karmalkar, S., Klivans, A. R., & Soltanolkotabi, M. (2020). Approximation schemes for relu regression. In *Conference on Learning Theory* (pp. 1452–1485).: PMLR.

Dong, Q., Li, L., Dai, D., Zheng, C., Wu, Z., Chang, B., Sun, X., Xu, J., & Sui, Z. (2022). A survey for in-context learning. *arXiv preprint arXiv:2301.00234*.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*.

Du, S. S., Zhai, X., Poczos, B., & Singh, A. (2018). Gradient descent provably optimizes over-parameterized neural networks. *arXiv preprint arXiv:1810.02054*.

Dudley, R. M. (1967). The sizes of compact subsets of hilbert space and continuity of gaussian processes. *Journal of Functional Analysis*, 1(3), 290–330.

Edelman, B., Goel, S., Kakade, S., Malach, E., & Zhang, C. (2024a). Pareto frontiers in deep feature learning: Data, compute, width, and luck. *Advances in Neural Information Processing Systems*, 36.

Edelman, B. L., Edelman, E., Goel, S., Malach, E., & Tsilivis, N. (2024b). The evolution of statistical induction heads: In-context learning markov chains. *arXiv preprint arXiv:2402.11004*.

Edelman, B. L., Goel, S., Kakade, S., & Zhang, C. (2022). Inductive biases and variable creation in self-attention mechanisms. In *International Conference on Machine Learning* (pp. 5793–5831).: PMLR.

Elhage, N., Hume, T., Olsson, C., Schiefer, N., Henighan, T., Kravec, S., Hatfield-Dodds, Z., Lasenby, R., Drain, D., Chen, C., et al. (2022). Toy models of superposition. *arXiv preprint arXiv:2209.10652*.

Elhage, N., Nanda, N., Olsson, C., Henighan, T., Joseph, N., Mann, B., Askell, A., Bai, Y., Chen, A., Conerly, T., et al. (2021). A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 1.

Engel, A. & Van den Broeck, C. (2001). *Statistical mechanics of learning*. Cambridge University Press.

Esseen, C.-G. (1942). On the Liapunov limit error in the theory of probability. *Ark. Mat. Astr. Fys.*, 28, 1–19.

Feldman, V. (2008). Evolvability from learning algorithms. In *Proceedings of the fortieth annual ACM symposium on Theory of computing* (pp. 619–628).

Feldman, V., Guzman, C., & Vempala, S. (2017). Statistical query algorithms for mean vector estimation and stochastic convex optimization. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms* (pp. 1265–1277).: SIAM.

Frankle, J. & Carbin, M. (2018). The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*.

Frei, S., Cao, Y., & Gu, Q. (2020). Agnostic learning of a single neuron with gradient descent. *Advances in Neural Information Processing Systems*, 33, 5417–5428.

Frei, S., Chatterji, N. S., & Bartlett, P. L. (2022a). Random feature amplification: Feature learning and generalization in neural networks. *arXiv preprint arXiv:2202.07626*.

Frei, S., Vardi, G., Bartlett, P., & Srebro, N. (2023). Benign overfitting in linear classifiers and leaky relu networks from kkt conditions for margin maximization. In *The Thirty Sixth Annual Conference on Learning Theory* (pp. 3173–3228).: PMLR.

Frei, S., Vardi, G., Bartlett, P., Srebro, N., & Hu, W. (2022b). Implicit bias in leaky relu networks trained on high-dimensional data. In *The Eleventh International Conference on Learning Representations*.

Friedman, D., Wettig, A., & Chen, D. (2024). Learning transformer programs. *Advances in Neural Information Processing Systems*, 36.

Gabaix, X. & Laibson, D. (2008). The seven properties of good models. *The foundations of positive and normative economics: A handbook*, (pp. 292–319).

Gardner, E. & Derrida, B. (1989). Three unfinished works on the optimal storage capacity of networks. *Journal of Physics A: Mathematical and General*, 22(12), 1983.

Garg, S., Tsipras, D., Liang, P., & Valiant, G. (2022). What can transformers learn in-context? A case study of simple function classes. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, & A. Oh (Eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Gilbert, N. (2019). *Agent-based models*. Sage Publications.

Glasgow, M. (2023). Sgd finds then tunes features in two-layer neural networks with near-optimal sample complexity: A case study in the xor problem. In *The Twelfth International Conference on Learning Representations*.

Glorot, X. & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (pp. 249–256).: JMLR Workshop and Conference Proceedings.

Goel, S., Karmalkar, S., & Klivans, A. (2019). Time/accuracy tradeoffs for learning a ReLU with respect to Gaussian marginals. *Advances in Neural Information Processing Systems*, 32.

Goldreich, O. & Levin, L. A. (1989). A hard-core predicate for all one-way functions. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing* (pp. 25–32).

Goldt, S., Advani, M., Saxe, A. M., Krzakala, F., & Zdeborová, L. (2019). Dynamics of stochastic gradient descent for two-layer neural networks in the teacher-student setup. *Advances in neural information processing systems*, 32.

Golowich, N., Rakhlin, A., & Shamir, O. (2018). Size-independent sample complexity of neural networks. In *Conference On Learning Theory* (pp. 297–299).: PMLR.

Gorishniy, Y., Rubachev, I., Khrulkov, V., & Babenko, A. (2021). Revisiting deep learning models for tabular data. *Advances in Neural Information Processing Systems*, 34, 18932–18943.

Goyal, A., Didolkar, A., Ke, N. R., Blundell, C., Beaudoin, P., Heess, N., Mozer, M. C., & Bengio, Y. (2021). Neural production systems. *Advances in Neural Information Processing Systems*, 34.

Goyal, A., Lamb, A., Hoffmann, J., Sodhani, S., Levine, S., Bengio, Y., & Schölkopf, B. (2020). Recurrent independent mechanisms. In *International Conference on Learning Representations*.

Grinsztajn, L., Oyallon, E., & Varoquaux, G. (2022). Why do tree-based models still outperform deep learning on typical tabular data? In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, & A. Oh (Eds.), *Advances in Neural Information Processing Systems*, volume 35 (pp. 507–520).: Curran Associates, Inc.

Gromov, A. (2023). Grokking modular arithmetic. *arXiv preprint arXiv:2301.02679*.

Grosse, R. (2022). Neural net training dynamics chapter 1 - a toy model: Linear regression.

Gunasekar, S., Lee, J. D., Soudry, D., & Srebro, N. (2018). Implicit bias of gradient descent on linear convolutional networks. *Advances in neural information processing systems*, 31.

Guo, T., Hu, W., Mei, S., Wang, H., Xiong, C., Savarese, S., & Bai, Y. (2023). How do transformers learn in-context beyond simple functions? a case study on learning with representations. *arXiv preprint arXiv:2310.10616*.

Hahn, M. (2020). Theoretical limitations of self-attention in neural sequence models. *Transactions of the Association for Computational Linguistics*, 8, 156–171.

Hansel, D., Mato, G., & Meunier, C. (1992). Memorization without generalization in a multilayered neural network. *EPL (Europhysics Letters)*, 20(5), 471.

Hardy, G. H., Littlewood, J. E., Pólya, G., Pólya, G., et al. (1952). *Inequalities*. Cambridge University Press.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision* (pp. 1026–1034).

Hendel, R., Geva, M., & Globerson, A. (2023). In-context learning creates task vectors. In *Findings of the Association for Computational Linguistics: EMNLP 2023* (pp. 9318–9333).

Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., & Steinhardt, J. (2021). Measuring mathematical problem solving with the math dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Henighan, T., Kaplan, J., Katz, M., Chen, M., Hesse, C., Jackson, J., Jun, H., Brown, T. B., Dhariwal, P., Gray, S., et al. (2020). Scaling laws for autoregressive generative modeling. *arXiv preprint arXiv:2010.14701*.

Hewitt, J., Thickstun, J., Manning, C. D., & Liang, P. (2023). Backpack language models. *arXiv preprint arXiv:2305.16765*.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., de Las Casas, D., Hendricks, L. A., Welbl, J., Clark, A., et al. (2022). An empirical analysis of compute-optimal large language model training. *Advances in Neural Information Processing Systems*, 35, 30016–30030.

Hoogland, J., Wang, G., Farrugia-Roberts, M., Carroll, L., Wei, S., & Murfet, D. (2024). The developmental landscape of in-context learning. *CoRR*, abs/2402.02364.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5), 359–366.

Hron, J., Bahri, Y., Sohl-Dickstein, J., & Novak, R. (2020). Infinite attention: Nngp and ntk for deep attention networks. In *International Conference on Machine Learning* (pp. 4376–4386).: PMLR.

Hupkes, D., Dankers, V., Mul, M., & Bruni, E. (2020). Compositionality decomposed: How do neural networks generalise? *Journal of Artificial Intelligence Research*, 67, 757–795.

Hutter, M. (2021). Learning curve theory. *arXiv preprint arXiv:2102.04074*.

Ishai, Y., Kushilevitz, E., Ostrovsky, R., & Sahai, A. (2008). Cryptography with constant computational overhead. In *Proceedings of the 40th Annual ACM Symposium on the Theory of Computing* (pp. 433–442).

Jacot, A., Gabriel, F., & Hongler, C. (2018). Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31.

Jaegle, A., Borgeaud, S., Alayrac, J.-B., Doersch, C., Ionescu, C., Ding, D., Koppula, S., Zoran, D., Brock, A., Shelhamer, E., et al. (2021a). Perceiver io: A general architecture for structured inputs & outputs. *arXiv preprint arXiv:2107.14795*.

Jaegle, A., Gimeno, F., Brock, A., Zisserman, A., Vinyals, O., & Carreira, J. (2021b). Perceiver: General perception with iterative attention. *arXiv preprint arXiv:2103.03206*.

Janner, M., Li, Q., & Levine, S. (2021). Reinforcement learning as one big sequence modeling problem. *arXiv preprint arXiv:2106.02039*.

Ji, Z. & Telgarsky, M. (2018). Gradient descent aligns the layers of deep linear networks. *arXiv preprint arXiv:1810.02032*.

Ji, Z. & Telgarsky, M. (2019). Polylogarithmic width suffices for gradient descent to achieve arbitrarily small test error with shallow relu networks. *arXiv preprint arXiv:1909.12292*.

Ji, Z. & Telgarsky, M. (2020). Directional convergence and alignment in deep learning. *Advances in Neural Information Processing Systems*, 33, 17176–17186.

Jiang, Y., Neyshabur, B., Mobahi, H., Krishnan, D., & Bengio, S. (2019). Fantastic generalization measures and where to find them. *arXiv preprint arXiv:1912.02178*.

Johnson, W. B., Lindenstrauss, J., & Schechtman, G. (1986). Extensions of lipschitz maps into banach spaces. *Israel Journal of Mathematics*, 54(2), 129–138.

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., et al. (2021). Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873), 583–589.

Kabashima, Y. (1994). Perfect loss of generalization due to noise in k= 2 parity machines. *Journal of Physics A: Mathematical and General*, 27(6), 1917.

Kaggle (2021). State of data science and machine learning 2021.

Kalimeris, D., Kaplun, G., Nakkiran, P., Edelman, B., Yang, T., Barak, B., & Zhang, H. (2019). Sgd on neural networks learns functions of increasing complexity. *Advances in neural information processing systems*, 32.

Kamath, P., Montasser, O., & Srebro, N. (2020). Approximate is good enough: Probabilistic variants of dimensional and margin complexity. In *Conference on Learning Theory* (pp. 2236–2262).: PMLR.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., & Amodei, D. (2020). Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Karpathy, A. (2023). Mingpt. https://github.com/karpathy/minGPT/tree/master.

Kearns, M. (1998). Efficient noise-tolerant learning from statistical queries. *Journal of the ACM (JACM)*, 45(6), 983–1006.

Kerg, G., Kanuparthi, B., Goyal, A., Goyette, K., Bengio, Y., & Lajoie, G. (2020). Untangling tradeoffs between recurrence and self-attention in artificial neural networks. *Advances in Neural Information Processing Systems*, 33.

Kim, H., Papamakarios, G., & Mnih, A. (2021). The lipschitz constant of self-attention. In *International Conference on Machine Learning* (pp. 5562–5571).: PMLR.

Kingma, D. P. & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kirsch, L., Harrison, J., Sohl-Dickstein, J., & Metz, L. (2022). General-purpose in-context learning by meta-learning transformers. *CoRR*, abs/2212.04458.

Klivans, A. & Kothari, P. (2014). Embedding hard learning problems into gaussian space. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2014)*: Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Klivans, A. R., O'Donnell, R., & Servedio, R. A. (2004). Learning intersections and thresholds of halfspaces. *Journal of Computer and System Sciences*, 68(4), 808–840.

Kol, G., Raz, R., & Tal, A. (2017). Time-space hardness of learning sparse parities. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing* (pp. 1067–1080).

Kosmann-Schwarzbach, Y. et al. (2010). *Groups and symmetries*. Springer.

Krebs, H. A. (1975). The august krogh principle:"for many problems there is an animal on which it can be most conveniently studied". *Journal of Experimental Zoology*, 194(1), 221–226.

Kumar, T., Bordelon, B., Gershman, S. J., & Pehlevan, C. (2023). Grokking as the transition from lazy to rich training dynamics. *CoRR*, abs/2310.06110.

Kunin, D., Yamamura, A., Ma, C., & Ganguli, S. (2022). The asymmetric maximum margin bias of quasi-homogeneous neural networks. In *The Eleventh International Conference on Learning Representations*.

Kushilevitz, E. & Mansour, Y. (1993). Learning decision trees using the fourier spectrum. *SIAM Journal on Computing*, 22(6), 1331–1348.

Laurent, B. & Massart, P. (2000). Adaptive estimation of a quadratic functional by model selection. *Annals of Statistics*, (pp. 1302–1338).

Lee-Thorp, J., Ainslie, J., Eckstein, I., & Ontanon, S. (2021). Fnet: Mixing tokens with fourier transforms. *arXiv preprint arXiv:2105.03824*.

Lewkowycz, A., Andreassen, A., Dohan, D., Dyer, E., Michalewski, H., Ramasesh, V., Slone, A., Anil, C., Schlag, I., Gutman-Solo, T., et al. (2022). Solving quantitative reasoning problems with language models. *Advances in Neural Information Processing Systems*, 35, 3843–3857.

Li, Y., Ildiz, M. E., Papailiopoulos, D., & Oymak, S. (2023). Transformers as algorithms: Generalization and stability in in-context learning. In *International Conference on Machine Learning* (pp. 19565–19594).: PMLR.

Liang, P., Bommasani, R., Lee, T., Tsipras, D., Soylu, D., Yasunaga, M., Zhang, Y., Narayanan, D., Wu, Y., Kumar, A., et al. (2023). Holistic evaluation of language models. *Transactions on Machine Learning Research*.

Likhosherstov, V., Choromanski, K., & Weller, A. (2021). On the expressive power of self-attention matrices. *arXiv preprint arXiv:2106.03764*.

Liu, B., Ash, J., Goel, S., Krishnamurthy, A., & Zhang, C. (2024). Exposing attention glitches with flip-flop language modeling. *Advances in Neural Information Processing Systems*, 36.

Liu, B., Ash, J. T., Goel, S., Krishnamurthy, A., & Zhang, C. (2022a). Transformers learn shortcuts to automata. *arXiv preprint arXiv:2210.10749*.

Liu, P. J., Saleh, M., Pot, E., Goodrich, B., Sepassi, R., Kaiser, L., & Shazeer, N. (2018). Generating wikipedia by summarizing long sequences. *arXiv preprint arXiv:1801.10198*.

Liu, Z., Kitouni, O., Nolte, N. S., Michaud, E., Tegmark, M., & Williams, M. (2022b). Towards understanding grokking: An effective theory of representation learning. *Advances in Neural Information Processing Systems*, 35, 34651–34663.

Liu, Z., Michaud, E. J., & Tegmark, M. (2023). Omnigrok: Grokking beyond algorithmic data. In *The Eleventh International Conference on Learning Representations*.

Long, P. M. & Sedghi, H. (2019). Generalization bounds for deep convolutional neural networks. *arXiv preprint arXiv:1905.12600*.

Lu, K., Grover, A., Abbeel, P., & Mordatch, I. (2021). Pretrained transformers as universal computation engines. *arXiv preprint arXiv:2103.05247*.

Luong, M.-T., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Lutz, P., Arnould, L., Boyer, C., & Scornet, E. (2022). Sparse tree-based initialization for neural networks. *arXiv preprint arXiv:2209.15283*.

Lyu, K., Jin, J., Li, Z., Du, S. S., Lee, J. D., & Hu, W. (2023). Dichotomy of early and late phase implicit biases can provably induce grokking. *CoRR*, abs/2311.18817.

Lyu, K. & Li, J. (2019). Gradient descent maximizes the margin of homogeneous neural networks. In *International Conference on Learning Representations*.

Lyu, K., Li, Z., Wang, R., & Arora, S. (2021). Gradient descent on two-layer nets: Margin maximization and simplicity bias. *Advances in Neural Information Processing Systems*, 34, 12978–12991.

Makkuva, A. V., Bondaschi, M., Girish, A., Nagle, A., Jaggi, M., Kim, H., & Gastpar, M. (2024). Attention with markov: A framework for principled analysis of transformers via markov chains. *CoRR*, abs/2402.04161.

Malach, E., Kamath, P., Abbe, E., & Srebro, N. (2021). Quantifying the benefit of using differentiable learning over tangent kernels. In *International Conference on Machine Learning* (pp. 7379–7389).: PMLR.

Malach, E. & Shalev-Shwartz, S. (2019). Is deeper better only when shallow is good? *Advances in Neural Information Processing Systems*, 32.

Malach, E. & Shalev-Shwartz, S. (2022). When hardness of approximation meets hardness of learning. *Journal of Machine Learning Research*, 23(91), 1–24.

Merrill, W., Tsilivis, N., & Shukla, A. (2023). A tale of two circuits: Grokking as competition of sparse and dense subnetworks. In *ICLR 2023 Workshop on Mathematical and Empirical Understanding of Foundation Models*.

Michaud, E. J., Liu, Z., Girit, U., & Tegmark, M. (2023). The quantization model of neural scaling. *arXiv preprint arXiv:2303.13506*.

Minsky, M. & Papert, S. (1969). *Perceptrons: an introduction to computational geometry*. MIT Press.

Mitchison, G. & Durbin, R. (1989). Bounds on the learning capacity of some multi-layer networks. *Biological Cybernetics*, 60(5), 345–365.

Morwani, D., Batra, J., Jain, P., & Netrapalli, P. (2023a). Simplicity bias in 1-hidden layer neural networks.

Morwani, D., Edelman, B. L., Oncescu, C.-A., Zhao, R., & Kakade, S. (2023b). Feature emergence via margin maximization: case studies in algebraic tasks. *arXiv preprint arXiv:2311.07568*.

Mossel, E., O'Donnell, R., & Servedio, R. P. (2003). Learning juntas. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing* (pp. 206–212).

Nakkiran, P. & Bansal, Y. (2020). Distributional generalization: A new kind of generalization. *arXiv preprint arXiv:2009.08092*.

Nakkiran, P., Kaplun, G., Bansal, Y., Yang, T., Barak, B., & Sutskever, I. (2021). Deep double descent: Where bigger models and more data hurt. *Journal of Statistical Mechanics: Theory and Experiment*, 2021(12), 124003.

Nanda, N., Chan, L., Liberum, T., Smith, J., & Steinhardt, J. (2023). Progress measures for grokking via mechanistic interpretability. *arXiv preprint arXiv:2301.05217*.

Nanda, N. & Lieberum, T. (2022). A mechanistic interpretability analysis of grokking. *Alignment Forum*.

Neyshabur, B., Bhojanapalli, S., & Srebro, N. (2017). A pac-bayesian approach to spectrally-normalized margin bounds for neural networks. *arXiv preprint arXiv:1707.09564*.

Neyshabur, B., Tomioka, R., & Srebro, N. (2015). Norm-based capacity control in neural networks. In *Conference on Learning Theory* (pp. 1376–1401).: PMLR.

Nielsen, M. A. (2015). *Neural networks and deep learning*, volume 25. Determination press San Francisco, CA.

O'Donnell, R. (2014). *Analysis of Boolean functions*. Cambridge University Press.

Olah, C., Cammarata, N., Schubert, L., Goh, G., Petrov, M., & Carter, S. (2020). Zoom in: An introduction to circuits. *Distill*. https://distill.pub/2020/circuits/zoom-in.

Olsson, C., Elhage, N., Nanda, N., Joseph, N., DasSarma, N., Henighan, T., Mann, B., Askell, A., Bai, Y., Chen, A., Conerly, T., Drain, D., Ganguli, D., Hatfield-Dodds, Z., Hernandez, D., Johnston, S., Jones, A., Kernion, J., Lovitt, L., Ndousse, K., Amodei, D., Brown, T., Clark, J., Kaplan, J., McCandlish, S., & Olah, C. (2022). In-context learning and induction heads. *Transformer Circuits Thread*. https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html.

Opper, M. (1994). Learning and generalization in a two-layer neural network: The role of the vapnik-chervonvenkis dimension. *Physical review letters*, 72(13), 2113.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., & Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 32* (pp. 8024–8035). Curran Associates, Inc.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.

Petroni, F., Rocktäschel, T., Riedel, S., Lewis, P., Bakhtin, A., Wu, Y., & Miller, A. (2019). Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (pp. 2463–2473).

Pietrzak, K. (2012). Cryptography from learning parity with noise. In *International Conference on Current Trends in Theory and Practice of Computer Science* (pp. 99–114).: Springer.

Polu, S. & Sutskever, I. (2020). Generative language modeling for automated theorem proving. *arXiv preprint arXiv:2009.03393*.

Power, A., Burda, Y., Edwards, H., Babuschkin, I., & Misra, V. (2021). Grokking: Generalization beyond overfitting on small algorithmic datasets. In *ICLR MATH-AI Workshop*.

Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training. preprint, available at https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, 1(8), 9.

Rahaman, N., Baratin, A., Arpit, D., Draxler, F., Lin, M., Hamprecht, F., Bengio, Y., & Courville, A. (2019). On the spectral bias of neural networks. In *International Conference on Machine Learning* (pp. 5301–5310).: PMLR.

Rahimi, A. & Recht, B. (2007). Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20.

Reddy, G. (2023). The mechanistic basis of data dependence and abrupt learning in an in-context classification task.

Refinetti, M., Goldt, S., Krzakala, F., & Zdeborová, L. (2021). Classifying high-dimensional gaussian mixtures: Where kernel methods fail and neural networks succeed. In *International Conference on Machine Learning* (pp. 8936–8947).: PMLR.

Rogers, A., Kovaleva, O., & Rumshisky, A. (2020). A primer in BERTology: What we know about how BERT works. *Transactions of the Association for Computational Linguistics*, 8, 842–866.

Rosen-Zvi, M., Klein, E., Kanter, I., & Kinzel, W. (2002). Mutual learning in a tree parity machine and its application to cryptography. *Physical Review E*, 66(6), 066135.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6), 386.

Saad, D. & Solla, S. (1995a). Dynamics of on-line gradient descent learning for multilayer neural networks. *Advances in neural information processing systems*, 8.

Saad, D. & Solla, S. A. (1995b). On-line learning in soft committee machines. *Physical Review E*, 52(4), 4225.

Sanford, C., Hsu, D. J., & Telgarsky, M. (2024). Representational strengths and limitations of transformers. *Advances in Neural Information Processing Systems*, 36.

Saxe, A., McClelland, J., & Ganguli, S. (2014). Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *Proceedings of the International Conference on Learning Representations 2014*: International Conference on Learning Representations 2014.

Saxton, D., Grefenstette, E., Hill, F., & Kohli, P. (2018). Analysing mathematical reasoning abilities of neural models. In *International Conference on Learning Representations*.

Schaeffer, R., Miranda, B., & Koyejo, S. (2023). Are emergent abilities of large language models a mirage? *CoRR*, abs/2304.15004.

Shah, H., Tamuly, K., Raghunathan, A., Jain, P., & Netrapalli, P. (2020). The pitfalls of simplicity bias in neural networks. *Advances in Neural Information Processing Systems*, 33, 9573–9585.

Shalev-Shwartz, S. & Ben-David, S. (2014). *Understanding machine learning: From theory to algorithms*. Cambridge university press.

Shalev-Shwartz, S., Shamir, O., & Shammah, S. (2017). Failures of gradient-based deep learning. In *International Conference on Machine Learning* (pp. 3067–3075).: PMLR.

Shamir, O. & Zhang, T. (2013). Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes. In *International conference on machine learning* (pp. 71–79).: PMLR.

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell system technical journal*, 27(3), 379–423.

Shaw, P., Uszkoreit, J., & Vaswani, A. (2018). Self-attention with relative position representations. In M. A. Walker, H. Ji, & A. Stent (Eds.), *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 2 (Short Papers)* (pp. 464–468).: Association for Computational Linguistics.

Shi, Z., Wei, J., & Liang, Y. (2021). A theoretical analysis on feature learning in neural networks: Emergence from inputs and advantage over fixed features. In *International Conference on Learning Representations*.

Shpilka, A., Yehudayoff, A., et al. (2010). Arithmetic circuits: A survey of recent results and open questions. *Foundations and Trends® in Theoretical Computer Science*, 5(3–4), 207–388.

Siegelmann, H. T. & Sontag, E. D. (1995). On the computational power of neural nets. *Journal of computer and system sciences*, 50(1), 132–150.

Simonetti, R. & Caticha, N. (1996). On-line learning in parity machines. *Journal of Physics A: Mathematical and General*, 29(16), 4859.

Snell, C., Zhong, R., Klein, D., & Steinhardt, J. (2021). Approximating how single head attention learns. *arXiv preprint arXiv:2103.07601*.

Soudry, D., Hoffer, E., Nacson, M. S., Gunasekar, S., & Srebro, N. (2018). The implicit bias of gradient descent on separable data. *The Journal of Machine Learning Research*, 19(1), 2822–2878.

Srivastava, A., Rastogi, A., Rao, A., Shoeb, A. A. M., Abid, A., Fisch, A., Brown, A. R., Santoro, A., Gupta, A., Garriga-Alonso, A., et al. (2022). Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*.

Tancik, M., Srinivasan, P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J., & Ng, R. (2020). Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in neural information processing systems*, 33, 7537–7547.

Tay, Y., Dehghani, M., Abnar, S., Shen, Y., Bahri, D., Pham, P., Rao, J., Yang, L., Ruder, S., & Metzler, D. (2020). Long range arena: A benchmark for efficient transformers. *arXiv preprint arXiv:2011.04006*.

Telgarsky, M. (2022). Feature selection with gradient descent on two-layer networks in low-rotation regimes. *arXiv preprint arXiv:2208.02789*.

Tenney, I., Das, D., & Pavlick, E. (2019). Bert rediscovers the classical nlp pipeline. *arXiv preprint arXiv:1905.05950*.

Titsworth, R. C. (1962). *Correlation properties of cyclic sequences*. PhD thesis, California Institute of Technology.

Tolstikhin, I. O., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., Yung, J., Steiner, A., Keysers, D., Uszkoreit, J., et al. (2021). MLP-Mixer: An all-MLP architecture for vision. *Advances in neural information processing systems*, 34, 24261–24272.

Valle-Perez, G., Camargo, C. Q., & Louis, A. A. (2018). Deep learning generalizes because the parameter-function map is biased towards simple functions. *arXiv preprint arXiv:1805.08522*.

Valvoda, J., Saphra, N., Rawski, J., Williams, A., & Cotterell, R. (2022). Benchmarking compositionality with formal languages. *arXiv preprint arXiv:2208.08195*.

Vardi, G. (2023). On the implicit bias in deep-learning algorithms. *Communications of the ACM*, 66(6), 86–93.

Vardi, G., Shamir, O., & Srebro, N. (2022). On margin maximization in linear and relu networks. *Advances in Neural Information Processing Systems*, 35, 37024–37036.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Von Oswald, J., Niklasson, E., Randazzo, E., Sacramento, J., Mordvintsev, A., Zhmoginov, A., & Vladymyrov, M. (2023). Transformers learn in-context by gradient descent. In *International Conference on Machine Learning* (pp. 35151–35174).: PMLR.

Vuckovic, J., Baratin, A., & Combes, R. T. d. (2020). A mathematical theory of attention. *arXiv preprint arXiv:2007.02876*.

Wainwright, M. J. (2019). *High-dimensional statistics: A non-asymptotic viewpoint*, volume 48. Cambridge University Press.

Watkin, T. L., Rau, A., & Biehl, M. (1993). The statistical mechanics of learning a rule. *Reviews of Modern Physics*, 65(2), 499.

Wei, C., Chen, Y., & Ma, T. (2021). Statistically meaningful approximation: a case study on approximating turing machines with transformers. *arXiv preprint arXiv:2107.13163*.

Wei, C., Lee, J. D., Liu, Q., & Ma, T. (2019a). Regularization matters: generalization and optimization of neural nets vs their induced kernel. *Advances in Neural Information Processing Systems*, 32.

Wei, C., Lee, J. D., Liu, Q., & Ma, T. (2019b). Regularization matters: Generalization and optimization of neural nets v.s. their induced kernel. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems*, volume 32: Curran Associates, Inc.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. (2022). Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35, 24824–24837.

Wu, J., Zou, D., Chen, Z., Braverman, V., Gu, Q., & Bartlett, P. L. (2023). How many pretraining tasks are needed for in-context learning of linear regression? *arXiv preprint arXiv:2310.08391*.

Xie, S. M., Raghunathan, A., Liang, P., & Ma, T. (2022). An explanation of in-context learning as implicit bayesian inference. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., & Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning* (pp. 2048–2057).: PMLR.

Yang, G. (2019). Wide feedforward or recurrent neural networks of any architecture are gaussian processes. *Advances in Neural Information Processing Systems*, 32.

Yang, G. (2020). Tensor programs II: Neural tangent kernel for any architecture. *arXiv preprint arXiv:2006.14548*.

Yang, G., Hu, E. J., Babuschkin, I., Sidor, S., Liu, X., Farhi, D., Ryder, N., Pachocki, J., Chen, W., & Gao, J. (2022a). Tensor programs v: Tuning large neural networks via zero-shot hyperparameter transfer. *arXiv preprint arXiv:2203.03466*.

Yang, J., Lindenbaum, O., & Kluger, Y. (2022b). Locally sparse neural networks for tabular biomedical data. In *International Conference on Machine Learning* (pp. 25123–25153).: PMLR.

Yao, S., Peng, B., Papadimitriou, C., & Narasimhan, K. (2021). Self-attention networks can process bounded hierarchical languages. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)* (pp. 3770–3785).

Yehudai, G. & Ohad, S. (2020). Learning a single neuron with gradient methods. In *Conference on Learning Theory* (pp. 3756–3786).: PMLR.

Yehudai, G. & Shamir, O. (2019). On the power and limitations of random features for understanding neural networks. *Advances in Neural Information Processing Systems*, 32.

Yun, C., Bhojanapalli, S., Rawat, A. S., Reddi, S. J., & Kumar, S. (2019). Are transformers universal approximators of sequence-to-sequence functions? *arXiv preprint arXiv:1912.10077*.

Zeiler, M. D. & Fergus, R. (2014). Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13* (pp. 818–833).: Springer.

Zhai, X., Kolesnikov, A., Houlsby, N., & Beyer, L. (2022). Scaling vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 12104–12113).

Zhang, C., Raghu, M., Kleinberg, J., & Bengio, S. (2021). Pointer value retrieval: A new benchmark for understanding the limits of neural network generalization. *arXiv preprint arXiv:2107.12580*.

Zhang, J., Karimireddy, S. P., Veit, A., Kim, S., Reddi, S., Kumar, S., & Sra, S. (2020). Why are adaptive methods good for attention models? *Advances in Neural Information Processing Systems*, 33, 15383–15393.

Zhang, J., Karimireddy, S. P., Veit, A., Kim, S., Reddi, S. J., Kumar, S., & Sra, S. (2019). Why are adaptive methods good for attention models? *arXiv preprint arXiv:1912.03194*.

Zhang, T. (2002). Covering number bounds of certain regularized linear function classes. *Journal of Machine Learning Research*, 2(Mar), 527–550.

Zhang, Y., Backurs, A., Bubeck, S., Eldan, R., Gunasekar, S., & Wagner, T. (2022). Unveiling transformers with lego: a synthetic reasoning task. *arXiv preprint arXiv:2206.04301*.

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E., et al. (2024). Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36.

Zhenmei, S., Wei, J., & Liang, Y. (2022). A theoretical analysis on feature learning in neural networks: Emergence from inputs and advantage over fixed features. In *International Conference on Learning Representations*.

Zhong, Z., Liu, Z., Tegmark, M., & Andreas, J. (2023). The clock and the pizza: Two stories in mechanistic explanation of neural networks. *arXiv preprint arXiv:2306.17844*.

Zhou, H., Bradley, A., Littwin, E., Razin, N., Saremi, O., Susskind, J., Bengio, S., & Nakkiran, P. (2023). What algorithms can transformers learn? a study in length generalization. In *The 3rd Workshop on Mathematical Reasoning and AI at NeurIPS'23*.

Zou, D., Cao, Y., Zhou, D., & Gu, Q. (2020). Gradient descent optimizes over-parameterized deep relu networks. *Machine learning*, 109, 467–492.