Why Do You Grok? A Theoretical Analysis on Grokking Modular Addition

Mohamad Amin Mohamadi [1 2] **Zhiyuan Li** [1] **Lei Wu** [3] **Danica J. Sutherland** [2 4]

Abstract

We present a theoretical explanation of the "grokking" phenomenon (Power et al., 2022), where a model generalizes long after overfitting, for the originally-studied problem of modular addition. First, we show that early in gradient descent, when the "kernel regime" approximately holds, no permutation-equivariant model can achieve small population error on modular addition unless it sees at least a constant fraction of all possible data points. Eventually, however, models escape the kernel regime. We show that one-hidden-layer quadratic networks that achieve zero training loss with bounded ℓ_∞ norm generalize well with substantially fewer training points, and further show such networks exist and can be found by gradient descent with small ℓ_∞ regularization. We further provide empirical evidence that these networks leave the kernel regime only after initially overfitting. Taken together, our results strongly support the case for grokking as a consequence of the transition from kernel-like behavior to limiting behavior of gradient descent on deep networks.

1. Introduction

Understanding the generalization patterns of modern over-parameterized neural networks has been a long-standing goal of the deep learning community. Power et al. (2022) demonstrated an intriguing phenomenon they called "grokking" when learning transformers on small algorithmic tasks: neural networks can find a generalizing solution long after overfitting to the training dataset with poor generalization. This observation has lead to a stream of recent works aimed at uncovering the mechanisms that can lead a network to "grok," and properties of the final solutions, on various algorithmic tasks. Later, it was discovered that grokking can happen in tasks beyond modular arithmetic: in learning sparse parities (Barak et al., 2022; Bhattamishra et al., 2023), image classifiers (Liu et al., 2022b), greatest common divisors (Charton, 2023), matrix completion (Lyu et al., 2023), and k-sparse linear predictors (Lyu et al., 2023).

Grokking has been variously attributed to difficulty of representation learning (Liu et al., 2022a), the "slingshot" mechanism (Thilak et al., 2022), weight norm (Liu et al., 2022b; Varma et al., 2023), properties of the loss landscape (Notsawo et al., 2023), simplicity of the learned solution (Nanda et al., 2023) and other feature learning mechanisms (Levi et al., 2023; Rubin et al., 2024). Theoretically, Gromov (2023) presented an analytical construction for a one-hidden-layer MLP that solve modular addition and compatible with a grokking pattern with vanilla gradient descent.[1] Kumar et al. (2023) demonstrated grokking when training a one-hidden-layer MLP on a polynomial regression problem, as did Xu et al. (2023) for XOR data. The notion of delayed generalization was perhaps earlier observed by Li et al. (2022) when training diagonal linear networks with label noise SGD and through sharpness minimization, before it was known as grokking (Power et al., 2022).

Lyu et al. (2023) present a rigorous theoretical framework in which grokking can be provably demonstrated through a dichotomy of early and late implicit biases of the training algorithm. More specifically, they attribute the overfitting stage of grokking to the initial behavior of gradient descent as similar to a kernel predictor (Jacot et al., 2018; Arora et al., 2019b; Lee et al., 2019), and the generalization stage is then attributed to different late-phase implicit biases such as sharpness minimization (Blanc et al., 2020; Li et al., 2022; Damian et al., 2021; HaoChen et al., 2020), margin maximization (Soudry et al., 2022; Nacson et al., 2019; Wei et al., 2020; Lyu & Li, 2020), or parameter norm minimazation (Gunasekar et al., 2017; 2020; Arora et al., 2019a). Consistent with this framework, Kumar et al. (2023) hypothesized that grokking can be explained through the transition from the kernel regime to the "rich" regime, as long as

[1]Toyota Technological Institute at Chicago [2]Computer Science Department, University of British Columbia [3]School of Mathematical Sciences, Peking University [4]Alberta Machine Intelligence Institute. Correspondence to: Mohamad Amin Mohamadi <mohamadamin@ttic.edu>, Zhiyuan Li <zhiyuanli@ttic.edu>, Danica J. Sutherland <dsuth@ubc.ca>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

[1]Gromov (2023) claims this solution is the one found by gradient descent, but this did not seem to be the case in our experience.

the size of the training dataset is neither too small (where generalization would be impossible) nor too large (where generalization would be easy). They provided empirical support by considering scaling the model output, which is a rough proxy for the rate of feature learning in modular addition on one-hidden-layer MLPs.

This dichotomy between early kernel regime and late feature learning triggered by weak implicit or explicit regularization (i.e. the transition from lazy to rich regime) seems to be the most promising theory to explain grokking. Even so, two fundamental questions as to ***why*** grokking occurs on the original problem of **modular addition** have remained unanswered:

1. ***Why*** do a wide variety of architectures all fail to generalize in the initial phase of training, i.e. in the kernel regime? Is it because their kernels accidentally share some common property, or it is indeed a property of the modular addition task itself?

2. ***How*** does weak regularization encourage the network to learn generalizable features later in training?

Our Contributions. In this work, we address these questions with rigorous theoretical analyses of learning modular addition with gradient descent on one-hidden-layer MLPs.

We specifically focus on the problem

$$a + b \equiv c \pmod{p} \tag{1.1}$$

where $a, b, c \in \mathbb{Z}_p$ for a fixed $p \in \mathbb{N}$. We use (regularized) gradient descent on a one-hidden-layer MLP with quadratic activation on two different tasks: it is somewhat easier to analyze a "regression" task where we use square loss to learn a function of (a, b, c) that indicates whether (1.1) holds, but we also study the "classification" task in which we use cross-entropy loss to learn a p-way classifier to predict which c value satisfies (1.1) for a given (a, b). This discrete, noiseless setting has only a finite number of possible distinct data points: p^3 for regression, p^2 for classification.

To address the first question, we prove in Sections 3.1 and 4.1 that this task is *fundamentally hard* for permutation-equivariant kernel methods, due to inherent symmetries. Thus, permutation-equivariant networks which are well-approximated by their neural tangent kernel approximation cannot generalize well. This result is highly suggestive of why drastic overfitting with poor generalization has been empirically observed on this problem across a wide variety of architectures, losses, and learning algorithms (e.g. Power et al., 2022; Liu et al., 2022a; Gromov, 2023). We also prove that, although no such method can generalize, neural tangent kernel approaches *can* achieve zero training error.

To prove these results, we have developed a novel general technique to analyze the population ℓ_2 loss of learning general function classes with predictors of intrinsic dimension n (for instance kernel predictors with n training points), and present it in Appendix D.3. This framework allows us to prove lower bounds on population ℓ_2 loss for the general case of learning modular addition on m (instead of 2 or 3) summands with kernels, and might be of independent interest.

On the other hand, it has been consistently empirically observed that as long as a certain threshold of the available data is used for training, many different training algorithms on this setup will eventually generalize. Lyu et al. (2023) attribute this late phase generalization to the implicit biases of these algorithms, such as parameter norm minimization or margin maximization. This transition from kernel to rich regime has been widely observed (Chizat et al., 2019; Moroshko et al., 2020; Geiger et al., 2020; Telgarsky, 2022; Lyu et al., 2023). Likewise, we empirically confirm that in our setup, gradient descent drives the network to eventually leave the kernel regime and begin learning features. To address the second question – *why* this occurs – we prove in Sections 3.2 and 4.2 that networks similar enough to the min-norm or max-margin solution generalize with far fewer samples than required in the kernel regime. Thus, models with corresponding implicit biases can generalize well, answering the second question. In regression, our proofs are based on the smoothness of the ℓ_2 loss and utilize Rademacher complexity of networks whose weights have bounded norm. In classification, our proof is based on the PAC-Bayesian framework of McAllester (2003) for networks with bounded weights.

In summary, our **main contributions** are as follows:

1. We prove that networks in the kernel regime can only generalize if trained on **at least a constant fraction** of all possible data points, i.e. an $\Omega(1)$ portion, for regression (Section 3.1) and classification (Section 4.1).

2. We prove that networks with appropriate regularization can generalize with many fewer samples: $\boldsymbol{\mathcal{O}(1/p)}$ portion of all possible data points for square loss generalization on the regression task (Section 3.2), and $\boldsymbol{\tilde{\mathcal{O}}(1/p^{1/3})}$ for zero-one loss generalization on classification (Section 4.2).

2. Notations and Setup

We focus on learning modular addition, (1.1), with a one-hidden-layer network with no biases and quadratic activation, following Gromov (2023). Specifically, let e_i denote the vector of length p with 1 in its ith component and 0 elsewhere. In both models, our "base" network f maps the pair of integers (a, b) – represented as the vector $(e_a, e_b) \in \mathbb{R}^{2p}$

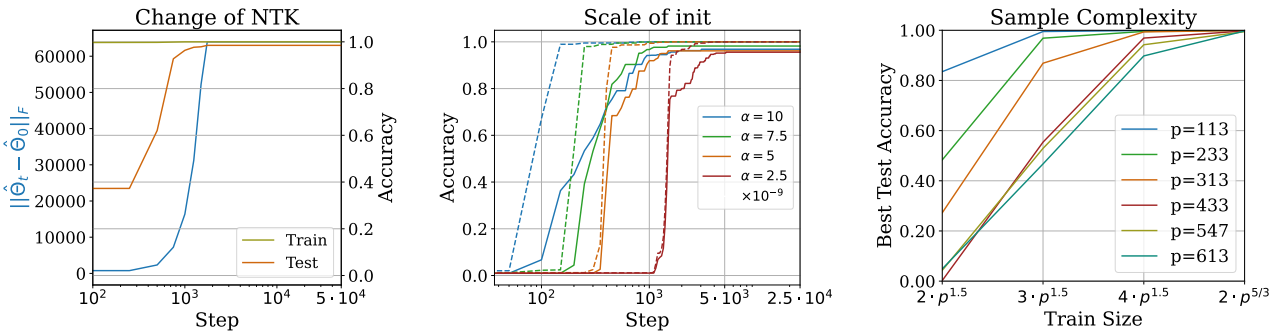

Figure 1: Empirical investigation into grokking modular addition on one-hidden-layer networks in the classification task with cross-entropy loss. **Left:** Change of empirical NTK[1] ($\|\hat{\Theta}_t - \hat{\Theta}_0\|_F$) is negligible before fitting the training data. NTK changes drastically after overfitting, implying that the delayed generalization might be caused by delayed a transitioning from kernel to rich regime. **Middle:** Controlling the scale of initialization, as a means of controlling the rate of feature learning, can mitigate grokking, to the point of completely eliminating the gap between train and test curves. α denotes scale multiplied by θ_0, the initial weights according to default PyTorch initialization (He et al., 2015). The dashed lines indicate train set statistics, and the solid lines correspond to the test set. **Right:** Empirical evaluations support a sample complexity of $\tilde{\mathcal{O}}(p^{5/3})$ on the classification task with cross-entropy loss. More details in Section 4.

– to a vector in $\mathbb{R}^p$. We use the form

$$f(\theta; (e_a, e_b)) = V\left(W(e_a, e_b)\right)^{\odot 2},$$

where the network weights are $W \in \mathbb{R}^{h\times 2p}$, $V \in \mathbb{R}^{p\times h}$, which we collect into a flattened vector $\theta \in \mathbb{R}^{3hp}$ where $a^{\odot 2}$ denotes the element-wise square of the vector a. The width of the hidden layer, h, will be set later.

We use $[p]$ to denote the set $\{1, \dots, p\}$. For any nonempty set $\mathcal{X}$, a symmetric function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a positive semi-definite kernel (p.s.d) kernel on $\mathcal{X}$ if for any $n \in \mathbb{N}$, any $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n \in \mathcal{X}$ and $\lambda_1, \dots, \lambda_n \in \mathbb{R}$, it holds that $\sum_{i=1}^n \sum_{j=1}^n \lambda_i \lambda_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0$.

In classification, we train f with cross-entropy loss to identify the c such that $a+b \equiv c \pmod p$: that is, we try to map $\boldsymbol{x} = (e_a, e_b)$ to $e_{a+b(\pmod p)}$. Letting $\mathcal{Z} = \{e_i : i \in \mathbb{Z}_p\}$, the set of all possible inputs is $\mathcal{X} = \mathcal{Z} \times \mathcal{Z}$ and outputs is $\mathcal{Y} = \mathcal{Z}$; there are $N = p^2$ distinct data points.

In regression, we instead try to map $x = (e_a, e_b, e_c)$ to the scalar $y = p\,\mathbf{1}(a + b \equiv c \pmod p)$, where $\mathbf{1}$ is the 0-1 indicator function.[2] Here $\mathcal{X} = (\mathcal{Z})^3$ and $\mathcal{Y} = \{0, p\}$, and $N = p^3$; we train the model $g(\theta; \boldsymbol{x}) = \langle e_c, f(\theta; (e_a, e_b))\rangle$ to minimize the square loss.

In either setting, we use $\mathcal{D}$ to denote the distribution over $\mathcal{X} \times \mathcal{Y}$ which is uniform over all N possible input-output pairs, while $\mathcal{D}_{\text{train}} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ is the sequence of training points, of size n. Using ψ to denote the predictor (g for regression, f for classification), we train the model with gradient descent on the possibly regularized objective

$$\mathcal{L}_\lambda(\psi, \theta, \mathcal{D}_{\text{train}}) \triangleq \frac{1}{n} \sum_{(\boldsymbol{x},y)\in\mathcal{D}_{\text{train}}} \ell(\psi(\theta; \boldsymbol{x}), y) + \frac{\lambda}{2} \|\theta\|, \tag{2.1}$$

where ℓ denotes either square (ℓ_2) or cross-entropy loss, $\lambda \geq 0$ controls the strength of regularization, and $\|\cdot\|$ is a parameter norm to be specified later. We define the population loss of a predictor ψ with a loss ℓ as $\mathcal{L}_\ell(\psi) = \mathbb{E}_{x,y\sim\mathcal{D}}\, \ell(\psi(x), y)$. We also define $\psi_{\mathbf{0}}$ to be the predictor which returns zero on any input (e.g. set $V = 0$).

Definition 2.1. A deterministic supervised *learning algorithm* $\mathcal{A}$ is a mapping from a sequence of training data, $\mathcal{D}_{\text{train}} \in (\mathcal{X} \times \mathcal{Y})^n$, to a hypothesis $\mathcal{A}(\mathcal{D}_{\text{train}}) : \mathcal{X} \to \mathcal{Y}$. The algorithm $\mathcal{A}$ could also be randomized, in which case the output $\mathcal{A}(\mathcal{D}_{\text{train}})$ is a distribution on hypotheses. Two randomized algorithms $\mathcal{A}$ and $\mathcal{A}'$ are the same if for any input, their outputs have the same distribution in function space, written as $\mathcal{A}(\mathcal{D}_{\text{train}}) \overset{d}{=} \mathcal{A}'(\mathcal{D}_{\text{train}})$.

Definition 2.2 (Equivariant Algorithms)**.** A learning algorithm is *equivariant* under group $\mathcal{G}_\mathcal{X}$ (or $\mathcal{G}_\mathcal{X}$-equivariant) if for any dataset $\mathcal{D}_{\text{train}} \in (\mathcal{X} \times \mathcal{Y})^n$ and for all $T \in \mathcal{G}_\mathcal{X}$ and $\boldsymbol{x} \in \mathcal{X}$, it holds that $\mathcal{A}(\{(T(\boldsymbol{x}_i), y_i)\}_{i=1}^n)(T(\boldsymbol{x})) = [\mathcal{A}(\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n)](\boldsymbol{x})$.[3]

[1] $\hat{\Theta}_t$ is the NTK on the training data, using the parameter at step t: $\hat{\Theta}_t \triangleq [J_\theta f(\theta_t; \mathcal{X}_{\text{train}})][J_\theta f(\theta_t; \mathcal{X}_{\text{train}})]^\top$.

[2] This scaling implies that the predictor $\psi_{\mathbf{0}}(\cdot) = 0$ has population square loss p; it is the scaling that allows bounded $\|\theta\|_\infty$.

[3] For randomized algorithms, the condition becomes $\mathcal{A}(\{(T(\boldsymbol{x}_i), y_i)\}_{i=1}^n) \circ T \overset{d}{=} \mathcal{A}(\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n)$ – stronger than $\forall \boldsymbol{x} \in \mathcal{X}, \mathcal{A}(\{(T(\boldsymbol{x}_i), y_i)\}_{i=1}^n)(T(\boldsymbol{x})) \overset{d}{=} [\mathcal{A}(\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n)](\boldsymbol{x})$.

3. Regression Task

We will first present our theoretical analysis of the grokking phenomenon on modular arithmetic with one-hidden-layer quadratic networks. The regression task, although perhaps a somewhat less natural way to model modular arithmetic, admits some useful theoretical tools.

A recent line of work on the *neural tangent kernel* (NTK) framework (Jacot et al., 2018; Arora et al., 2019b; Lee et al., 2019; Chizat et al., 2019; Yang & Hu, 2021) has shown that with typical initialization schemes, gradient descent in over-parameterized neural networks locally approximates the behavior of a kernel model using the empirical neural tangent kernel (eNTK) $K_\theta(\boldsymbol{x}, \boldsymbol{x}') \triangleq \nabla g(\theta; \boldsymbol{x}) \nabla g(\theta; \boldsymbol{x}')^\mathsf{T}$. In the "kernel regime," the change in θ over the course of gradient descent does not substantially change the eNTK K_θ, and hence the neural network behaves similarly to a kernel predictor trained with K_{θ_0}. With square loss, as in the regression task at hand, these kernel predictors follow a particularly simple optimization path for which a closed form (corresponding to kernel regression) is available.

For networks of finite width (and in certain infinite-width cases), the eNTK will stay roughly constant and the network will closely track the kernel model for the first phase of optimization, but it will eventually depart. Thus, bounds on kernel models are informative about deep networks in the first part of optimization.

3.1. Kernel Regime

In the first phase of grokking, the model overfits to the training data, achieving very low training error but retaining high error on test points. We prove in Appendix B that kernel regression with the (empirical) neural tangent kernel of our quadratic network achieves zero training error as long as the network width h is $\tilde{\Omega}(n/p)$. Thus, networks trained with gradient descent can achieve zero training error when, for instance, $n = \tilde{\Theta}(p^2)$ and $h = \tilde{\Theta}(p)$.

The remainder of this section shows that for any permutation-equivariant kernel model (and hence networks trained by gradient descent near initialization), generalization is possible only when training on $n = \Omega(p^3)$ samples, i.e. the portion of all possible data points is $\frac{n}{N} = \Omega(1)$.

Definition 3.1. For $\mathcal{Y} \subseteq \mathbb{R}$, we say a learning algorithm $\mathcal{A}$ is a *kernel method* if it first picks a (potentially random) positive semi-definite kernel K on $\mathcal{X}$ before seeing the data,[4] and then outputs some hypothesis h based on $\mathcal{D}_{\text{train}}$ such that there exist $\{\lambda_i\}_{i=1}^n \in \mathbb{R}$ for which $h(\cdot) = \sum_{i=1}^n K(\cdot, \boldsymbol{x}_i)\lambda_i$.

The key component of our analysis is the permutation equivariance of learning modular addition in this setting. We first define the permutation group on the modular addition data:

Definition 3.2. Let $\mathbb{S}_p$ denote the set of all permutations on $\mathbb{Z}_p$. We define the *permutation group* on $\mathcal{X}$ as the group

$$\left\{(e_a, e_b, e_c) \mapsto (e_{\sigma_1(a)}, e_{\sigma_2(b)}, e_{\sigma_3(c)}) : \sigma_1, \sigma_2, \sigma_3 \in \mathbb{S}_p\right\},$$

with the operation being composition of each permutation.

The following lemma establishes that our learning algorithm is permutation-equivariant, meaning that it is equivariant under the permutation group in the sense of Li et al. (2020). Note that this result applies to the actual process of gradient descent on our neural network, not simply to its NTK approximation. (This result is not particularly specific to the architecture defined in Section 2; it holds broadly.)

Lemma 3.3 (Permuation Invariance in Regression)**.** *Gradient descent training of $g(\theta; x)$ with ℓ_2 loss and i.i.d. randomly initialized parameters in (2.1) on the modular addition problem is an equivariant learning algorithm with respect to the permutation group of Definition 3.2.*

More details and the full proof are in Appendix C.

Roughly speaking, this indicates that learning the modular addition task on this setup is the exactly same difficulty as learning any permuted version of the dataset. Following the same argument, we can show that the kernel method corresponding to the neural nets in the early phase of training is also permutation-equivariant. Further, as the distribution $\mathcal{D}$ is uniform and thus invariant under permutation, the above Lemma 3.3 shows that the difficulty of learning the original ground-truth is the same as simultaneously learning all the permuted versions of the ground-truths, which turns out to be difficult for any kernel method. The following theorem formalizes this proof idea and establishes that any such kernel predictor needs at least $n = \Omega(p^3)$, or equivalently $\frac{n}{N} = \Omega(1)$, to generalize on the modular addition task. A full proof of this result is deferred to Appendix D.

Theorem 3.4 (Lower Bound)**.** *There exists a constant $C > 0$ such that for any $p \geq 2$, training data size $n < Cp^3$, and any permutation-equivariant kernel method $\mathcal{A}$, it holds that*

$$\mathop{\mathbb{E}}_{(\boldsymbol{x}_i, y_i)_{i=1}^n \sim \mathcal{D}^n} \mathop{\mathbb{E}}_{\mathcal{A}} \mathcal{L}_{\ell_2}\left(\mathcal{A}\left(\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n\right)\right) \geq \frac{1}{2}\,\mathcal{L}_{\ell_2}\left(h_{\mathbf{0}}\right) = \frac{p}{2},$$

where $\mathbb{E}_{\mathcal{A}}$ is over the randomness in algorithm $\mathcal{A}$.

All permutation-invariant kernel methods have error at least half as large as that of the trivial all-zero predictor, unless we see a constant fraction of all possible data points ($\frac{n}{N} \geq C$).

Proposition 3.7 is in fact a special case of a more general Theorem that lower bounds the population ℓ_2 loss of learning modular addition with m-summands using permutation-equivariant kernels. showing that poor generalization in kernel regime in this setting is always inevitable. We present

[4] Our definition of kernel methods does not cover learning algorithms that choose the kernel based on the training data.

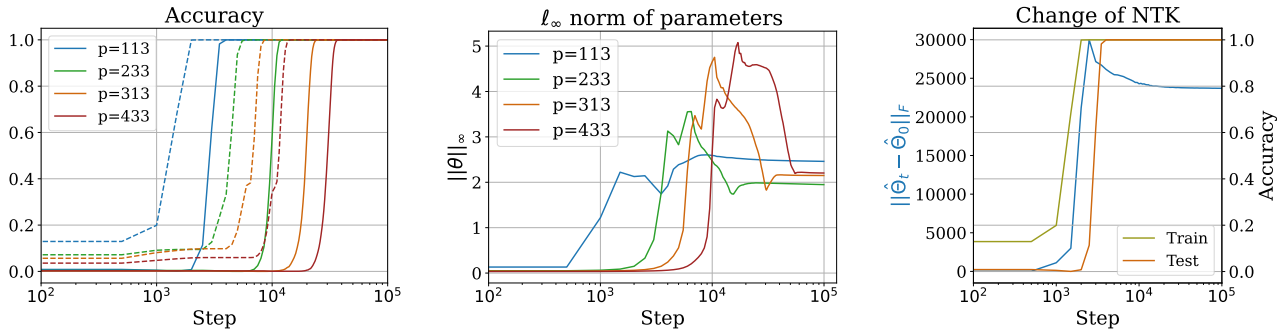

Figure 2: Empirical investigation of training the network with gradient descent on ℓ_2 loss and ℓ_∞ regularization. As problem dimension (p) grows, generalization requires more training steps, but the ℓ_∞ norm of parameters stay constant with respect to p. The dashed lines in the left figure indicate train set statistics, and the solid line correspond to the test set.

an informal version of this result below and refer the reader to Corollary D.7 for the formal version.

Theorem 3.5 (Informal)**.** *There exists a constant $C > 0$ such that for any $p \geq 2$, training data size $n < Cp^m$ and any permutateion equivariant kernel method $\mathcal{A}$ with kernel $K : [p]^m \times [p]^m \to \mathbb{R}$, it holds that*

$$\mathop{\mathbb{E}}_{(\boldsymbol{x}_i, y_i)_{i=1}^n} \mathop{\mathbb{E}}_{\mathcal{A}} \mathcal{L}_{\ell_2}\left(\mathcal{A}\left(\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n\right)\right) \geq \frac{1}{2}\,\mathcal{L}_{\ell_2}\left(h_{\mathbf{0}}\right)$$

where $x_1, x_2, \cdots, x_t \sim \mathrm{Unif}\left([p]^m\right)$ and y_i denotes the corresponding label for $i \in [t]$. In other words, if $n < (1 - \Omega(1))p^m$, then the expected population ℓ_2 loss is at least $\Omega(p^{m-1})$, which is of the same magnitude as the trivial all-zero predictor.

Poor population error is thus inevitable for models which are well-approximated by a permutation-equivariant kernel; since our quadratic network can also achieve zero training error in the kernel regime, this strongly suggests drastic overfitting in early training. Fortunately, however, gradient descent on finite networks will eventually leave the kernel regime and begin learning features.

3.2. Rich Regime

While the transient behavior of gradient descent after leaving the kernel regime may be complicated, it is often the case that the limiting behavior can be better-understood based on analyses of implicit bias. Motivated by this, we present a generalization analysis of networks that achieve zero training error (as we expect in the long-term optimization limit) with bounded $\|\theta\|_\infty$. This choice of regularizer is motivated by recent work of Xie & Li (2024), who show that full-batch AdamW can only converge to the KKT points of the ℓ_∞ norm constrained optimization problem, $\min_{\|\theta\|_\infty \leq \frac{1}{\lambda}} \mathcal{L}(\theta)$ where $\mathcal{L}$ is the training loss and λ is the weight-decay coefficient. We will discuss the feasibility of this assumption more afterwards.

Theorem 3.6 (Upper Bound)**.** *For any width $h \geq 8p$, with probability at least $1 - \delta$ over the randomness of training dataset $\mathcal{D}_{\mathrm{train}}$ of size n, define the set of interpolating solutions as $\tilde{\Theta}^* = \{\theta \mid \mathcal{L}(g, \theta, \mathcal{D}_{\mathrm{train}}) = 0\}$. For any interpolating solution $\tilde{\theta}^* \in \tilde{\Theta}^*$ with small ℓ_∞ norm,* i.e.*, satisfying that $\|\theta^*\|_\infty = \mathcal{O}(\min_{\theta \in \tilde{\Theta}^*} \|\theta\|_\infty)$, it holds that*

$$\mathcal{L}_{\ell_2}(g(\theta; \cdot)) = \mathcal{O}\left(\frac{p^2}{n}\left(\log^3 n + \frac{1}{p}\log\frac{1}{\delta}\right)\right) \cdot \mathcal{L}_{\ell_2}\left(h_{\mathbf{0}}\right).$$

Comparing Theorem 3.6 to Theorem 3.4, we see that when $n = \tilde{\omega}(p^2)$ and $h = \Theta(p)$, there is a strict separation between generalization in the kernel and rich regimes. Note that the classifier $\psi(a, b) \triangleq \arg\max_c f_c(\theta^*; (e_a, e_b))$ has population error rate at most $\frac{2}{p}\mathcal{L}_{\ell_2}(g(\theta; \cdot))$, as shown in Proposition E.6, and thus goes to zero when $n = \tilde{\omega}(p^2)$.

The proof consists of showing all networks with small training error and small ℓ_∞ norms generalize (Proposition 3.7), and at least one such network exists (Proposition 3.8).

Proposition 3.7. *Choose $R, \delta > 0$ to be positive constants. For any $\mathcal{D}_{\mathrm{train}}$ of size n and $\theta^* \in \{\theta = (W, V) : \mathcal{L}(g, \theta, \mathcal{D}_{\mathrm{train}}) = 0 \wedge \|\theta\|_\infty \leq R\}$, there exists a positive constant $C > 0$ such that with probability at least $1 - \delta$ over the choice of $\mathcal{D}_{\mathrm{train}}$,*

$$\mathcal{L}_{\ell_2}(g(\theta^*; \cdot)) \leq \frac{CR^6h^2}{n}\left(p\log^3 n + \log\frac{1}{\delta}\right).$$

Proof sketch. We bound the Rademacher complexity of the set of networks with small ℓ_∞ weights, and then apply Theorem E.5, due to Srebro et al. (2010), which gives an "optimistic" bound on the excess risk of smooth loss functions. Details in Appendix E. □

Proposition 3.8. *Let the set of models with zero population loss be $\Theta^* \triangleq \{\theta \mid \mathcal{L}_{\ell_2}(g(\theta; \cdot)) = 0\}$. For any $p \geq 2$ and $h \geq 8p$, Θ^* is nonempty and $\min_{\theta \in \Theta^*} \|\theta\|_\infty \leq \left\lfloor \frac{h}{8p} \right\rfloor^{-\frac{1}{3}}$.*

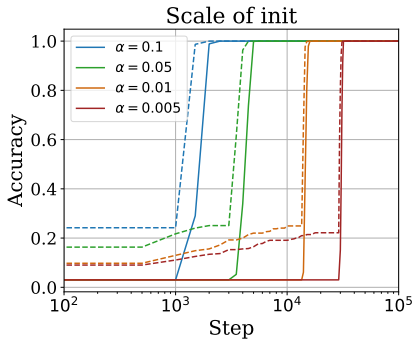

Figure 3: Train (dashed) and test (solid) accuracy while training in the regression setting with various initialization scales α. Shrinking the scale of initialization can mitigate grokking in the classification task, eventually eliminating of the gap between train and test accuracies (at the cost of slower improvement in each).

Proof sketch. The main difficulty is a manual Fourier-based construction of a zero-loss solution with $h = 8p$ and ℓ_∞ norm at most one; this is inspired by similar constructions of Gromov (2023) and Morwani et al. (2023) results of Nanda et al. (2023). Once we have that, because our model is 3-homogeneous in its parameters, we can easily reduce the ℓ_∞ norm by duplicating neurons, without changing the input-output function. Details in Appendix F. □

We know that an interpolating solution exists with small ℓ_∞ norm, and that any such solution will generalize. Does gradient descent find one? In Figure 2, we empirically evaluate the ℓ_∞ norm of weights learned by running gradient descent on the regression task, with varying p, and a very small ℓ_∞ regularizer.[5] Consistent with our manual construction of weights, the ℓ_∞ norm of the network weights does not grow with the problem dimension p. This supports the applicability of Theorem 3.6 and a far better sample complexity compared to the kernel regime.

3.3. Implications for Kernel and Rich Regimes

Networks in the kernel regime will provably fail to generalize as long as they do not have access to nearly all the points in the dataset (Theorem 3.4), although they can achieve zero training error (Appendix B). We further established that if, once the network has departed the kernel regime, it manages to achieve zero training error with bounded $\|\theta\|_\infty$, it will generalize with only a $\frac{n}{N} = \tilde{\omega}(1/p)$ portion of the overall dataset (Theorem 3.6). We also proved that such networks exist (Proposition 3.8), and demonstrated that gradient descent can find them with a small amount of explicit regularization (Figure 2).

We do not yet know, though, *when* the network will leave the kernel regime. It is well-known that the rate of transitioning between the kernel and rich regimes in the trajectory of gradient descent can be controlled by the *scale of initialization* (Chizat et al., 2019; Moroshko et al., 2020; Geiger et al., 2020; Lyu et al., 2023; Telgarsky, 2022). Smaller scales of initialization, e.g. smaller variance scales for weight initializations in the scheme of He et al. (2015), lead training to escape the kernel regime faster; larger scales prolong the time spent in kernel regime. However, this comes at a cost: training becomes exponentially more difficult as the scale of initialization decreases (Moroshko et al., 2020). Figure 3 confirms that in our setting, using a very small weight initialization can substantially mitigate the grokking effect, at the cost of much slower optimization.

In Figure 2, we evaluate the change of the empirical NTK during training, by computing the difference in eNTK matrices through training. To make this empirical investigation computationally feasible, we evaluated the eNTK approximation of Mohamadi et al. (2023) on 20,000 random data points, similar to previous schemes (Fort et al., 2020; Wei et al., 2020; Mohamadi et al., 2023). We see that the change in empirical NTK is orders of magnitude larger after overfitting the training set, implying that most feature learning happens only later, supporting our hypothesis that the initial overfitting occurs roughly in the kernel regime.

4. Classification Task

We now move onto the multi-class classification setting established in Section 2, where we train with cross-entropy loss. Similar to the regression problem, we first analyze the early stage of training where the network operates like a kernel machine, and then move onto the rich regime and focus on the weights learned through margin-maximization implicit bias of gradient descent.

4.1. Kernel Regime

In the multi-class setting, the output is p-dimensional, and thus the eNTK for each pair of points is a $p \times p$ psd matrix (see Alvarez et al., 2012). Our notion of kernel methods (Definition 3.1) still holds, and the main lower bound result is also similar to that in the regression case: kernel methods must see a constant fraction of data before learning.

Theorem 4.1. *There exists constant $C > 0$ such that for any training data size $n < Cp^2$ and any kernel method $\mathcal{A}$ which is also permutation-equivariant, it holds that*

$$\mathop{\mathbb{E}}_{(\boldsymbol{x}_i, y_i)_{i=1}^n \sim \mathcal{D}^n} \mathop{\mathbb{E}}_{\mathcal{A}} \mathcal{L}_{\ell_2}\left(\mathcal{A}\left(\{\boldsymbol{x}_i, y_i\}_{i=1}^n\right)\right) \geq \frac{1}{2}\, \mathcal{L}_{\ell_2}\left(h_{\mathbf{0}}\right),$$

where $\mathbb{E}_{\mathcal{A}}$ is the mean over the randomness in algorithm $\mathcal{A}$.

[5] We used a regularization weight of 10^{-4}. Without explicit regularization, a small number of "unimportant" network weights do grow with p, but we believe this phenomenon is not important to the overall behavior of gradient descent on this task.

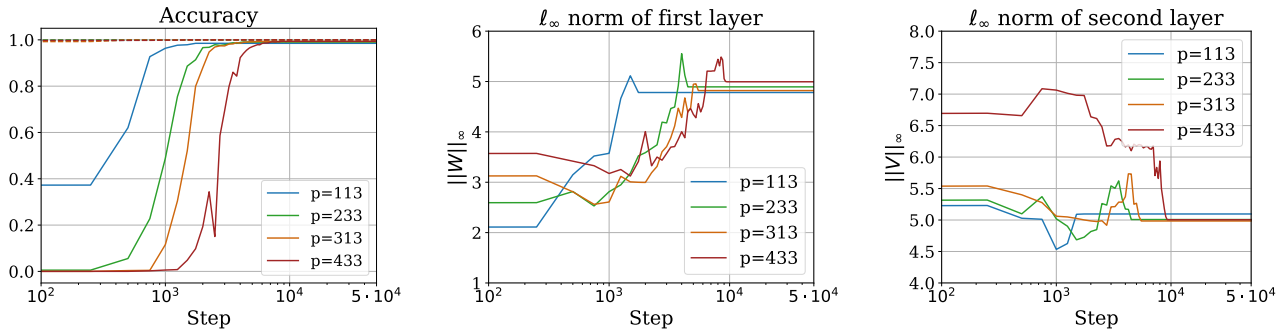

Figure 4: Empirical investigation into training the network with gradient descent on cross-entropy loss and ℓ_∞ regularization. As problem dimension (p) grows, generalization requires training longer, but the ℓ_∞ norms of the weights do not grow. The **dashed lines** in the left panel indicate **train set statistics**, and the **solid lines correspond to the test set**.

Intuitively, the classification setting is not very different from the regression setting. They share the same goal and the amount of knowledge contained in each data in the classification setting is exactly equal to p data in the regression setting, where these p data share the same first two coordinates and only differ in the last coordinate.

More formally, we can define the following correspondence: given one data point in classification $(\boldsymbol{x}, y) \in \mathbb{R}^{2p} \times \mathbb{R}^p$, we can view it as p data points in regression, $\{((\boldsymbol{x}, e_i), y_i)\}_{i=1}^p$, denoted by $F(\boldsymbol{x}, y)$. Similarly, any function ψ mapping from $\boldsymbol{x} = \{(e_i, e_j)\}$ to $\mathbb{R}^p$ can be viewed as a function mapping $\{(e_i, e_j, e_k)\}$ to $\mathbb{R}$ by defining $\psi'(\boldsymbol{x}, e_k) = [\psi(\boldsymbol{x})]_k$. Moreover, these two functions ψ and ψ' share the same population ℓ_2 loss. Under this view, matrix-valued kernel learning with n classification data points is exactly the same as scalar-valued kernel learning with np regression data points.

The only obstacle to directly applying Theorem 3.4 is that the data distribution is different. For the regression setting, each data is sampled independently and uniformly, and the number of data points can be any integer; in the classification setting, the data points are sampled in independent groups and the number of data must be a multiple of p. However, it is easy to see this new distribution is still invariant under permutations as in Definition 3.2. Thus, we can directly apply Theorem 3.4 on this new distribution to get Theorem 4.1.

Therefore, for networks operating in the kernel regime, generalization to unseen data in ℓ_2 loss is impossible unless $n/N = \Omega(1)$, i.e. we have observed a constant fraction of all the $N = p^2$ possible data points.

It is worth emphasizing, however, that a large ℓ_2 lower bound does not necessarily imply large classification error, or large cross-entropy; the proof technique is difficult to generalize to those losses.

4.2. Rich Regime

To analyze the dynamics of the network in the rich regime, we first introduce the notion of margin and elaborate on the margin-maximization implicit bias of gradient descent. For a multi-class classification problem with p classes with fixed network f, the margin of a data point $(\boldsymbol{x}, y)$ is

$$q(\theta; x, y) \triangleq (f(\theta; x))_y - \max_{y' \neq y} (f(\theta; x))_{y'}.$$

The margin for a dataset $\mathcal{D}$ is defined as the minimum margin of all points on the dataset:

$$q_{\min}(\theta) \triangleq \min_{(x,y) \in \mathcal{D}} q(\theta; x, y).$$

When the network f is homogeneous with respect to its parameters (as is the case in our setup), one can observe that as long as θ linearly separates the dataset $\mathcal{D}$ such that $q_{\min}(\theta) > 0$, it is possible to arbitrary scale the minimum margin, through scaling the parameters of the network. Hence, in such homogenous networks, one is usually concerned with a normalized margin $q_{\min}(\theta / \|\theta\|)$ according to some norm.

Lyu & Li (2020) proved that gradient descent on homogeneous models with the cross-entropy (or similar) losses, in the absence of explicit regularization, maximizes the normalized margin. Specifically, although $\|\theta\|_2 \to \infty$ as $t \to \infty$, $\theta / \|\theta\|_2$ converges to a solution (or more generally, a KKT point) of the following problem when one exists:

$$\min \frac{1}{2} \|\theta\|_2^2 \quad \text{s.t.} \quad q_{\min}(\theta) \geq 1. \tag{4.1}$$

To establish our results in the classification setting, we borrow the following theorem from Wei et al. (2020), who prove that when the strength of the regularization used in (2.1) is small enough, the maximum normalized margin of the regularized loss converges to that of the unregularized loss.

Proposition 4.2 (Wei et al., 2020, Theorem 4.1)**.** *Consider a positively homogeneous function f with respect to parameters θ and a dataset $\mathcal{D}_{\text{train}}$ linearly separable with f. Let $\|\cdot\|$ be any norm. Let γ^* be the maximum normalized margin of the unregularized loss and γ^λ be the maximum normalized margin of the regularized loss with strength λ. As $\lambda \to 0$, $\gamma^\lambda \to \gamma^*$.*

That is, if we use a small enough regularization weight λ with any norm in (2.1), we will obtain approximately the same solution as the unregularized problem. This justifies again applying a small ℓ_∞ regularizer.

We thus now present a generalization bound for one-hidden-layer networks with quadratic activation with bounded $\|\theta\|_\infty$. Our proof is based on the PAC-Bayesian framework (McAllester, 2003), specifically using Lemma 1 of Neyshabur et al. (2018). This result provides a margin-based high probability generalization bound for any predictor based on the *margin loss*, which counts a prediction as correct only if it predicts with a margin at least γ:

$$L_\gamma\left(f,\theta,\mathcal{D}\right) \triangleq \mathbb{P}_{(\boldsymbol{x},y)\in\mathcal{D}}\left[\, q(\theta,\boldsymbol{x},y) > \gamma \,\right]. \tag{4.2}$$

L_0 is simply the misclassification rate, or 0-1 error.

Theorem 4.3 (Informal)**.** *Choose $R, M, \delta > 0$ to be positive constants such that $h \leq Mp$. For any $\mathcal{D}_{\text{train}}$ of size n and θ with $\|\theta\|_\infty \leq R$, it holds with probability at least $1-\delta$ over the randomness of $\mathcal{D}_{\text{train}}$ that*

$$L_0(f,\theta,\mathcal{D}) \leq L_p(f,\theta,\mathcal{D}_{train}) + \widetilde{\mathcal{O}}\left(\sqrt{\frac{p^{5/3}}{n}}\right). \tag{4.3}$$

Theorem 4.3 implies that as long as gradient descent with cross-entropy loss finds a solution whose margin is larger than p, a sample complexity of $\tilde{\mathcal{O}}(p^{5/3})$ on the modular addition problem in multi-class classification setting is guaranteed. This confirms and explains previous observations (e.g. Power et al., 2022; Gromov, 2023; Nanda et al., 2023; Liu et al., 2022b) on the minimum threshold for the fraction of data used to achieve generalization. In combination with Theorem 4.2 of Lyu & Li (2020), this shows that with enough training data, gradient descent will eventually find a generalizing solution for this setting of the modular addition problem.

The construction of Appendix F achieves a margin of p with constant $\|\theta\|_\infty$, so a network satisfying the conditions of Theorem 4.3 exists. In terms of whether gradient descent finds a model satisfying these conditions: Figure 4 empirically evaluates the ℓ_∞ norm of the learned weights through gradient descent on the classification task, with tiny ℓ_∞ regularization (weight of 10^{-20}), across different values of p. Again, it can be seen that the ℓ_∞ norm of the weights of the network do not grow with the problem dimension, supporting the main assumption of Theorem 4.3. These experiment use $n = 2\,p^{5/3}$ data points for each p, with a learning rate of 10.

4.3. Implications of Results for Kernel and Rich Regimes

Similar to our results for the regression task, we have established that a sample complexity gap between kernel and rich regime exists when learning a one-hidden-layer neural network with gradient descent on modular addition modeled as a multi-class classification task. Our results imply that as long as the max-margin implicit bias of gradient descent on exponential-type loss functions drives the net to leave the kernel regime and $\hat{\Omega}(p^{5/3})$ samples are used for training, generalization to the whole population is guaranteed.

Similar to the regression setting, Figure 1 presents empirical evidence on the impact of kernel regime on the poor generalization capabilities in the early phase of training by showing the minimal change of empirical NTK until after overfitting in the classification setting.

Through changing the scale of initialization, in Figure 1 we demonstrate that by increase or decrease the scale of initialization and controlling the rate of feature learning in the classification task one can intensify or mitigate grokking such that the gap between overfitting and generalization becomes larger or smaller. This further supports that as long as the network lies in the kernel regime, generalization without having access to the full dataset is impossible.

5. Grokking Modular Addition in Transformers

Figure 5 suggests that, similarly to the one-hidden-layer network, grokking in the original Transformer studied by Power et al. (2022) might be explainable through the same mechanism. In fact, Lemma 3.3, albeit with small modifications to incorporate the embedding layer, applies to this one-layer Transformer as well. Unfortunately, however, AdamW is not known to initially follow a kernel method the way gradient descent is, and so concrete rigorous claims about this setup would rely on understanding this behavior.

6. Additional Related Work

Limitations imposed by Equivariance of Learning Algorithms. Abbe & Boix-Adsera (2022) study the impact of equivariance of the training algorithms on the efficiency of learning different functions on different networks. In particular, they consider two main setups: **a)** learning with FCNs using noisy GD with clipped gradients throughout the training, and **b)** learning a specific instance of the modular addition task ($p = 2$ with noisy inputs) with FCNs using

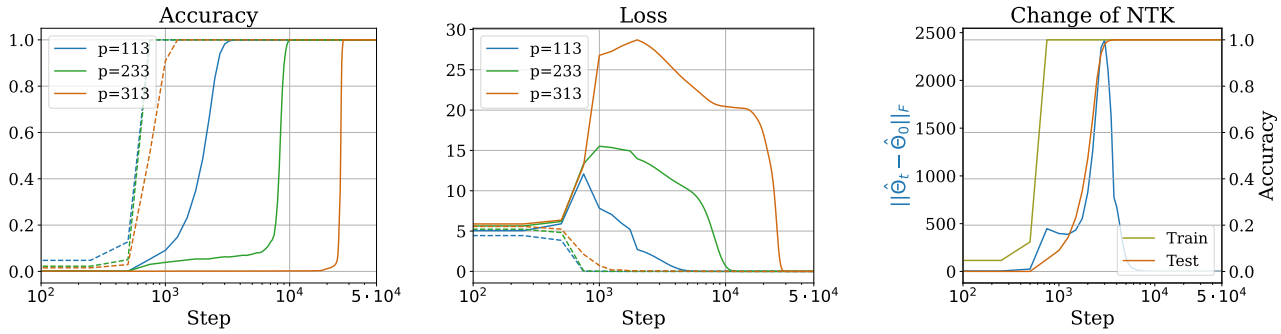

Figure 5: Empirical investigation of training a one-layer transformer with AdamW using cross-entropy loss on the modular addition problem with various ps. Change of eNTK up to the point of fitting the training set is negligible. The eNTK has a drastic change only after fitting the whole training set, implying minimal feature learning until past overfitting. The dashed lines in the middle and left figures indicate train set statistics, and the solid lines correspond to the test set.

SGD. Although their approach in studying lower bounds for efficient learning shares some high level similarity with ours in using equivariance of the training algorithm, the settings considered are significantly differs from ours. Moreover, in Appendix D.3 we present a novel abstract framework for analyzing lower bounds on population ℓ_2 loss for general function classes. Malach & Shalev-Shwartz (2022) present another technique in analysis of population loss lower bounds, which shares some high-level similarity with our framework, albeit their analysis is more restrictive on function classes. Ng (2004) also discusses rotational equivariance of many learning algorithms and presents a general lower bound on the 0-1 population error of such algorithms in the general case.

Margin Maximization as the Late Phase Implicit Bias. Morwani et al. (2023) present an analytical solution for the max-margin solution of learning modular addition in a classification setting similar to Section 4 when using all of the dataset in training the network. Similar to our analysis of the rich regime in this setting, they face difficulties in proving results under the assumption of bounded ℓ_2 norm for weights of the network, and assume an $\ell_{2,3}$ bound instead.

7. Discussion

In this work, we studied the phenomenon of grokking in learning modular addition with gradient descent on one-hidden-layer networks, modeled as regression or classification. We showed that learning modular addition as presented is fundamentally a difficult task for kernel models (for example neural networks in kernel regime) due to the inherent symmetry and permutation-equivariance of the task. We theoretically established this difficulty by presenting sample complexity lower bounds of order of constant fraction of the whole dataset. We further showed that networks satisfying certain conditions generalize far better than those in the kernel regime, that such networks exist, and showed empirical evidence that simple regularized gradient descent can eventually find them, once it escapes the kernel regime. These results, in combination, attempt to address ***why*** grokking is observed when learning modular addition. We provide strong evidence to the hypothesis (Lyu et al., 2023; Kumar et al., 2023) that, on this important problem, it is indeed due to a separation between kernel and non-kernel behavior of gradient descent.

Future Work. We have guaranteed large ℓ_2 population loss for networks in kernel regime on both regression and classification settings. It is possible, however, for networks to have arbitrarily high ℓ_2 loss while having perfect classification accuracy. We conjecture that impossibility results for accuracy-based generalization may also be possible based on permutation equivariance in the early phase of training, but leave it as future work. Moreover, we only study the cause of grokking in these settings, but do not analyze possible training techniques to enable quick generalization on this task. Although we are able to eliminate grokking through changing the scale of initialization, doing so actually slows down the time to final generalization; finding practical methods to enable quick generalization would be more useful.

Impact Statement

This paper presents work whose goal is to advance the field of machine learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

Acknowledgments

We would like to thank Kaifeng Lyu and Wei Hu for helpful discussions. This work was enabled in part by support

provided by the Natural Sciences and Engineering Research Council of Canada, the Canada CIFAR AI Chairs program, Advanced Research Computing at the University of British Columbia, Calcul Québec, the BC DRI Group, and the Digital Research Alliance of Canada.

References

Abbe, E. and Boix-Adsera, E. On the non-universality of deep learning: quantifying the cost of symmetry, 2022.

Alvarez, M. A., Rosasco, L., Lawrence, N. D., et al. Kernels for vector-valued functions: A review. *Foundations and Trends® in Machine Learning*, 4(3):195–266, 2012.

Ansel, J., Yang, E., He, H., Gimelshein, N., Jain, A., Voznesensky, M., Bao, B., Bell, P., Berard, D., Burovski, E., Chauhan, G., Chourdia, A., Constable, W., Desmaison, A., DeVito, Z., Ellison, E., Feng, W., Gong, J., Gschwind, M., Hirsh, B., Huang, S., Kalambarkar, K., Kirsch, L., Lazos, M., Lezcano, M., Liang, Y., Liang, J., Lu, Y., Luk, C. K., Maher, B., Pan, Y., Puhrsch, C., Reso, M., Saroufim, M., Siraichi, M. Y., Suk, H., Zhang, S., Suo, M., Tillet, P., Zhao, X., Wang, E., Zhou, K., Zou, R., Wang, X., Mathews, A., Wen, W., Chanan, G., Wu, P., and Chintala, S. PyTorch 2: Faster machine learning through dynamic Python bytecode transformation and graph compilation. In *Architectural Support for Programming Languages and Operating Systems*, 2024.

Arora, S., Cohen, N., Hu, W., and Luo, Y. Implicit regularization in deep matrix factorization, 2019a.

Arora, S., Du, S. S., Hu, W., Li, Z., Salakhutdinov, R. R., and Wang, R. On exact computation with an infinitely wide neural net. *NeurIPS*, 32, 2019b.

Awasthi, P., Frank, N., and Mohri, M. On the rademacher complexity of linear hypothesis sets, 2020.

Barak, B., Edelman, B., Goel, S., Kakade, S., Malach, E., and Zhang, C. Hidden progress in deep learning: SGD learns parities near the computational limit. *NeurIPS*, 35: 21750–21764, 2022.

Bhattamishra, S., Patel, A., Kanade, V., and Blunsom, P. Simplicity bias in transformers and their ability to learn sparse boolean functions, 2023.

Blanc, G., Gupta, N., Valiant, G., and Valiant, P. Implicit regularization for deep neural networks driven by an ornstein-uhlenbeck like process, 2020.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

Charton, F. Can transformers learn the greatest common divisor?, 2023.

Chizat, L., Oyallon, E., and Bach, F. On lazy training in differentiable programming. In *NeurIPS*, 2019.

Courant, R. and Hilbert, D. *Methods of Mathematical Physics*, volume 1. Interscience Publishers, 1953.

Damian, A., Ma, T., and Lee, J. D. Label noise sgd provably prefers flat global minimizers, 2021.

Fort, S., Dziugaite, G. K., Paul, M., Kharaghani, S., Roy, D. M., and Ganguli, S. Deep learning versus kernel learning: an empirical study of loss landscape geometry and the time evolution of the neural tangent kernel, 2020.

Geiger, M., Spigler, S., Jacot, A., and Wyart, M. Disentangling feature and lazy training in deep neural networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2020(11):113301, November 2020. ISSN 1742-5468.

Gromov, A. Grokking modular arithmetic, 2023.

Gunasekar, S., Woodworth, B., Bhojanapalli, S., Neyshabur, B., and Srebro, N. Implicit regularization in matrix factorization, 2017.

Gunasekar, S., Lee, J., Soudry, D., and Srebro, N. Characterizing implicit bias in terms of optimization geometry, 2020.

HaoChen, J. Z., Wei, C., Lee, J. D., and Ma, T. Shape matters: Understanding the implicit bias of the noise covariance, 2020.

He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

Jacot, A., Gabriel, F., and Hongler, C. Neural tangent kernel: Convergence and generalization in neural networks. *NeurIPS*, 31, 2018.

Kumar, T., Bordelon, B., Gershman, S. J., and Pehlevan, C. Grokking as the transition from lazy to rich training dynamics, 2023.

Lee, J., Xiao, L., Schoenholz, S., Bahri, Y., Novak, R., Sohl-Dickstein, J., and Pennington, J. Wide neural networks of any depth evolve as linear models under gradient descent. *NeurIPS*, 32, 2019.

Levi, N., Beck, A., and Bar-Sinai, Y. Grokking in linear estimators – a solvable model that groks without understanding, 2023.

Li, Z., Zhang, Y., and Arora, S. Why are convolutional nets more sample-efficient than fully-connected nets? *arXiv preprint arXiv:2010.08515*, 2020.

Li, Z., Wang, T., and Arora, S. What happens after sgd reaches zero loss? –a mathematical framework, 2022.

Liu, Z., Kitouni, O., Nolte, N. S., Michaud, E., Tegmark, M., and Williams, M. Towards understanding grokking: An effective theory of representation learning. *NeurIPS*, 35:34651–34663, 2022a.

Liu, Z., Michaud, E. J., and Tegmark, M. Omnigrok: Grokking beyond algorithmic data. *arXiv preprint arXiv:2210.01117*, 2022b.

Lyu, K. and Li, J. Gradient descent maximizes the margin of homogeneous neural networks. In *ICLR*, 2020.

Lyu, K., Jin, J., Li, Z., Du, S. S., Lee, J. D., and Hu, W. Dichotomy of early and late phase implicit biases can provably induce grokking, 2023.

Malach, E. and Shalev-Shwartz, S. When hardness of approximation meets hardness of learning. *Journal of Machine Learning Research*, 23(91):1–24, 2022.

McAllester, D. A. Simplified pac-bayesian margin bounds. In *COLT*, 2003.

Mohamadi, M. A., Bae, W., and Sutherland, D. J. A fast, well-founded approximation to the empirical neural tangent kernel. In *International Conference on Machine Learning*, pp. 25061–25081. PMLR, 2023.

Moroshko, E., Gunasekar, S., Woodworth, B., Lee, J. D., Srebro, N., and Soudry, D. Implicit bias in deep linear classification: Initialization scale vs training accuracy, 2020.

Morwani, D., Edelman, B. L., Oncescu, C.-A., Zhao, R., and Kakade, S. Feature emergence via margin maximization: case studies in algebraic tasks, 2023.

Nacson, M. S., Gunasekar, S., Lee, J. D., Srebro, N., and Soudry, D. Lexicographic and depth-sensitive margins in homogeneous and non-homogeneous deep models, 2019.

Nanda, N., Chan, L., Liberum, T., Smith, J., and Steinhardt, J. Progress measures for grokking via mechanistic interpretability. *arXiv preprint arXiv:2301.05217*, 2023.

Neyshabur, B., Bhojanapalli, S., and Srebro, N. A PAC-Bayesian approach to spectrally-normalized margin bounds for neural networks. In *International Conference on Learning Representations*, 2018.

Ng, A. Y. Feature selection, L_1 vs. L_2 regularization, and rotational invariance. In *ICML*, 2004.

Notsawo, P. J. T., Zhou, H., Pezeshki, M., Rish, I., and Dumas, G. Predicting grokking long before it happens: A look into the loss landscape of models which grok, 2023.

Power, A., Burda, Y., Edwards, H., Babuschkin, I., and Misra, V. Grokking: Generalization beyond overfitting on small algorithmic datasets, 2022.

Raab, M. and Steger, A. "Balls into bins" — a simple and tight analysis. In *Randomization and Approximation Techniques in Computer Science*, 1998.

Rubin, N., Seroussi, I., and Ringel, Z. Grokking as a first order phase transition in two layer networks, 2024.

Soudry, D., Hoffer, E., Nacson, M. S., Gunasekar, S., and Srebro, N. The implicit bias of gradient descent on separable data, 2022.

Srebro, N., Sridharan, K., and Tewari, A. Smoothness, low noise and fast rates. *NeurIPS*, 23, 2010.

Telgarsky, M. Feature selection with gradient descent on two-layer networks in low-rotation regimes, 2022.

Thilak, V., Littwin, E., Zhai, S., Saremi, O., Paiss, R., and Susskind, J. The slingshot mechanism: An empirical study of adaptive optimizers and the grokking phenomenon. *arXiv preprint arXiv:2206.04817*, 2022.

Tropp, J. A. An introduction to matrix concentration inequalities. *Foundations and Trends® in Machine Learning*, 8 (1-2):1–230, 2015.

Varma, V., Shah, R., Kenton, Z., Kramár, J., and Kumar, R. Explaining grokking through circuit efficiency, 2023.

Wainwright, M. J. *High-Dimensional Statistics: A Non-Asymptotic Viewpoint*. Cambridge University Press, 2019.

Wei, C., Lee, J. D., Liu, Q., and Ma, T. Regularization matters: Generalization and optimization of neural nets v.s. their induced kernel, 2020.

Xie, S. and Li, Z. Implicit bias of AdamW: ℓ_∞ norm constrained optimization. *arXiv preprint arXiv:2404.04454*, 2024.

Xu, Z., Wang, Y., Frei, S., Vardi, G., and Hu, W. Benign overfitting and grokking in ReLU networks for XOR cluster data, 2023.

Yang, G. and Hu, E. J. Tensor Programs IV: Feature learning in infinite-width neural networks. In *ICML*, 2021.

A. Experimental Setup

In this section, we briefly explain the setup used for our experimental evaluations.

A.1. Regression

We have used vanilla gradient descent with squared less for 100,000 or 200,000 for each experiment. In regression, our learning has been fixed to 1, and the regularization strength has been set to 10^{-4}. The network has been initialized according to He et al. (2015). The amount of data used for training in regression task has been set to $2 \times p^{2.25}$.

A.2. Classification

In all experiments, we have used vanilla gradient descent with cross-entropy loss, for up to 100,000 steps. To accelerate the training with cross-entropy loss, we use the "normalized" GD trick, where the learning rate of each step is scaled by the inverse of the norm of the gradient:

$$\theta_{t+1} = \theta_t - \eta \frac{\nabla_\theta \ell(\theta_t)}{\|\nabla_\theta \ell(\theta_t)\|_2} \tag{A.1}$$

where ℓ denotes the loss function and η denotes the learning rate. The learning rate in the presented experiments was set to 10 and was kept constant during the training. The regularization strength of ℓ_∞ regularizer has been set to 10^{-20}. The network has been initialized according to He et al. (2015). The amount of data used for training in regression task has been set to $2 \times p^{5/3}$.

A.3. Transformer

To train the one-layer transformer we have used full-batch AdamW with a learning rate of $\eta = 10^{-3}$ and a regularization strength of 0.5. For AdamW, we have set $\beta_1 = 0.9$ and $\beta_2 = 0.99$. The network has been initialized according to default PyTorch initialziation (migrated ot JAX). We have used $2 \times p^{5/3}$ of the data for training.

A.4. Logistics

We used the JAX framework (Bradbury et al., 2018) to implement and run the experiments on machines using NVIDIA V100 or A100 GPUs.

B. Ability of Neural Tangent Kernel Models to Interpolate

In this section, we prove that neural tangent kernel models for our one-hidden-layer quadratic network are able to exactly interpolate their training data, i.e. to achieve zero training loss, in the regression setting.

B.1. Full-Rank Kernels are Expressive

Recalling that neural tangent kernel models correspond to "ridgeless" kernel regression (e.g. Jacot et al., 2018; Lee et al., 2019), we first notice that this method can interpolate *any* set of training labels when the kernel is strictly positive definite.

Specifically, empirical neural tangent kernel regression corresponds to ridgeless regression with a prior mean,

$$\operatorname*{argmin}_f \|f - f_0\|_K \text{ s.t. } \forall i,\ f(\boldsymbol{x}_i) = y_i,$$

where f_0 is given by the initial value of the network. This does not matter for the following two results, however; we can simply change the labels to $y_i - f_0(\boldsymbol{x}_i)$ and then assume that $f_0 = 0$.

Lemma B.1. *Let $\mathcal{D}_{\text{train}} = \{(\boldsymbol{x}_i, y_i)\}_{i\in[n]}$, and let K be a kernel such that the kernel matrix $\mathbf{K} = [K(\boldsymbol{x}_i, \boldsymbol{x}_j)]_{ij}$ is strictly positive definite. Then kernel ridgeless regression achieves zero training error on $\mathcal{D}_{\text{train}}$.*

Proof. This well-known result follows from the fact that kernel ridgeless regression finds the function $\hat{f}(\boldsymbol{x}) = \sum_{i=1}^n K(\boldsymbol{x}, \boldsymbol{x}_i)\lambda_i$ for $\lambda_i = [\mathbf{K}^\dagger \mathbf{y}]_i$, where $\mathbf{K}^\dagger$ denotes the Moore-Penrose pseudo-inverse of $\mathbf{K}$, and $\mathbf{y} \in \mathbb{R}^n$ has ith entry y_i. Thus $[\hat{f}(\boldsymbol{x}_i)]_i = \mathbf{K}\mathbf{K}^\dagger \mathbf{y}$. If $\mathbf{K}$ is strictly positive definite, $\mathbf{K}^\dagger = \mathbf{K}^{-1}$, and so $[\hat{f}(\boldsymbol{x}_i)]_i = \mathbf{y}$. □

The following result is a partial converse.

Lemma B.2. *Let $\{\boldsymbol{x}_i\}_{i\in[n]}$ and K be a kernel such that the kernel matrix $\mathbf{K} = [K(\boldsymbol{x}_i, \boldsymbol{x}_j)]_{ij}$ is singular. Then there exists an assignment of $y_i \in \mathbb{R}$ such that kernel ridgeless regression achieves nonzero training error on $\mathcal{D}_{\text{train}} = \{(\boldsymbol{x}_i, y_i)\}_{i\in[n]}$.*

Proof. Let $\mathbf{y}$ be any nonzero vector in the null space of $\mathbf{K}$; this is possible since $\mathbf{K}$ is singular. Then $\mathbf{y}$ is also in the null space of $\mathbf{K}^\dagger$, and the training set predictions are $\mathbf{K}\mathbf{K}^\dagger\mathbf{y} = 0 \neq \mathbf{y}$. □

B.2. Expected NTK is Full-Rank

We next show that, in the regression setting, the *expected* neural tangent kernel is strictly positive definite. While perhaps of interest of its own accord, this will be a key component in our analysis of finite-width networks that follows.

Proposition B.3. *Let the entries of W follow a distribution $\mathcal{P}_W$, and those of V follow $\mathcal{P}_V$, all mutually independent. Assume $\mathcal{P}_W$ has mean zero, variance $\sigma_W^2 > 0$, skewness zero, and kurtosis $\kappa_W > 0$ such that $\mathbb{E}_{w\sim\mathcal{P}_W} w^4 = \kappa_W \sigma_W^4$. Assume that $\mathcal{P}_V$ has mean zero and variance $\sigma_V^2 \geq 0$. Then the expected neural tangent kernel for the regression network with any finite $h \geq 1$ is strictly positive definite on any set of distinct inputs $\{\boldsymbol{x}_i : i \in [n]\} \subseteq [p]^3$.*

We first note that if $\mathcal{P}_W = \mathcal{N}(0, \sigma_W^2)$, $\kappa_V = 3$. If $\mathcal{P}_W = \text{Unif}([-s_W, s_W])$, $\sigma_W^2 = \frac{1}{3}s_W^2$ and $\kappa_W = \frac{9}{5}$. These two distributions cover the vast majority of initialization schemes used in practice. Their parameters likely depend on h and p; for instance, PyTorch (Ansel et al., 2024) defaults to uniform distributions, following He et al. (2015), with $s_W = 1/\sqrt{2p}$ and $s_V = 1/\sqrt{h}$.

Proof. It will be more convenient in this proof to give different names to the sub-matrix of the parameter vector W that act on the a inputs and the b inputs; we will write $W = \begin{bmatrix} Q & R \end{bmatrix}$, where Q, R are each $h \times p$ matrices. Then we can write the full model as

$$g\left(\theta; (e_a, e_b, e_c)\right) = e_c^\mathsf{T} V (Qe_a + Re_b)^{\odot 2} = \sum_{k=1}^{h} V_{ck}(Q_{ka} + R_{kb})^2.$$

The empirical neural tangent kernel between two inputs $\boldsymbol{x} = (e_a, e_b, e_c)$ and $\boldsymbol{x}' = (e_{a'}, e_{b'}, e_{c'})$ is then given by

$$K_\theta(\boldsymbol{x}, \boldsymbol{x}') = \sum_{k=1}^{h}\left[\sum_{c''=1}^{p}\frac{\partial g(\theta;\boldsymbol{x})}{\partial V_{c''k}}\frac{\partial g(\theta;\boldsymbol{x}')}{\partial V_{c''k}} + \sum_{a''=1}^{p}\frac{\partial g(\theta;\boldsymbol{x})}{\partial Q_{ka''}}\frac{\partial g(\theta;\boldsymbol{x}')}{\partial Q_{ka''}} + \sum_{b''=1}^{p}\frac{\partial g(\theta;\boldsymbol{x})}{\partial R_{kb''}}\frac{\partial g(\theta;\boldsymbol{x}')}{\partial R_{kb''}}\right].$$

Evaluating the necessary derivatives gives

$$\frac{\partial g(\theta;\boldsymbol{x})}{\partial V_{c''k}} = \begin{cases}(Q_{ka} + R_{kb})^2 & \text{if } c = c'' \\ 0 & \text{otherwise}\end{cases}$$

$$\sum_{c''=1}^{p}\frac{\partial g(\theta;\boldsymbol{x})}{\partial V_{c''k}}\frac{\partial g(\theta;\boldsymbol{x}')}{\partial V_{c''k}} = \begin{cases}(Q_{ka} + R_{kb})^2(Q_{ka'} + R_{kb'})^2 & \text{if } c = c' \\ 0 & \text{otherwise}\end{cases}$$

$$\frac{\partial g(\theta;\boldsymbol{x})}{\partial Q_{ka''}} = \begin{cases}2V_{ck}(Q_{ka} + R_{kb}) & \text{if } a = a'' \\ 0 & \text{otherwise,}\end{cases}$$

$$\sum_{a''=1}^{p}\frac{\partial g(\theta;\boldsymbol{x})}{\partial Q_{ka''}}\frac{\partial g(\boldsymbol{x}')}{\partial Q_{ka''}} = \begin{cases}4V_{ck}V_{c'k}(Q_{ka} + R_{kb})(Q_{ka} + R_{kb'}) & \text{if } a = a' \\ 0 & \text{otherwise}\end{cases}$$

$$\frac{\partial g(\theta;\boldsymbol{x})}{\partial R_{kb''}} = \begin{cases}2V_{ck}(Q_{ka} + R_{kb}) & \text{if } b = b'' \\ 0 & \text{otherwise}\end{cases}$$

$$\sum_{b''=1}^{p}\frac{\partial g(\boldsymbol{x})}{\partial R_{kb''}}\frac{\partial g(\boldsymbol{x}')}{\partial R_{kb''}} = \begin{cases}4V_{ck}V_{c'k}(Q_{ka} + R_{kb})(Q_{ka'} + R_{kb}) & \text{if } b = b' \\ 0 & \text{otherwise.}\end{cases}$$

Combining, we can write

$$K_\theta(\boldsymbol{x}, \boldsymbol{x}') = \sum_{k=1}^{h} K_{\theta_k}(\boldsymbol{x}, \boldsymbol{x}') \tag{B.1}$$
$$K_{\theta_k}(\boldsymbol{x}, \boldsymbol{x}') = (Q_{ka} + R_{kb})^2 (Q_{ka'} + R_{kb'})^2 \mathbb{1}(c = c') + 4V_{ck}V_{c'k}(Q_{ka} + R_{kb})(Q_{ka'} + R_{kb'})[\mathbb{1}(a = a') + \mathbb{1}(b = b')].$$

The K_{θ_k} functions are iid, so, we will now find $\mathbb{E}_{\theta_k} K_{\theta_k}(\boldsymbol{x}, \boldsymbol{x}') = \frac{1}{h}\mathbb{E}_\theta K_\theta(\boldsymbol{x}, \boldsymbol{x}')$. Throughout, $\boldsymbol{x} = (e_a, e_b, e_c)$ and $\boldsymbol{x}' = (e_{a'}, e_{b'}, e_{c'})$ are fixed, and all expectations are over the relevant set of parameters θ_k.

We have that

$$\mathbb{E}\big[4V_{ck}V_{c'k}(Q_{ka} + R_{kb})(Q_{ka'} + R_{kb'})\big] = 4\,\mathbb{E}\big[V_{ck}V_{c'k}\big]\,\mathbb{E}\big[(Q_{ka} + R_{kb})(Q_{ka'} + R_{kb'})\big].$$

If $c \neq c'$, then $\mathbb{E}[V_{ck}V_{c'k}] = \mathbb{E}[V_{ck}]\,\mathbb{E}[V_{c'k}] = 0$; if $c = c'$, it is $\mathbb{E}\,V_{ck}^2 = \sigma_V^2$. Thus

$$\mathbb{E}\,K_{\theta_k}(\boldsymbol{x}, \boldsymbol{x}') = \mathbb{E}\Big[(Q_{ka} + R_{kb})^2(Q_{ka'} + R_{kb'})^2 + 4\sigma_V^2(Q_{ka} + R_{kb})(Q_{ka'} + R_{kb'})\big[\mathbb{1}(a = a') + \mathbb{1}(b = b')\big]\Big]\mathbb{1}(c = c').$$

If $a \neq a'$ and $b \neq b'$, we have that

$$\mathbb{E}[(Q_{ka} + R_{kb})^2] = \underbrace{\mathbb{E}\,Q_{ka}^2}_{\sigma_W^2} + 2\underbrace{\mathbb{E}\,Q_{ka}}_{0}\underbrace{\mathbb{E}\,R_{kb}}_{0} + \underbrace{\mathbb{E}\,R_{kb}^2}_{\sigma_W^2} = 2\sigma_W^2$$

and so

$$\mathbb{E}[(Q_{ka} + R_{kb})^2(Q_{ka'} + R_{kb'})^2] = \mathbb{E}[(Q_{ka} + R_{kb})^2]\,\mathbb{E}[(Q_{ka'} + R_{kb'})^2] = 4\sigma_W^4;$$

in this case the remaining indicator functions are zero, leaving $\mathbb{E}\,K_{\theta_k}(\boldsymbol{x}, \boldsymbol{x}') = 4\sigma_W^4\mathbb{1}(c = c')$.

If we have $a = a'$ but $b \neq b'$, then

$$\begin{aligned}
\mathbb{E}\left[(Q_{ka} + R_{kb})(Q_{ka} + R_{kb'})\right] &= \underbrace{\mathbb{E}\,Q_{ka}^2}_{\sigma_W^2} + \underbrace{\mathbb{E}\,Q_{ka}}_{0}\underbrace{\mathbb{E}\,R_{kb'}}_{0} + \underbrace{\mathbb{E}\,R_{kb}}_{0}\underbrace{\mathbb{E}\,Q_{ka}}_{0} + \underbrace{\mathbb{E}\,R_{kb}}_{0}\underbrace{\mathbb{E}\,R_{kb'}}_{0} = \sigma_W^2 \\
\mathbb{E}\left[(Q_{ka} + R_{kb})^2(Q_{ka} + R_{kb'})^2\right] &= \mathbb{E}\left[(Q_{ka}^2 + 2Q_{ka}R_{kb} + R_{kb}^2)(Q_{ka}^2 + 2Q_{ka}R_{kb'} + R_{kb'}^2)\right] \\
&= \underbrace{\mathbb{E}\,Q_{ka}^4}_{\kappa_W\sigma_W^4} + 2\underbrace{\mathbb{E}\,Q_{ka}^3}_{0}\underbrace{\mathbb{E}\,R_{kb'}}_{0} + \underbrace{\mathbb{E}\,Q_{ka}^2}_{\sigma_W^2}\underbrace{\mathbb{E}\,R_{kb'}^2}_{\sigma_W^2} \\
&\quad + 2\underbrace{\mathbb{E}\,Q_{ka}^3}_{0}\underbrace{\mathbb{E}\,R_{kb}}_{0} + 4\underbrace{\mathbb{E}\,Q_{ka}^2}_{\sigma_W^2}\underbrace{\mathbb{E}\,R_{kb}}_{0}\underbrace{\mathbb{E}\,R_{kb'}}_{0} + 2\underbrace{\mathbb{E}\,Q_{ka}}_{0}\underbrace{\mathbb{E}\,R_{kb}}_{0}\underbrace{\mathbb{E}\,R_{kb'}^2}_{\sigma_W^2} \\
&\quad + \underbrace{\mathbb{E}\,R_{kb}^2}_{\sigma_W^2}\underbrace{\mathbb{E}\,Q_{ka}^2}_{\sigma_W^2} + 2\underbrace{\mathbb{E}\,R_{kb}^2}_{\sigma_W^2}\underbrace{\mathbb{E}\,Q_{ka}}_{0}\underbrace{\mathbb{E}\,R_{kb'}}_{0} + \underbrace{\mathbb{E}\,R_{kb}^2}_{\sigma_W^2}\underbrace{\mathbb{E}\,R_{kb'}^2}_{\sigma_W^2} \\
&= (\kappa_W + 3)\,\sigma_W^4 \\
\mathbb{E}\,K_{\theta_k}(\boldsymbol{x}, \boldsymbol{x}') &= \big[(\kappa_W + 3)\sigma_W^2 + 4\sigma_V^2\big]\,\sigma_W^2\mathbb{1}(c = c');
\end{aligned}$$

the situation for $a \neq a'$ but $b = b'$ is symmetric, and hence the mean kernel is the same.

Finally, when $a = a'$ and $b = b'$,

$$\begin{aligned}
\mathbb{E}\left[(Q_{ka} + R_{kb})^4\right] &= \underbrace{\mathbb{E}\,Q_{ka}^4}_{\kappa_W\sigma_W^4} + 4\underbrace{\mathbb{E}\,Q_{ka}^3}_{0}\underbrace{\mathbb{E}\,R_{kb}}_{0} + 6\underbrace{\mathbb{E}\,Q_{ka}^2}_{\sigma_W^2}\underbrace{\mathbb{E}\,R_{kb}^2}_{\sigma_W^2} + 4\underbrace{\mathbb{E}\,Q_{ka}}_{0}\underbrace{\mathbb{E}\,R_{kb}^3}_{0} + \underbrace{\mathbb{E}\,R_{kb}^4}_{\kappa_W\sigma_W^4} \\
&= (2\kappa_W + 6)\sigma_W^4 \\
\mathbb{E}\,K_{\theta_k}(\boldsymbol{x}, \boldsymbol{x}') &= \big[(2\kappa_W + 6)\sigma_W^4 + 8\sigma_V^2\sigma_W^2\big]\,\mathbb{1}(c = c').
\end{aligned}$$

Combining the cases, it holds in general that

$$\mathbb{E}\, K_{\theta_k}(\boldsymbol{x}, \boldsymbol{x}') = \sigma_W^2 \mathbb{1}(c = c') \begin{cases} 4\sigma_W^2 & \text{if } a \neq a', b \neq b' \\ (\kappa_W + 3)\sigma_W^2 + 4\sigma_V^2 & \text{if } a = a', b \neq b' \\ (\kappa_W + 3)\sigma_W^2 + 4\sigma_V^2 & \text{if } a \neq a', b = b' \\ (2\kappa_W + 6)\sigma_W^2 + 8\sigma_V^2 & \text{if } a = a', b = b' \end{cases}$$
$$= 4\sigma_W^4 \mathbb{1}(c = c') + \left[(\kappa_W - 1)\sigma_W^2 + 4\sigma_V^2\right]\sigma_W^2 \mathbb{1}(a = a', c = c') + \left[(\kappa_W - 1)\sigma_W^2 + 4\sigma_V^2\right]\sigma_W^2 \mathbb{1}(b = b', c = c') + 4\sigma_W^4 \mathbb{1}(a = a', b = b', c = c').$$

The kurtosis of any probability distribution is at least 1 by Jensen's inequality, so all the coefficients in this last form are nonnegative. We can then use this to construct an explicit feature map, $\mathbb{E}\, K_{\theta_k}(\boldsymbol{x}, \boldsymbol{x}') = \phi(\boldsymbol{x})^\mathsf{T} \phi(\boldsymbol{x}')$, because if $\gamma \geq 0$,

$$\gamma \mathbb{1}(c = c') = \begin{bmatrix} \sqrt{\gamma}\mathbb{1}(c = 1) \\ \sqrt{\gamma}\mathbb{1}(c = 2) \\ \vdots \\ \sqrt{\gamma}\mathbb{1}(c = p) \end{bmatrix}^\mathsf{T} \begin{bmatrix} \sqrt{\gamma}\mathbb{1}(c' = 1) \\ \sqrt{\gamma}\mathbb{1}(c' = 2) \\ \vdots \\ \sqrt{\gamma}\mathbb{1}(c' = p) \end{bmatrix} \tag{B.2}$$

corresponds to features in $\mathbb{R}^p$. The other indicator functions can be implemented in the same way, in $\mathbb{R}^{p^2}$ or $\mathbb{R}^{p^3}$; their sum can then be obtained by concatenating the individual features together.

This construction makes it clear that the function

$$J(\boldsymbol{x}, \boldsymbol{x}') = 4\sigma_W^4 \mathbb{1}(c = c') + \left[(\kappa_W - 1)\sigma_W^2 + 4\sigma_V^2\right]\sigma_W^2 \mathbb{1}(a = a', c = c') + \left[(\kappa_W - 1)\sigma_W^2 + 4\sigma_V^2\right]\sigma_W^2 \mathbb{1}(b = b', c = c')$$

is a positive semi-definite kernel on $[p]^3$, so $\mathbf{J} = [J(\boldsymbol{x}_i, \boldsymbol{x}_j)]_{ij}$ is a positive semi-definite matrix for any $\{\boldsymbol{x}_i : i \in [n]\} \subseteq [p]^3$.

Finally, since $\mathbb{1}(a = a', b = b', c = c') = \mathbb{1}(\boldsymbol{x} = \boldsymbol{x}')$, when the $\{\boldsymbol{x}_i : i \in [n]\}$ are distinct we have that the kernel matrix $\mathbf{K} = [\mathbb{E}\, K_{\theta_k}(\boldsymbol{x}_i, \boldsymbol{x}_j)]_{ij}$ is given as $\mathbf{J} + 4\sigma_W^4 \mathbf{I}$. Since $\mathbf{J}$ has minimum eigenvalue at least 0, $\mathbf{K}$ must have minimum eigenvalue at least $4\sigma_W^4 > 0$, and is hence full-rank. Thus the kernel matrix for $\mathbb{E}_\theta\, K_\theta$ has minimum eigenvalue at least $4\sigma_W^4 h > 0$. □

B.3. Empirical NTKs are Likely Full-Rank

Now, for *bounded* initialization schemes, we use matrix concentration inequalities to show that the empirical neural tangent kernel is also likely to be full-rank when h is large enough.

Proposition B.4. *In the setting of Proposition B.3, further assume that* $\Pr_{w \sim \mathcal{P}_W}(|w| \leq s_W) = 1$, $\Pr_{v \sim \mathcal{P}_V}(|v| \leq s_V) = 1$. *Let the set of inputs* $\{\boldsymbol{x}_i : i \in [n]\} \subseteq [p]^3$ *be distinct and such that no* a *value is seen more than* ρ_a *times, no* b *value is seen more than* ρ_b *times, and no* c *value is seen more than* ρ_c *times. Then the empirical neural tangent kernel for the regression network is strictly positive definite with probability at least* $1 - \delta$ *over the choice of random parameters* θ *as long as*

$$h > \frac{4s_W^2(s_V^2\rho_a + s_V^2\rho_b + s_W^2\rho_c)}{\sigma_W^4} \log \frac{n}{\delta}.$$

If $\mathcal{P}_W$ *is uniform on* $[-s_W, s_W]$, *this condition is equivalent to*

$$h > 36\left(\frac{s_V^2}{s_W^2}(\rho_a + \rho_b) + \rho_c)\right) \log \frac{n}{\delta}.$$

Under the default PyTorch (Ansel et al., 2024) initialization scheme where $s_V^2 = 1/h$ *and* $s_W^2 = 1/(2p)$, *this condition is guaranteed to hold if*

$$h > 36\rho_c \log \frac{n}{\delta} + 6\sqrt{2p(\rho_a + \rho_b) \log \frac{n}{\delta}}.$$

Recall that we are primarily interested in $n = \omega(p)$; otherwise, in the regression setup we have almost no information about the problem (corresponding to a constant number of samples in the classification framing). In this setting, with randomly selected inputs, we expect ρ_a, ρ_b, ρ_c to each be roughly n/p. In fact, as long as $n = \omega(p \log(p)^3)$, Theorem 1 of Raab & Steger (1998) shows that with high probability over the choice of training data selected uniformly (with replacement), the asymptotic value of each ρ is

$$\frac{n}{p} + \sqrt{2\frac{n}{p}\log p\left(1 - \frac{\log\log p}{2\log p}\right)} = \Theta\left(\frac{n}{p}\right).$$

Proposition B.4 thus shows high-probability interpolation for h which grow like $n \log n/p$. Up to logarithmic factors, this indeed explains the setting of most interest in our paper, when $h = \Theta(p)$ and $n = \Theta(p^2)$: NTK-regime models can interpolate the data (Proposition B.4) but not generalize (Theorem D.1), while rich-regime models can generalize (Theorem G.7).

Proposition B.5, afterwards, shows that this n/p threshold on h is tight up to logarithmic factors.

Proof of Proposition B.4. Recall the decomposition of $K_\theta(\boldsymbol{x}, \boldsymbol{x}') = \sum_{k=1}^h K_{\theta_k}(\boldsymbol{x}, \boldsymbol{x}')$ from (B.1). For a fixed set of inputs $\{\boldsymbol{x}_i : i \in [n]\}$, define the $n \times n$ matrix $\mathbf{K}_\theta = [K_\theta(\boldsymbol{x}_i, \boldsymbol{x}_j)]_{ij}$ and h iid matrices $\mathbf{K}_{\theta_k} = [K_{\theta_k}(\boldsymbol{x}_i, \boldsymbol{x}_j)]_{ij}$, so that $\mathbf{K}_\theta = \sum_{k=1}^h \mathbf{K}_{\theta_k}$.

We will next need to show that the operator norm $\|\mathbf{K}_{\theta_k}\|$ is bounded. Using the operator $\mathrm{diag} : \mathbb{R}^n \to \mathbb{R}^{n\times n}$ to construct a diagonal matrix from a vector, we can write

$$\begin{aligned}
\mathbf{K}_{\theta_k} &= \mathrm{diag}(\mathbf{w}_{\theta_k})\left[\mathrm{diag}(\mathbf{w}_{\theta_k})\mathbf{F}\,\mathrm{diag}(\mathbf{w}_{\theta_k}) + \mathrm{diag}(\mathbf{v}_{\theta_k})\mathbf{G}\,\mathrm{diag}(\mathbf{v}_{\theta_k})\right]\mathrm{diag}(\mathbf{w}_{\theta_k}) \qquad \text{(B.3)}\\
(\mathbf{F})_{ij} &= \mathbb{1}(c_i = c_j) \qquad (\mathbf{G})_{ij} = \mathbb{1}(a_i = a_j) + \mathbb{1}(b_i = b_j)\\
(\mathbf{w}_{\theta_k})_i &= Q_{ka_i} + R_{kb_i} \qquad (\mathbf{v}_{\theta_k})_i = 2V_{c_i k}.
\end{aligned}$$

Using $\|AB\| \le \|A\|\|B\|$, $\|\mathrm{diag}(v)\| = \max_i |v_i|$, and $\|A + B\| \le \|A\| + \|B\|$, we obtain that

$$\begin{aligned}
\|\mathbf{K}_{\theta_k}\| &\le (2s_W)^4\|\mathbf{F}\| + (2s_W)^2(2s_V)^2\|\mathbf{G}\|\\
&= 16s_W^4\|\mathbf{F}\| + 16s_W^2 s_V^2\|\mathbf{G}\|.
\end{aligned}$$

Using (B.2), we can write $\mathbf{F} = \Phi_c\Phi_c^\mathsf{T}$, $\mathbf{G} = \Phi_a\Phi_a^\mathsf{T} + \Phi_b\Phi_b^\mathsf{T}$, where $\Phi_a \in \mathbb{R}^{n\times p}$ is given by

$$\Phi_a = \begin{bmatrix} \mathbb{1}(a_1 = 1) & \cdots & \mathbb{1}(a_1 = p) \\ \vdots & \ddots & \vdots \\ \mathbb{1}(a_n = 1) & \cdots & \mathbb{1}(a_n = p) \end{bmatrix},$$

and similarly for Φ_b and Φ_c. Moreover, $\|\mathbf{F}\| = \|\Phi_c\|^2$, and $\|\mathbf{G}\| \le \|\Phi_a\|^2 + \|\Phi_b\|^2$. These norms are simple to evaluate:

$$\begin{aligned}
(\Phi_a x)_i &= \sum_{\ell=1}^p \mathbb{1}(a_i = \ell)x_\ell = x_{a_i}\\
\|\Phi_a x\|^2 &= \sum_{i=1}^n x_{a_i}^2 = \sum_{\ell=1}^p x_\ell^2\left(\sum_{i=1}^n \mathbb{1}(a_i = \ell)\right)\\
\|\Phi_a\|^2 &= \sup_{\|x\|=1}\sum_{\ell=1}^p x_\ell^2\left(\sum_{i=1}^n \mathbb{1}(a_i = \ell)\right) = \max_{\ell\in[p]}\sum_{i=1}^n \mathbb{1}(a_i = \ell).
\end{aligned}$$

Thus it holds almost surely that

$$\|\mathbf{K}_{\theta_k}\| \le 16s_W^2(s_V^2\rho_a + s_V^2\rho_b + s_W^2\rho_c) =: L.$$

We also know from the very end of the proof of Proposition B.3 that the smallest eigenvalue of $\mathbb{E}\,\mathbf{K}_\theta$ is at least $\mu_{\min} := 4\sigma_W^4 h$. Applying a matrix Chernoff bound (Tropp, 2015, Theorem 5.1.1), it holds for any $\epsilon \in [0, 1)$ that

$$\Pr\left(\lambda_{\min}(\mathbf{K}_\theta) > (1-\epsilon)\mu_{\min}\right) \ge 1 - n\left(\frac{e^{-\epsilon}}{(1-\epsilon)^{1-\epsilon}}\right)^{\mu_{\min}/L}.$$

Because we only care about the smallest eigenvalue of $\mathbf{K}_\theta$ being strictly positive, we can take the limit as $\epsilon \nearrow 1$; since $e^{-\epsilon}/(1-\epsilon)^{1-\epsilon} \to 1/e$, this gives

$$\begin{aligned}\Pr\left(\lambda_{\min}(\mathbf{K}_\theta) > 0\right) &\geq 1 - n\exp\left(-\frac{\mu_{\min}}{L}\right) \\ &= 1 - n\exp\left(-\frac{4\sigma_W^4 h}{16 s_W^2(s_V^2\rho_a + s_V^2\rho_b + s_W^2\rho_c)}\right).\end{aligned}$$

Thus, to achieve a full-rank $\mathbf{K}_\theta$ with probability at least $1-\delta$, it suffices to have

$$h > \frac{4s_W^2(s_V^2\rho_a + s_V^2\rho_b + s_W^2\rho_c)}{\sigma_W^4}\log\frac{n}{\delta}.$$

For uniform distributions, $\sigma_W^2 = \frac{1}{3}s_W^2$, hence the condition becomes

$$h > 36\left(\frac{s_V^2}{s_W^2}(\rho_a + \rho_b) + \rho_c\right)\log\frac{n}{\delta}.$$

With the PyTorch default initialization scheme, $s_V^2 = 1/h$ and $s_W^2 = 1/(2p)$, making the condition

$$\begin{aligned}h &> 36\left(\frac{2p}{h}(\rho_a + \rho_b) + \rho_c\right)\log\frac{n}{\delta} \\ h^2 &> 36\left(2p(\rho_a + \rho_b) + h\rho_c\right)\log\frac{n}{\delta} \\ h^2 - \underbrace{36\rho_c\log\frac{n}{\delta}}_{\beta}h - \underbrace{36\cdot 2p(\rho_a+\rho_b)\log\frac{n}{\delta}}_{\gamma} &> 0.\end{aligned}$$

The final left-hand-side is a quadratic function of h, which is positive for very large or very negative h and negative at $h = 0$. Thus, the condition holds when h exceeds $\frac{1}{2}\beta + \frac{1}{2}\sqrt{\beta^2 + 4\gamma} \leq \beta + \sqrt{\gamma}$, i.e. it suffices that

$$h > 36\rho_c\log\frac{n}{\delta} + 6\sqrt{2p(\rho_a+\rho_b)\log\frac{n}{\delta}}.$$ □

This threshold is indeed tight up to logarithmic factors, in the sense that there are some labels which kernel ridgeless regression cannot achieve when $h = o(n/p)$.

Proposition B.5. *In the setting of Proposition B.3, the empirical neural tangent kernel of the regression network $\mathbf{K}_\theta$ is guaranteed to be singular when $h < n/(3p)$.*

Proof. Recall the decomposition of $\mathbf{K}_{\theta_k}$ from (B.3). By (B.2), the matrix $\mathbf{F}$ has rank at most p, and $\mathbf{G}$ has rank at most $2p$; thus $\mathbf{K}_{\theta_k}$ has rank at most $3p$, and $\operatorname{rank}(\mathbf{K}_\theta) \leq 3ph$. If $3ph < n$, this $n \times n$ matrix cannot be full-rank. □

C. Permutation-Equivariance of Gradient-Based Training

We first define the notion of permutation equivariance. Towards this, we borrow the following proof from Appendix C of Li et al. (2020):

Definition C.1 (Gradient-Based Algorithm $\mathcal{A}$)**.** We borrow the definition of Algorithm 1 in Li et al. (2020) with the slight modification of restricting the update rule $F(U, \mathcal{M}, \mathcal{D}_{\text{train}})$ where $\mathcal{M} : U \to (\mathcal{X} \to \mathbb{R})$ to only allow gradient-based update rules, such that

$$F(U, \mathcal{M}, \mathcal{D}_{\text{train}}) = U - \eta\nabla_U\mathcal{L}(g, U, \mathcal{D}_{\text{train}})$$

for some $\eta \in \mathbb{R}$ and loss function $\mathcal{L}$ where $g : \mathcal{X} \to \mathbb{R}$ denotes the neural network defined in Section 2.

Theorem C.2. *Suppose $\mathcal{G}_\mathcal{X}$ is a group acting on $\mathcal{X}$. The gradient-based iterative algorithm $\mathcal{A}$ (defined in Definition C.1) is $\mathcal{G}_\mathcal{X}$-equivariant if:*

1. *There exists a group $\mathcal{G}_\theta$ acting on parameters U and a group isomorphism $\tau : \mathcal{G}_\mathcal{X} \to \mathcal{G}_\theta$ such that for all $x \in \mathcal{X}, T \in \mathcal{G}_\mathcal{X}, U$ we have that $f(U; x) = f(\tau(T)(U); T(x))$.*
2. *The gradient update rule is invariant under joint group action $(T, \tau(T))$ for all $T \in \mathcal{G}_\mathcal{X}$: $\tau(T)(\nabla_U g(U; x)) = \nabla_U g(\tau(T)(U); T(x))$.*
3. *The initialization distribution P_{init} is invariant under $\mathcal{G}_\mathcal{X}$.*

We now present our permutation-equivariance results.

Definition C.3. We define $\mathcal{G}_\theta$ as a group of actions to be applied on $U = (W, V)$ as

$$\mathcal{G}_\theta \triangleq \{\pi_{\sigma_1,\sigma_2,\sigma_3} \mid \sigma_1, \sigma_2, \sigma_3 \in \mathbb{S}_p\}$$

where $\pi_{\sigma_1,\sigma_2,\sigma_3}$ is defined as $\pi_{\sigma_1,\sigma_2,\sigma_3}(W, V) \triangleq (W', V')$, where $\forall i \in [h], j \in [p], W'_{ij} = W_{i,\sigma_1(j)}, W'_{i(j+p)} = W_{i,\sigma_2(j)}, V'_{\sigma_3(j),i} = V_{j,i}$. This means the concatenation of the row is permuted in the same way as the data does. Furthermore, the rows of V also get permuted according to σ_3.

Remark C.4. *There is a one-to-one mapping τ between $\mathcal{G}_\theta$ and $\mathcal{G}_\mathcal{X}$, which is $\tau(\sigma_1, \sigma_2, \sigma_3) = \pi_{\sigma_1,\sigma_2,\sigma_3}$.*

Lemma C.5. *For every $x = (i, j, k)$ where $i, j, k \in [p]$, $T \in \mathcal{G}_\mathcal{X}$ we have*

$$g(U; x) = g(\tau(T)(U); T(x)).$$

Proof. For each $x = (i, j, k) \in \mathcal{X}; \sigma_1, \sigma_2, \sigma_3 \in \mathbb{S}_p; U = (W, V)$ we have that

$$\begin{aligned}
g(\tau(\sigma_1,\sigma_2,\sigma_3)(U); (\sigma_1,\sigma_2,\sigma_3)(x)) &= \left\langle e_{\sigma_3(k)}, V'_{\sigma_3} \left(\begin{bmatrix} W_1'^\top(\sigma_1(i), \sigma_2(j)) \\ W_2'^\top(\sigma_1(i), \sigma_2(j)) \\ \vdots \\ W_h'^\top(\sigma_1(i), \sigma_2(j)) \end{bmatrix} \right)^{\odot 2} \right\rangle \\
&= \left\langle e_k, V \left(\begin{bmatrix} W'_{1,\sigma_1(i)} + W'_{1,\sigma_2(j)+p} \\ W'_{2,\sigma_1(i)} + W'_{2,\sigma_2(j)+p} \\ \vdots \\ W'_{h,\sigma_1(i)} + W'_{h,\sigma_2(j)+p} \end{bmatrix} \right)^{\odot 2} \right\rangle \\
&= \left\langle e_k, V \left(\begin{bmatrix} W_{1,i} + W_{1,j+p} \\ W_{2,i} + W_{2,j+p} \\ \vdots \\ W_{h,i} + W_{h,j+p} \end{bmatrix} \right)^{\odot 2} \right\rangle \\
&= \langle e_k, V(Wx)^{\odot 2} \rangle \\
&= g(U; x).
\end{aligned}$$

□

Lemma C.6. *For every $x = (i, j, k)$ where $i, j, k \in [p]$, $T \in \mathcal{G}_\mathcal{X}$ we have*

$$\tau(T)(\nabla_U g(U; x)) = \nabla_U g(\tau(T)(U); T(x)).$$

Proof. We first consider the gradient of the second layer. For all $a, b \in [p]$ and $\sigma_1, \sigma_2, \sigma_3 \in \mathbb{S}_p$:

$$\begin{aligned}
&\tau^{-1}(\pi_{\sigma_1,\sigma_2,\sigma_3}) \left(\nabla_{V_b} g(\tau(\sigma_1,\sigma_2,\sigma_3)(U); (\sigma_1,\sigma_2,\sigma_3)(x)) \right) \\
&= I(\sigma_3(k) = \sigma_3(b)) \nabla_{V_b} g\left(\tau(\sigma_1,\sigma_2,\sigma_3)(U); (\sigma_1,\sigma_2,\sigma_3)(x) \right) \\
&= I(k = b)(W'(\sigma_1, \sigma_2)(x))^{\odot 2} \\
&= I(k = b)(Wx)^{\odot 2} \\
&= \nabla_{V_b} g(U; x).
\end{aligned}$$

For the gradients of the first layer we have:

$$\begin{aligned}
&\tau^{-1}(\pi_{\sigma_1,\sigma_2,\sigma_3})\left(\nabla_W g(\tau(\sigma_1,\sigma_2,\sigma_3)(U);(\sigma_1,\sigma_2,\sigma_3)(x))\right)\\
&=\tau^{-1}(\pi_{\sigma_1,\sigma_2,\sigma_3})\left(2V_k\odot(W'_{(\sigma_1,\sigma_2)}(e_{\sigma_1(i)},e_{\sigma_2(j)})^\top)\,(e_{\sigma_1(i)},e_{\sigma_2(j)})\right)\\
&=\tau^{-1}(\sigma_1,\sigma_2,\sigma_3)\left(2V_k\odot(W(e_i,e_j))\,(e_{\sigma_1(i)},e_{\sigma_2(j)})^\top\right)\\
&=2V_k\odot(Wx)\,(\sigma_1,\sigma_2)^{-1}\left((e_{\sigma_1(i)},e_{\sigma_2(j)})\right)\\
&=2V_k\odot(W(e_i,e_j)^\top)\,(e_i,e_j)\\
&=\nabla_W g(U;x).
\end{aligned}$$

□

Lemma C.7. *Training a one-hidden-layer neural network with quadratic activations using gradient-based update rules on modular addition data modeled as a regression task as defined in Equation (1.1) is permutation-equivariant, as defined in Theorem C.2.*

Proof. We show that training a neural network with gradient descent in our setting satisfies the three conditions proposed in Theorem C.2 and thus, this theorem applies to the training process. Equivariance of the forward pass and the backward pass (update rule) are settled through Lemmas C.5 and C.6. Finally, it is straightforward to see that the initialization is invariant under $\mathcal{G}_\theta$ defined in Definition C.3. Since the distribution of initialization is symmetric and each parameter is initialized independently and identically from the same distribution, the action of swapping rows or columns doesn't change the distribution. □

Corollary C.8 (Equivariance of Adam)**.** *Lemma C.7 applies to other gradient-based training algorithms that need memory, such as Adam.*

Sketch of Proof for Corollary C.8. To prove that other gradient-based algorithms, particularly Adam, are also equivariant to the permutation group $\mathcal{G}_\mathcal{X}$ we just need to ensure that the update rule of these algorithms is equivariant under the joint group action $(T,\tau(T))$. First, note that linear operations (such as weight decay) on gradients and parameters equivariant. To track the momentum at different steps of the algorithm we can apply an induction on equivariance of these variables on the step number. At $t=0$ they're both zero. m_t is a linear combination m_{t-1} and g_t which both are equivarint under the joint action $(T,\tau(T))$. Since the gradient is equivariant, the coordinate-wise squared gradient is also equivariant and the linear combination of it with v_{t_1} is also equivariant. This settles the equivariance of the update rule of Adam and other similar gradient-based algorithms that need memory and buffers. □

D. Lower Bound of Population Loss for Kernel Methods

In this section we present the formal version of Theorem 3.4 alongside a proof of it. Note that a kernel-based predictor h on a training data $\{(\boldsymbol{x}_i,y_i)\}_{i=1}^n$ can be expressed as $h(x)=\sum_{i=1}^n\lambda_iK(\boldsymbol{x}_i,x)$ where $\lambda_i;i\in[n]$ are constants. Assuming that the kernel's feature maps are of dimension d, the predictions are linear combinations of d-dimensional feature maps. We first present a general proof that every permutation invariant kernel requires $\Omega(p^2)$ training points to outperform the null predictor in terms of ℓ_2 loss, and then show that this theorem applies to the distribution of empirical NTKs at initialization.

D.1. Notation

We use $[p]$ to denote the set $\{1,\dots,p\}$. We use $\mathbb{S}_p$ to denote the permutation groups over p elements, $\mathbb{S}_m$ as permutation group over m elements and id is the identity mapping. For any nonempty set $\mathcal{X}$, a symmetric function $K:\mathcal{X}\times\mathcal{X}\to\mathbb{R}$ is called a positive semi-definite kernel (p.s.d) kernel on $\mathcal{X}$ if for any $n\in\mathbb{N}$, any $\boldsymbol{x}_1,\dots,\boldsymbol{x}_n$ and $\lambda_1,\dots,\lambda_n\in\mathbb{R}$, it holds that $\sum_{i=1}^n\sum_{j=1}^n\lambda_i\lambda_jK(\boldsymbol{x}_i,\boldsymbol{x}_j)\geq 0$. For a subspace V of $\mathbb{R}^n$ and vector $x\in\mathbb{R}^n$, we define $\mathsf{dist}(x,V)\triangleq\min_{v\in V}\|x-v\|_2$.

For any p^m-dimensional vector $v\in\mathbb{R}^{p^m}$ we denote by $v(x)$ the vector v indexed by an m-dimensional index vector $x\in[p]^m$. We define the vector $s_{i,a}\in\mathbb{R}^{p^m}$ whose entries are

$$s_{i,a}(x)\triangleq\begin{cases}1, & x[i]=a;\\ 0, & \text{otherwise.}\end{cases}\tag{D.1}$$

which helps us denote the all-once slices in this vector space, $V_s \triangleq \text{span}\{s_{i,a}\}_{i\in[m],a\in[p]}$. We further define $\Delta(x, x')$ where x, x' are two index vectors of size m as the number of equal indices between them, formally:

$$\Delta(x, x') = \sum_{i=1}^{m} \mathbf{1}_{x[i]=x'[i]}.$$

We also define $\mathcal{X}_{a,b}$ as the set of all $x, x' \in [p]^b$ index vector pairs such that $\mathsf{dist}(x, x') = a$, formally:

$$\mathcal{X}_{a,b} \triangleq \{(x, x') : x, x' \in [p]^b \wedge \Delta(x, x') = a\}.$$

When $b = m$, we drop the second index and write $\mathcal{X}_a$ (instead of $\mathcal{X}_{a,m}$) for simplicity. It's clear that the collection of all $\mathcal{X}_d$ for $0 \le d \le m$ is a partitioning of the set of all pairs of index vectors. Unless stated otherwise, x refers to the m-dimensional index vector. Finally, we define $\mathbb{U}_m \triangleq [\text{Unif}(\mathbb{S}_p)]^m$ as the product of m Uniform distribution on $\mathbb{S}_p$.

D.2. Loss Lower Bound: 3-dimensional Case

For convenience of notation, we will use the (i, j, k) and $e_i + e_{j+p} + e_{k+2p}$ interchangebly for $i, j, k \in [p]$. We denote the function $K(\boldsymbol{x}_t, \cdot) : [p] \times [p] \times [p] \to \mathbb{R}$ as a tensor on $\mathbb{R}^{p\times p\times p}$ by $v_t(\cdot)$ for each $t \in [n]$. We also define function $\psi_{\sigma_1,\sigma_2,\sigma_3}(i, j, k) \triangleq \mathbf{1}\Big(\sigma_1(i) + \sigma_2(j) \equiv \sigma_3(k) \pmod p\Big)$. We can view a function mapping from $[p] \times [p] \times [p] \to \mathbb{R}$ as a vector of size p^3 and define inner products and dist on the function space, *i.e.*, $\langle \psi, \psi' \rangle \triangleq \sum_{i,j,k\in[p]} \psi(i, j, k)\psi'(i, j, k)$ and $\|h\|_2^2 \triangleq \langle h, h\rangle$.

Theorem D.1. *For any integers $n \ge 1$, $p \ge 2$ and kernel $K : ([p] \times [p] \times [p]) \times ([p] \times [p] \times [p]) \to \mathbb{R}$, suppose $\boldsymbol{x}_t = (i_t, j_t, k_t) \overset{i.i.d.}{\sim} \text{Unif}([p] \times [p] \times [p])$ for each $t \in [n]$, it holds that*

$$\mathop{\mathbb{E}}_{\boldsymbol{x}_1,\dots,\boldsymbol{x}_n} \mathop{\mathbb{E}}_{\sigma_1,\sigma_2,\sigma_3\sim\text{Unif}(\mathbb{S}_p)} \inf_{\lambda_1,\dots,\lambda_n\in\mathbb{R}} \left\| \sum_{t=1}^{n} \lambda_t K(\boldsymbol{x}_t, \cdot) - \psi_{\sigma_1,\sigma_2,\sigma_2}(\cdot) \right\|_2^2 \ge p^2\left(1 - \frac{1}{p} - \frac{n}{p^3}\exp\left(\frac{2}{p-1}\right)\right) \tag{D.2}$$

In other words, if $n \le (1 - \Omega(1))p^3$, then the expected population ℓ_2 loss is at least $\Omega(p^2)$, which is of the same magnitude as the trivial all-zero predictor.

Proof of Theorem D.1. This Theorem is a direct result of Corollary D.7 for the case where $m = 3$. □

Theorem 3.4 (Lower Bound)**.** *There exists a constant $C > 0$ such that for any $p \ge 2$, training data size $n < Cp^3$, and any permutation-equivariant kernel method $\mathcal{A}$, it holds that*

$$\mathop{\mathbb{E}}_{(\boldsymbol{x}_i,y_i)_{i=1}^n\sim\mathcal{D}^n} \mathop{\mathbb{E}}_{\mathcal{A}} \mathcal{L}_{\ell_2}\left(\mathcal{A}\left(\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n\right)\right) \ge \frac{1}{2}\,\mathcal{L}_{\ell_2}\left(h_{\mathbf{0}}\right) = \frac{p}{2},$$

where $\mathbb{E}_{\mathcal{A}}$ is over the randomness in algorithm $\mathcal{A}$.

Proof of Theorem 3.4. Because $\mathcal{A}$ is permutation-equivariant, we have that

$$\begin{aligned}
&\mathop{\mathbb{E}}_{(\boldsymbol{x}_i,y_i)_{i=1}^n\sim\mathcal{D}^n} \mathop{\mathbb{E}}_{\mathcal{A}} \mathcal{L}_{\ell_2}\left(\mathcal{A}\left(\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n\right)\right)\\
=&\mathop{\mathbb{E}}_{(\boldsymbol{x}_i,y_i)_{i=1}^n\sim\mathcal{D}^n} \mathop{\mathbb{E}}_{\mathcal{A}} \left\|\mathcal{A}\left(\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n\right) - p\cdot\psi_{\mathsf{id},\mathsf{id},\mathsf{id}}\right\|^2/p^3\\
=&\mathop{\mathbb{E}}_{(\boldsymbol{x}_i,y_i)_{i=1}^n\sim\mathcal{D}^n} \mathop{\mathbb{E}}_{\mathcal{A}} \left\|\mathcal{A}\left(\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n\right)/p - \psi_{\mathsf{id},\mathsf{id},\mathsf{id}}\right\|^2/p\\
=&\mathop{\mathbb{E}}_{(\boldsymbol{x}_i,y_i)_{i=1}^n\sim\mathcal{D}^n} \mathop{\mathbb{E}}_{\sigma_1,\sigma_2,\sigma_3\sim\text{Unif}(\mathbb{S}_p)} \mathop{\mathbb{E}}_{\mathcal{A}} \left\|\mathcal{A}\left(\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n\right)/p - \psi_{\sigma_1,\sigma_2,\sigma_3}\right\|^2/p.
\end{aligned}$$

Because $\mathcal{A}$ is a kernel method, we have for any $(\boldsymbol{x}_i, y_i)_{i=1}^n$ and $\sigma_1, \sigma_2, \sigma_3 \in \mathbb{S}_p$

$$\left\|\mathcal{A}\left(\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n\right)/p - \psi_{\sigma_1,\sigma_2,\sigma_3}\right\|^2 \ge \inf_{\lambda_1,\dots,\lambda_n\in\mathbb{R}} \left\| \sum_{t=1}^{n} \lambda_t K(\boldsymbol{x}_t, \cdot) - \psi_{\sigma_1,\sigma_2,\sigma_2}(\cdot) \right\|_2^2.$$

Applying Theorem D.1 completes the proof. □

D.3. Loss Lower Bound: General Theorem For Arbitrary Functions

Lemma D.2. *For any subspace V of $\mathbb{R}^n$ and vector $x \in \mathbb{R}^n$, let $\{v_i\}_{i=1}^m$ be an orthonormal basis of V. It holds that* $\mathsf{dist}^2(x, V) = \|x\|_2^2 - \sum_{i=1}^m \langle x, v_i\rangle^2$.

The proof of Lemma D.2 is straightforward and thus omitted.

Lemma D.3. *For any distribution $\mathcal{D}$ over functions mapping from $\mathcal{X} \to \mathbb{R}$ and n functions $\{k_i\}_{i=1}^n$ where $k_i : \mathcal{X} \to \mathbb{R}$ for each $i \in [n]$ it holds that*

$$\mathop{\mathbb{E}}_{h\sim\mathcal{D}}\left[\min_{\alpha\in\mathbb{R}^n} \frac{1}{|\mathcal{X}|}\sum_{x\in\mathcal{X}}\left(\psi(x) - \sum_{i=1}^n \alpha_i k_i(x)\right)^2\right] \geq \frac{1}{|\mathcal{X}|}\sum_{i=n+1}^{|\mathcal{X}|}\lambda_i(\Sigma).$$

where $\Sigma_{\mathcal{D}}(x, x') \triangleq \mathbb{E}_{\psi\sim\mathcal{D}}[\psi(x)\psi(x')]$ and λ_i denotes the i'th largest eigenvalue function. For notational convenience, we also view $\Sigma_{\mathcal{D}}$ as a $\mathbb{R}^{|\mathcal{X}|\times|\mathcal{X}|}$ matrix.

Proof of Lemma D.3. For each h, we define $r_h \in \mathbb{R}^{|\mathcal{X}|}$ whose entries are realization of the function h on inputs $x \in \mathcal{X}$. It suffices to show that

$$\mathop{\mathbb{E}}_{h\sim\mathcal{D}}\left[\mathsf{dist}^2(r_h, V)\right] \geq \sum_{i=n+1}^{|\mathcal{X}|}\lambda_i(\Sigma_{\mathcal{D}})$$

for any subspace $V = \mathrm{span}\{v_1, v_2, \cdots, v_n\} \subset \mathbb{R}^{|\mathcal{X}|\times|\mathcal{X}|}$ where v_i for $i \in [n]$ are orthonormal vectors. Note that

$$\begin{aligned}
\mathop{\mathbb{E}}_{h\sim\mathcal{D}}\left[\mathsf{dist}^2(r_h, V)\right] &= \mathop{\mathbb{E}}_{h\sim\mathcal{D}}\left[\|r_h\|_2^2 - \sum_{t=0}^n \langle v_t, r_h\rangle^2\right] \\
&= \mathrm{Tr}\left(\mathop{\mathbb{E}}_{\psi\sim\mathcal{D}}\left[\psi\psi^\top\right]\right) - \sum_{t=1}^n v_t^\top\left(\mathop{\mathbb{E}}_{h\sim\mathcal{D}}\left[\psi\psi^\top\right]\right)v_t \\
&\geq \mathrm{Tr}\left(\Sigma_{\mathcal{D}}\right) - \sum_{i=1}^n \lambda_i\left(\Sigma_{\mathcal{D}}\right) \\
&= \sum_{i=n+1}^{|\mathcal{X}|}\lambda_i(\Sigma_{\mathcal{D}})
\end{aligned} \tag{D.3}$$

where the inequality in the second to last line is due to the min-max theorem (also called Courant–Fischer–Weyl min-max principle) (Courant & Hilbert, 1953). This completes the proof. □

The following Corollary D.4 is a direct consequence of Lemma D.3, noting that $\mathbb{E}_{\psi\sim\mathcal{D}}\left[\frac{1}{|\mathcal{X}|}\sum_{x\in\mathcal{X}}\psi(x)^2\right] = \mathrm{Tr}(\Sigma_{\mathcal{D}})$.

Corollary D.4. *For any distribution $\mathcal{D}$ over functions mapping from $\mathcal{X} \to \mathbb{R}$ and n functions $\{k_i\}_{i=1}^n$ where $k_i : \mathcal{X} \to \mathbb{R}$ for each $i \in [n]$, if $\sum_{i=1}^n \lambda_i(\Sigma_{\mathcal{D}}) \leq \frac{1}{2}\mathrm{Tr}(\Sigma_{\mathcal{D}})$ then it is guaranteed that*

$$\mathop{\mathbb{E}}_{\psi\sim\mathcal{D}}\left[\min_{\alpha\in\mathbb{R}^n} \frac{1}{|\mathcal{X}|}\sum_{x\in\mathcal{X}}\left(\psi(x) - \sum_{i=1}^n \alpha_i k_i(x)\right)^2\right] \geq \frac{1}{2}\mathop{\mathbb{E}}_{\psi\sim\mathcal{D}}\left[\frac{1}{|\mathcal{X}|}\sum_{x\in\mathcal{X}}\psi(x)^2\right]$$

where the right-hand side denotes the expected loss of the all-0 predictor.

Lemma D.5. *For any matrix $\Sigma \in \mathbb{R}^{d\times d}$, subspace V and projection matrix $P_V \in \mathbb{R}^{d\times d}$ corresponding to V, it holds that*

$$\sum_{i=1}^n \lambda_i(\Sigma) \leq n\cdot\lambda_1\Big((I - P_V)\Sigma(I - P_V)\Big) + \mathrm{Tr}\left(P_V\Sigma P_V\right).$$

Proof. Let $V_n = \text{span}\{\alpha_i\}_{i=1}^n$ where $\alpha_i \in \mathbb{R}^d$ is the i'th eigenvector of Σ and let P_{V_n} be its corresponding projection matrix. Let P_{V_n+V} be the projection onto the sum of two subspaces $V_n + V$, it holds that

$$\begin{aligned}\sum_{i=1}^{n} \lambda_i(\Sigma) &= \text{Tr}(P_{V_n}\Sigma P_{V_n}) \\ &\le \text{Tr}(P_{V_n+V}\Sigma P_{V_n+V}) \\ &= \text{Tr}(P_V \Sigma P_V) + \text{Tr}\left((P_{V_n+V} - P_V)\Sigma(P_{V_n+V} - P_V)\right) \\ &\le \text{Tr}(P_V \Sigma P_V) + n \cdot \lambda_1\left((P_{V_n+V} - P_V)\Sigma(P_{V_n+V} - P_V)\right) \\ &\le \text{Tr}(P_V \Sigma P_V) + n \cdot \lambda_1\left((I - P_V)\Sigma(I - P_V)\right). \end{aligned} \tag{D.4}$$

Here the second to last inequality is because $P_{V_n+V} - P_V$ is at most rank-n and so is $(P_{V_n+V} - P_V)\Sigma(P_{V_n+V} - P_V)$. The last inequality is because $P_{V_n+V} \le I$ and thus $(P_{V_n+V} - P_V)\Sigma(P_{V_n+V} - P_V) \preceq (I - P_V)\Sigma(I - P_V)$. □

D.4. Loss Lower Bound: Modular Addition with m Summands

Lemma D.6. *Let $\mathcal{D}$ be the uniform distribution over*

$$\mathcal{H} \triangleq \left\{ \psi(x) = \mathbf{1}\left(\sum_{i=1}^{m} \sigma_i(x_i) \equiv 0 \pmod p\right) \,\middle|\, \sigma_i \in \mathbb{S}_p \text{ for all } i \in [m] \right\} \tag{D.5}$$

and $\Sigma_{\mathcal{D}}(x, x') = \mathbb{E}_{\psi\sim\mathcal{D}}[\psi(x)\psi(x')]$. It holds that

$$\sum_{i=1}^{n} \lambda_i(\Sigma) \le p^{m-2} + \frac{n}{p}\exp\left(\frac{m-1}{p-1}\right).$$

Proof of Lemma D.6. Consider the vector space V_s defined in Equation (D.1) and the projection matrix $P_{V_s} \in \mathbb{R}^{|\mathcal{X}|\times|\mathcal{X}|}$ onto V_S. To find $\text{Tr}\left(P_{V_s}\Sigma P_{V_s}\right)$ we can first derive $\text{Tr}\left(P_{V_s}\psi\psi^\top P_{V_s}\right)$ where $\psi \in \mathbb{R}^{|\mathcal{X}|}$ is a sample from $\mathcal{D}$. Note that since $(I - P_{V_s})h$ should be orthogonal to V_s (the sum of each slice of the projected vector should be zero) we have that

$$(I - P_{V_s})\psi(x) = \begin{cases} -\frac{1}{p} & \psi(x) = 0 \\ \frac{p-1}{p} & \psi(x) = 1 \end{cases}.$$

Hence, $P_{V_s}\psi(x) = \frac{1}{p}$ for all $x \in \mathcal{X}$. Thus,

$$\begin{aligned}\text{Tr}\left(P_{V_s}\Sigma P_{V_s}\right) &= \mathop{\mathbb{E}}_{\psi\sim\mathcal{D}}\left[\text{Tr}\left(P_{V_s}\psi\psi^\top P_{V_s}\right)\right] \\ &= \mathop{\mathbb{E}}_{\psi\sim\mathcal{D}}\left[\text{Tr}\left(\frac{1}{p^2}\mathbf{1}_{|\mathcal{X}|\times|\mathcal{X}|}\right)\right] \\ &= \frac{|\mathcal{X}|}{p^2} \\ &= p^{m-2}\end{aligned} \tag{D.6}$$

where $\mathbf{1}_{|\mathcal{X}|\times|\mathcal{X}|}$ denotes the all-one square matrix of size $|\mathcal{X}|$. Moreover, by Lemma D.8 we have that

$$n \cdot \lambda_1\left((I - P_{V_s})\Sigma(I - P_{V_s})\right) = \sup_{\|v\|_2\le 1, v\perp V_s} v^\top \Sigma_{\mathcal{D}} v \le \frac{n}{p}\exp\left(\frac{m-1}{p-1}\right). \tag{D.7}$$

Combining the two equations above, we can see that

$$\sum_{i=1}^{n} \lambda_i(\Sigma) \le p^{m-2} + \frac{n}{p}\exp\left(\frac{m-1}{p-1}\right).$$

This concludes the proof. □

Corollary D.7. *Consider the function class $\mathcal{H}$ defined in Equation (D.5) and the uniform distribution over it denoted by $\mathcal{D}$. For any integers $n \geq 1, p \geq 2$, $1 \leq m < p$, kernel $K : [p]^m \times [p]^m \to \mathbb{R}$ it holds that*

$$\mathop{\mathbb{E}}_{x_1,x_2,\cdots x_n} \mathop{\mathbb{E}}_{\psi\sim\mathcal{D}} \inf_{\alpha\in\mathbb{R}^n} \left\| \sum_{t=1}^{n} \lambda_t K(x_t,\cdot) - \psi \right\|_2^2 \geq p^{m-1}\left(1 - \frac{1}{p} - \frac{n}{p^m}\exp\left(\frac{m-1}{p-1}\right)\right).$$

where $x_1, x_2, \cdots, x_t \sim \text{Unif}\left([p]^m\right)$. In other words, if $n < (1-\Omega(1))p^m$, then the expected population ℓ_2 loss is at least $\Omega(p^{m-1})$, which is of the same magnitude as the trivial all-zero predictor.

Proof of Corollary D.7. It suffices to show that for any n-dimensional subspace $V \subset \mathbb{R}^{p^m}$ it holds that

$$\mathop{\mathbb{E}}_{\psi\sim\mathcal{D}} \mathsf{dist}^2(V,\psi) \geq p^{m-1}\left(1 - \frac{1}{p} - \frac{n}{p^m - 1}\exp\left(\frac{m-1}{p-1}\right)\right).$$

Combining Lemma D.6 and Corollary D.4 yields this statement.

□

Lemma D.8. *For any $v \in \mathbb{R}^{p^m}$ such that $\|v\| = 1$ and $v \perp V_s$ (defined in Equation (D.1)), it holds that*

$$\mathop{\mathbb{E}}_{\sigma\sim\mathbb{U}_m}\left[\langle \psi_\sigma, v\rangle^2\right] \leq \frac{1}{p}\exp\left(\frac{m-1}{p-1}\right).$$

The main implication of this Lemma D.8 is that for any n-dimensional space $V \subset \mathbb{R}^{p^m}$ can not "cover" the vector space of all ψ_σ functions for different permutations $\sigma \in \mathbb{S}_p$. To prove this Lemma, we decompose the inner product $\langle v, h_\sigma\rangle$ to $d+1$ sums using the Multinomial Theorem. This decomposition is enabled through observing the fact that there are $d+1$ equivalence groups in the possible set of indices of v. Based on Lemma D.9, we can use the Binomial Theorem to decompose the expectation of the inner product as follows

$$\begin{aligned}
\mathop{\mathbb{E}}_{\sigma\sim\mathbb{U}_m}\left[\langle v, \psi_\sigma\rangle^2\right] &= \mathop{\mathbb{E}}_{\sigma\sim\mathbb{U}_m}\left[\left(\sum_x v(x)\psi_\sigma(x)\right)^2\right] \\
&= \mathop{\mathbb{E}}_{\sigma\sim\mathbb{U}_m}\left[\sum_{d=0}^{m} \sum_{\substack{x,x' \\ \mathsf{dist}(x,x')=d}} \psi_\sigma(x)\psi_\sigma(x')v(x)v(x')\right] \\
&= \sum_{d=0}^{m} C_d \sum_{\substack{x,x' \\ \mathsf{dist}(x,x')=d}} v(x)v(x') \qquad \text{(D.8)}
\end{aligned}$$

where $C_d \triangleq \mathbb{E}_{\sigma\sim\mathbb{U}_m}\left[\psi_\sigma(x)\psi_\sigma(x')\right]$ for any $(x,x') \in \mathcal{X}_d$.

To complete the proof, we now have to bound two terms. First, we need to show that the expectation $\mathbb{E}_{\sigma\sim\mathbb{U}_m}\left[\psi_\sigma(x)\psi_\sigma(x')\right]$ for each (x,x') in the same equivalence group is bounded. Next, we need to show that for each set $\mathcal{X}_d$, the sum $\sum_{(x,x')\in\mathcal{X}_d} v(x)v(x')$ is also bounded. These are correspondingly shown in Lemmas D.9 and D.10. Based on these two Lemmas, we can now present the proof of Lemma D.8.

Proof of Lemma D.8.

$$\begin{aligned}\mathop{\mathbb{E}}_{\sigma\sim\mathbb{U}_m}\left[\langle v,\psi_\sigma\rangle^2\right] &= \sum_{d=0}^{m} C_d \sum_{\substack{x,x'\\ \mathsf{dist}(x,x')=d}} v(x)v(x') \\ &= \sum_{d=0}^{m}\frac{1}{p^2}\left(1-\frac{1}{(1-p)^{d-1}}\right)(-1)^d\binom{m}{d} \\ &= \frac{1}{p}\left(1+\frac{1}{p-1}\right)^{m-1} \\ &\le \frac{1}{p}\exp\left(\frac{m-1}{p-1}\right). \end{aligned} \tag{D.9}$$

where the second to last step is due to the binomial Theorem and the last step is due to the fact that for all $x \in \mathbb{R}$, $1+x \le \exp(x)$.

□

Lemma D.9. *For any $(x,x') \in \mathcal{X}_d$ index vector pair it holds that*

$$\mathop{\mathbb{E}}_{\sigma\sim\mathbb{U}_m}[\psi_\sigma(x)\psi_\sigma(x')] = \frac{1}{p^2}\left(1-\frac{1}{(1-p)^{d-1}}\right).$$

Proof of Lemma D.9. Let us re-iterate the definition of h:

$$\psi_\sigma(x) = \mathbf{1}\Big(\sum_{i=1}^{m}\sigma_i(x_i) \equiv 0 \pmod p\Big).$$

First, note that when $d = \Delta(x,x') = \Delta(y,y') = 0$, it's clear that $\mathbb{E}_{\sigma\sim\mathbb{U}_m}[\psi_\sigma(x)\psi_\sigma(x')] = \mathbb{E}_{\sigma\sim\mathbb{U}_m}[\psi_\sigma(y)\psi_\sigma(y')]$ since there exists a permutation $\sigma' \in \mathbb{S}_m$ such that $x = \sigma'(y)$ and $x' = \sigma'(y')$ (as $x = x'$ and $y = y'$).

Next, for each pair (x',x') we define $I(x,x') \triangleq \{i : x[i] \neq x'[i]\}$ and $E(x,x') \triangleq \{i : x[i] = x'[i]\}$. Note that for each $(x,x') \in \mathcal{X}_d$ for any $1 \le d \le m$ it holds that

$$\begin{aligned}\mathop{\mathbb{E}}_{\sigma\sim\mathbb{U}_m}[\psi_\sigma(x)\psi_\sigma(x')] &= \mathop{\mathbb{E}}_{\sigma\sim\mathbb{U}_m}\left[\mathbf{1}\Big(\sum_{i=1}^{m}\sigma_i(x_i) \equiv \sum_{i=1}^{m}\sigma_i(x'_i) \equiv 0 \pmod p\Big)\right] \\ &= \mathop{\mathbb{E}}_{\sigma\sim\mathbb{U}_m}\left[\mathbf{1}\Big(\sum_{i\in I(x,x')}\sigma_i(x_i) + \sum_{i\in E(x,x')}\sigma_i(x_i) \equiv \sum_{i\in I(x,x')}\sigma_i(x'_i) + \sum_{i\in E(x,x')}\sigma_i(x_i) \equiv 0 \pmod p\Big)\right] \\ &= \mathop{\mathbb{E}}_{\sigma\sim\mathbb{U}_m}\left[\mathbf{1}\Big(\sum_{i\in I(x,x')}\sigma_i(x_i) \equiv \sum_{i\in I(x,x')}\sigma_i(x'_i) \equiv -\sum_{i\in E(x,x')}\sigma_i(x_i) \pmod p\Big)\right] \\ &= \mathop{\mathbb{E}}_{\substack{\sigma\sim\mathbb{U}_m\\ \zeta\sim\mathrm{Unif}([p])}}\left[\mathbf{1}\Big(\sum_{i\in I(x,x')}\sigma_i(x_i) \equiv \sum_{i\in I(x,x')}\sigma_i(x'_i) \equiv \zeta \pmod p\Big)\right] \\ &= \mathop{\mathbb{E}}_{\substack{(a,b)\sim\mathrm{Unif}(\mathcal{X}_{d,d})\\ \zeta\sim\mathrm{Unif}([p])}}\left[\mathbf{1}\Big(\sum_{i=1}^{d}a_i \equiv \sum_{i=1}^{d}b_i \equiv \zeta \pmod p\Big)\right] \end{aligned} \tag{D.10}$$

where in the second to last line, we replaced $\mathbb{E}_{\sigma\sim\mathbb{U}_m}\left[-\sum_{i\in E(x,x')}\sigma_i(x_i)\right]$ with $\mathbb{E}_{\zeta\sim\mathrm{Unif}([p])}[\zeta]$ (assuming $d \ge 1$).

Now we define

$$C_d \triangleq \mathop{\mathbb{E}}_{\sigma\sim\mathbb{U}_m}[\psi_\sigma(x)\psi_\sigma(x')] = \mathop{\Pr}_{\substack{\zeta\sim\mathrm{Unif}([p])\\ (y,y')\sim\mathrm{Unif}(\mathcal{X}_{d,d})}}\left[\mathbf{1}\left(\sum_{i=1}^{d}y_i \equiv \sum_{i=1}^{d}y'_i \equiv \zeta \pmod p\right)\right]. \tag{D.11}$$

It's easy to see that $C_0 = \frac{1}{p}$ (probability of ζ being 0) and $C_1 = 0$ (since $y_1 \neq y'_1$). We aim to find the closed form formula for the general $d \geq 2$. As $\zeta \sim \mathrm{Unif}([p])$ is independent of the two sums, we can absorb it and write (from here until the rest of the proof we drop $(\mathrm{mod}\ p)$ from equivalences for ease of presentation)

$$Q_d \triangleq \Pr_{(y,y')\sim \mathrm{Unif}(\mathcal{X}_{d,d})}\left[\mathbf{1}\left(\sum_{i=1}^{d} y_i \equiv \sum_{i=1}^{d} y'_i\right)\right] = p \cdot C_d. \tag{D.12}$$

Note that for $d \geq 2$ we have that

$$\begin{aligned} Q_d &= \Pr_{(y,y')\in \mathrm{Unif}(\mathcal{X}_{d,d})}\left[\mathbf{1}\left(\sum_{i=1}^{d-1} y_i \equiv \sum_{i=1}^{d-1} y'_i\right)\cdot \mathbf{1}(y_d = y'_d)\right] \\ &\quad + \Pr_{(y,y')\in \mathrm{Unif}(\mathcal{X}_{d,d})}\left[\mathbf{1}\left(\sum_{i=1}^{d-1} y_i \not\equiv \sum_{i=1}^{d-1} y'_i\right)\cdot \mathbf{1}\left(y_d \equiv y'_d + \sum_{i=1}^{d-1} y'_i - \sum_{i=1}^{d-1} y_i\right)\right] \\ &= (1 - Q_{d-1}) \cdot \frac{1}{p-1}. \end{aligned} \tag{D.13}$$

Hence for $d \geq 2$ we have that

$$\begin{aligned} C_d &= \frac{1}{p(p-1)}(1 - p \cdot C_{d-1}) \\ &= \frac{1}{p(p-1)} - \frac{C_{d-1}}{p-1} \\ &= \frac{1}{p^2}\left(1 - \frac{1}{(1-p)^{d-1}}\right). \end{aligned} \tag{D.14}$$

This completes the proof. □

Lemma D.10. *For any $\mathcal{X}_d$ where $d \in [m]$ and $v \in \mathbb{R}^{p^m}$ such that $\|v\|_2 = 1$ and $v \perp V_s$ (defined in Equation (D.1)) it holds that*

$$\sum_{(x,x')\in\mathcal{X}_d} v(x)v(x') = (-1)^d \binom{m}{d}.$$

Proof of Lemma D.10. Let us define $G(d)$ as

$$\begin{aligned} G(d) &\triangleq \frac{1}{\binom{m}{d}} \sum_{(x,x')\in\mathcal{X}_d} v(x)v(x') \\ &= \mathop{\mathbb{E}}_{\sigma\in\mathbb{S}_m} \sum_{y\in[p]^{m-d}} \sum_{(t,t')\in\mathcal{X}_{d,d}} v\big(\sigma(y\|t)\big)v\big(\sigma(y\|t')\big). \end{aligned} \tag{D.15}$$

Since $v \perp V_s$ we have that

$$\begin{aligned} 0 &= \sum_{y\in[p]^d} \sum_{(t,t')\in\mathcal{X}_{d-1,d-1}} \left(\sum_{s\in[p]} v\big(\sigma(y\|s\|t)\big)\right)\left(\sum_{s\in[p]} v\big(\sigma(y\|s\|t')\big)\right) \\ &= \sum_{(y\|s)\in[p]^{m-d+1}} \sum_{(t,t')\in\mathcal{X}_{d-1,d-1}} v\big(\sigma(y\|s\|t)\big)v\big(\sigma(y\|s\|t')\big) \\ &\quad + \sum_{y\in[p]^{m-d}} \sum_{(s\|t,s'\|t')\in\mathcal{X}_{d,d}} v\big(\sigma(y\|s\|t')\big)v\big(\sigma(y\|s'\|t')\big) \\ &= G(d-1) + G(d). \end{aligned} \tag{D.16}$$

Note that $G(0) = \sum_x v(x)^2 = \|v\|_2^2 = 1$. Hence, $G(d) = (-1)^d$. This completes the proof. □

E. Generalization Upper Bound for Regression

The original model is $f\left((e_i,e_j)^\top\right) = V(W(e_i,e_j)^\top)^{\odot 2}$. We consider the model $g\left((e_i,e_j,e_k)^\top\right) = \left\langle e_k, f\left((e_i,e_j)^\top\right)\right\rangle$ and the function class $\mathcal{H}$ is defined over g with different weights $\theta = (W,V)$ where $W \in \mathbb{R}^{h\times 2p}$ and $V \in \mathbb{R}^{p\times h}$. For an input $x \in \mathbb{R}^{3p}$, we define two slices $x' \triangleq x[:2\mathrm{p}]$ and $x'' \triangleq x[2\mathrm{p}:]$. We also define $\mathcal{W}_{h,r} \triangleq \{W \in \mathbb{R}^{h\times 2p} : \|W\|_\infty \le r\}$ and $\mathcal{V}_{h,r} \triangleq \{V \in \mathbb{R}^{p\times h} : \|V\|_\infty \le r\}$ which we will use later to denote parameters of our function. We further define $\mathcal{D}_n = \{x_1, x_2, \cdots, x_n\}$ such that for all $a \in [n]$ we have $x_a = (e_i,e_j,e_k)^\top$ for some $i,j,k \in [p]$. For this section, we fix the set $\{x_1,x_2,\cdots,x_n\} \sim \mathrm{Unif}(\mathcal{X})$ and denote by $\mathcal{R}_n(\mathcal{H})$ the empirical Rademacher complexity of the function class $\mathcal{H}$ defined as

$$\mathcal{R}_n(\mathcal{H}) \triangleq \mathop{\mathbb{E}}_{\sigma\sim\mathrm{Unif}(\{\pm1\}^n)}\left[\sup_{\psi\in\mathcal{H}}\frac{1}{n}\sum_{i=1}^n \psi(x_i)\sigma_i\right] \tag{E.1}$$

where $\mathcal{H}$ maps $\mathcal{X} \to \mathbb{R}$ and $n \in \mathbb{N}$.

Lemma E.1. *Consider the function classes*

$$\mathcal{H}^w_{r,r'} \triangleq \left\{\psi : \mathbb{R}^{3p} \to \mathbb{R} \mid \exists\left[W^\top \in \mathcal{W}_{w,r} \wedge V \in \mathcal{V}_{w,r'}\right] \text{ s.t. } \psi(x) = \left\langle x'', V\langle W,x'\rangle^2\right\rangle\right\} \tag{E.2}$$

and

$$\mathcal{G}_r \triangleq \left\{g : \mathbb{R}^{3p} \to \mathbb{R} \mid \exists\left[U \in \mathbb{R}^{4\times 3p} \wedge \|U\|_\infty \le r\right] \text{ s.t. } g(x) = \sum_{i=1}^4 \langle U_i^\top, x\rangle^3\right\} \tag{E.3}$$

where U_i denotes the i'th row of U. The function class $\mathcal{H}^1_{r,r'}$ is contained in $\mathcal{G}_{\max(r,r')}$ (and hence $\mathcal{R}_n(\mathcal{H}_{r,r'}) \le \mathcal{R}_n(\mathcal{G}_{\max(r,r')})$).

Proof of Lemma E.1. We prove this lemma by showing that for each pair of matrices W,V of $\psi \in \mathcal{H}^1_{r,r'}$, we can construct a matrix U of $g \in \mathcal{G}_{\max(r,r')}$ such that for all $x \in \mathbb{R}^{3p}$, $h(x) = g(x)$. Consider an arbitrary parameterization W,V of h such that $\psi(x) = \left\langle x'', V\langle W,x'\rangle^2\right\rangle$. We can construct $U = \sqrt[3]{\frac{2}{9}}(Q_1,Q_2,Q_3,Q_4)^\top$ where

$$Q_1 = \begin{pmatrix} W \\ V\end{pmatrix}, \qquad Q_2 = \begin{pmatrix} -W \\ V\end{pmatrix}, \qquad Q_3 = \begin{pmatrix} -W/2 \\ V\end{pmatrix}, \qquad Q_4 = \begin{pmatrix} W/2 \\ -V\end{pmatrix}. \tag{E.4}$$

Observe that

$$\begin{aligned}
g(x) &= \sum_{i=1}^4 \langle U_i^\top, x\rangle^3 \\
&= \frac{2}{9}\left[\langle U_1,x\rangle^3 + \langle U_2,x\rangle^3 + \langle U_3,x\rangle^3 + \langle U_4,x\rangle^3\right] \\
&= \frac{2}{9}\left[(\langle W,x'\rangle + \langle V,x''\rangle)^3 - (\langle W,x'\rangle - \langle V,x''\rangle)^3 - \left(\frac{\langle W,x'\rangle}{2} + \langle V,x''\rangle\right)^3 + \left(\frac{\langle W,x'\rangle}{2} - \langle V,x''\rangle\right)^3\right] \\
&= \left\langle x'', V\langle W,x'\rangle^2\right\rangle \\
&= \psi(x).
\end{aligned} \tag{E.5}$$

It is straightforward to see that $\|U\|_\infty = \max(\|W\|_\infty, \|V\|_\infty)$, which completes the proof. □

Lemma E.2. *Consider the function class $\mathcal{G}_r$ defined in Equation (E.3). It holds that*

$$\mathcal{R}_n(\mathcal{G}_r) \le \frac{12r^3\sqrt{p}}{\sqrt{n}}.$$

Proof of Lemma E.2. We can derive the Rademacher complexity of $\mathcal{G}_r$ as

$$\begin{aligned}\mathcal{R}_n(\mathcal{G}_r) &= \mathop{\mathbb{E}}_{\sigma\sim\mathrm{Unif}(\{\pm1\}^n)}\left[\sup_{U\in\mathbb{R}^{4\times 3p},\|U\|_\infty\le r}\frac{1}{n}\sum_{a=1}^n\sigma_a\sum_{c=1}^4\langle U_c^\top,x_a\rangle^3\right]\\ &\le \mathop{\mathbb{E}}_{\sigma\sim\mathrm{Unif}(\{\pm1\}^n)}\left[\sup_{U\in\mathbb{R}^{3p},\|U\|_\infty\le r}\frac{4}{n}\sum_{a=1}^n\sigma_a\langle U,x_a\rangle^3\right]\\ &\le \mathop{\mathbb{E}}_{\sigma\sim\mathrm{Unif}(\{\pm1\}^n)}\left[\sup_{U\in\mathbb{R}^{3p},\|U\|_\infty\le r}\frac{12r^2}{n}\sum_{a=1}^n\sigma_a\langle U,x_a\rangle\right]\end{aligned}\tag{E.6}$$

where we used Talagrand's contraction principal using $3r^2$-Lipschitzness of $\langle U,x_a\rangle^3$ with respect to $\langle U,x_a\rangle$, and we can continue:

$$\begin{aligned}&\le \mathop{\mathbb{E}}_{\sigma\sim\mathrm{Unif}(\{\pm1\}^n)}\left[\sup_{U\in\mathbb{R}^{3p},\|U\|_\infty\le r}\frac{12r^2}{n}\|U\|_2\left\|\sum_{a=1}^n\sigma_a x_a\right\|_2\right]\\ &\le \frac{12r^3\sqrt{3p}}{n}\mathop{\mathbb{E}}_{\sigma\sim\mathrm{Unif}(\{\pm1\}^n)}\left[\left\|\sum_{a=1}^n\sigma_a x_a\right\|_2\right]\\ &\le \frac{36r^3\sqrt{p}}{\sqrt{n}}.\end{aligned}\tag{E.7}$$

where the last step follows from Rademacher complexity of linear model and the fact that $\|x\|_2=\sqrt{3}$ for all $x\in\mathcal{D}_n$ (Awasthi et al., 2020) . □

Remark E.3. *Lemma E.2 directly implies the following Rademacher complexity bound for the function class $\mathcal{H}^1_{r,r'}$ defined in Equation (E.2):*

$$\mathcal{R}_n(\mathcal{H}^1_{r,r'})\le\frac{36\sqrt{p}}{\sqrt{n}}\max(r,r')^3.$$

Lemma E.4.

$$\mathcal{R}_n(\mathcal{H}^h_{r,r'})\le\frac{36h\sqrt{p}}{\sqrt{n}}\max(r,r')^3.$$

Proof of Lemma E.4.

$$\begin{aligned}\mathcal{R}_n(\mathcal{H}^h_{r,r'}) &= \mathop{\mathbb{E}}_{\sigma\sim\mathrm{Unif}(\{\pm1\}^n)}\left[\sup_{\psi\in\mathcal{H}^h_{r,r'}}\frac{1}{n}\sum_{i=1}^n\sigma_i\psi(x_i)\right]\\ &= \mathop{\mathbb{E}}_{\sigma\sim\mathrm{Unif}(\{\pm1\}^n)}\left[\sup_{W\in\mathcal{W}_{h,r},V\in\mathcal{V}_{h,r'}}\frac{1}{n}\sum_{a=1}^n\sigma_a\left\langle e_{k_a},V\left(W\begin{pmatrix}e_{i_a}\\e_{j_a}\end{pmatrix}\right)^{\odot 2}\right\rangle\right]\\ &= \mathop{\mathbb{E}}_{\sigma\sim\mathrm{Unif}(\{\pm1\}^n)}\left[\sup_{W\in\mathcal{W}_{h,r},V\in\mathcal{V}_{h,r'}}\frac{1}{n}\sum_{a=1}^n\sigma_a\left\langle e_{k_a},\sum_{b=1}^h V_{:,b}\left(W_b\begin{pmatrix}e_{i_a}\\e_{j_a}\end{pmatrix}\right)^{\odot 2}\right\rangle\right]\\ &\le \mathop{\mathbb{E}}_{\sigma\sim\mathrm{Unif}(\{\pm1\}^n)}\left[\sup_{W\in\mathcal{W}_{1,r},V\in\mathcal{V}_{1,r'}}\frac{h}{n}\sum_{a=1}^n\sigma_a\left\langle e_{k_a},V\left\langle W,\begin{pmatrix}e_{i_a}\\e_{j_a}\end{pmatrix}\right\rangle^2\right\rangle\right]\\ &= h\mathcal{R}_n(\mathcal{H}^1_{r,r'})\end{aligned}\tag{E.8}$$

where in the second to last line we used the fact that the Rademacher complexity of a two-layer NN with h hidden neurons is bounded by h times that of a single-hidden-neuron counterpart. Thus, applying Remark E.3 we can conclude that

$$\mathcal{R}_n(\mathcal{H}^h_{r,r'})\le\frac{36h\sqrt{p}}{\sqrt{n}}\max(r,r')^3.$$

□

We are now ready to present the proof of the sample complexity upper bound for one-hidden-layer networks in the regression task. We first present the following Theorem from Srebro et al. (2010) on bounding the excess risk of H-smooth loss functions.

Theorem E.5 (Theorem 1 from Srebro et al. (2010))**.** *For an H-smooth non-negative loss ℓ such that for all $x, y, \psi, |\ell(\psi(x), y)| \leq b$, for any $\delta > 0$ we have with probability at least $1 - \delta$ over a random sample size of n, for any $\psi \in \mathcal{H}$,*

$$L(\psi) \leq \hat{L}(\psi) + K\left(\sqrt{\hat{L}(\psi)}\left(\sqrt{H}\log^{1.5} n\mathcal{R}_n(\mathcal{H}) + \sqrt{\frac{b\log(1/\delta)}{n}}\right) + H\log^3 n\mathcal{R}_n^2(\mathcal{H}) + \frac{b\log(1/\delta)}{n}\right)$$

where K is a positive constant, $L(h)$ denotes the population loss of h according to ℓ and $\hat{L}(h)$ denotes the loss of h on the mentioned sample of size n according to ℓ.

We now present the proof of Proposition 3.7.

Proposition 3.7. *Choose $R, \delta > 0$ to be positive constants. For any $\mathcal{D}_{\text{train}}$ of size n and $\theta^* \in \{\theta = (W, V) : \mathcal{L}(g, \theta, \mathcal{D}_{\text{train}}) = 0 \wedge \|\theta\|_\infty \leq R\}$, there exists a positive constant $C > 0$ such that with probability at least $1 - \delta$ over the choice of $\mathcal{D}_{\text{train}}$,*

$$\mathcal{L}_{\ell_2}(g(\theta^*; \cdot)) \leq \frac{CR^6h^2}{n}\left(p\log^3 n + \log\frac{1}{\delta}\right).$$

Proof. Consider the function class $\mathcal{H}_{R,R}^h$ for whom we have already proved a Rademacher complexity upper bound. As $\|\theta\|_\infty \leq R$, and all the inputs are one-hot, for all $x = (e_i, e_j, e_k)$ and $g \in \mathcal{H}_{R,R}^h$ it holds that $g(x) \leq 4hR^3$. This boundedness accordingly implies smoothness of ℓ_2 loss on this function class with $H = 1$. Hence, Theorem E.5 directly applies to our function class, yielding:

$$\mathcal{L}_{l_2}(g(\theta, \cdot)) \leq \frac{CR^6h^2}{n}\left(p\log^3 n + \log(1/\delta)\right)$$

for some positive constant C independet of other values. □

Finally, we remark that if the ℓ_2 loss is small enough, then the misclassification error is guaranteed to be zero.

Proposition E.6. *Consider a predictor $g : \mathcal{X} \to \mathbb{R}$. The population misclassification error is upper-bounded by $2\mathcal{L}_{\ell_2}(g)/p$.*

Proof. Note that each $(x, y) \in \mathcal{X} \times \mathcal{Y}$ that is misclassified induces an ℓ_2 loss of at least $p^2/2$. To see that why, for each pair (e_a, e_b) to be misclassified while attaining minimum possible ℓ_2 loss we need

$$\begin{aligned} g((e_a, e_b, e_c)) &< \frac{p}{2} \\ g((e_a, e_b, e_d)) &> \frac{p}{2} \\ g((e_a, e_b, e_k)) &= 0 \quad \text{for all } k \notin \{c, d\} \end{aligned}$$

where $c = a + b \pmod p$, $d \in [p] \neq c$ and k. Hence, each of (e_a, e_b, e_c) and (e_a, e_b, e_d) introduce an ℓ_2 loss of at least $p^2/4$. □

F. Construction of Interpolating Solution with Small ℓ_∞ Norm

Proposition 3.8. *Let the set of models with zero population loss be $\Theta^* \triangleq \{\theta \mid \mathcal{L}_{\ell_2}(g(\theta; \cdot)) = 0\}$. For any $p \geq 2$ and $h \geq 8p$, Θ^* is nonempty and $\min_{\theta \in \Theta^*} \|\theta\|_\infty \leq \left\lfloor \frac{h}{8p} \right\rfloor^{-\frac{1}{3}}$.*

In this section, we prove Proposition 3.8. We present a construction of weights that interpolates the dataset for $h = 8p$. Then we generalize this result to any $h \geq 8p$ by duplicating the first $8p$ neurons $\left\lfloor \frac{h}{8p} \right\rfloor$ times, where each copy is $\left\lfloor \frac{h}{8p} \right\rfloor^{-\frac{1}{3}}$ times smaller in magnitude.

Proof of Proposition 3.8. We begin by constructing 8 matrices of size $p \times 2p$ denoted by $W^{(i)}$.

$$
\begin{aligned}
W^{(1)}_{k,n} &= \cos\left(\frac{2\pi k}{p}n\right) & W^{(1)}_{k,m+p} &= \cos\left(\frac{2\pi k}{p}m\right) \\
W^{(2)}_{k,n} &= \cos\left(\frac{2\pi k}{p}n\right) & W^{(2)}_{k,m+p=} &- \cos\left(\frac{2\pi k}{p}m\right) \\
W^{(3)}_{k,n} &= \sin\left(\frac{2\pi k}{p}n\right) & W^{(3)}_{k,m+p} &= \sin\left(\frac{2\pi k}{p}m\right) \\
W^{(4)}_{k,n} &= \sin\left(\frac{2\pi k}{p}n\right) & W^{(4)}_{k,m+p} &= -\sin\left(\frac{2\pi k}{p}m\right) \\
W^{(5)}_{k,n} &= \sin\left(\frac{2\pi k}{p}n\right) & W^{(5)}_{k,m+p} &= \cos\left(\frac{2\pi k}{p}m\right) \\
W^{(6)}_{k,n} &= \sin\left(\frac{2\pi k}{p}n\right) & W^{(6)}_{k,m+p} &= -\cos\left(\frac{2\pi k}{p}m\right) \\
W^{(7)}_{k,n} &= \cos\left(\frac{2\pi k}{p}n\right) & W^{(7)}_{k,m+p} &= \sin\left(\frac{2\pi k}{p}m\right) \\
W^{(8)}_{k,n} &= -\cos\left(\frac{2\pi k}{p}n\right) & W^{(8)}_{k,m+p} &= \sin\left(\frac{2\pi k}{p}m\right)
\end{aligned}
\tag{F.1}
$$

$$
V_{q,\,8k:8(k+1)} = \begin{pmatrix}
\cos\left(\frac{2\pi k}{p}q\right) \\
-\cos\left(\frac{2\pi k}{p}q\right) \\
-\cos\left(\frac{2\pi k}{p}q\right) \\
\cos\left(\frac{2\pi k}{p}q\right) \\
\sin\left(\frac{2\pi k}{p}q\right) \\
-\sin\left(\frac{2\pi k}{p}q\right) \\
\sin\left(\frac{2\pi k}{p}q\right) \\
-\sin\left(\frac{2\pi k}{p}q\right)
\end{pmatrix}^{\top} \tag{F.2}
$$

Each $W^{(i)}$ for $1 \le i \le 8$ is a $p \times 2p$ matrix, whose elements are given by the equations presented above. Hence, in each equation $k, n, m \in [p]$. The construction of the first layer is based on stacking $W^{(i)}$ for $1 \le i \le 8$ to construct $W \in \mathbb{R}^{8p \times 2p}$. The weights of the second layer are given in Equation (F.2), where $V_{q,\,8k:8(k+1)}$ presents a slice of the second layer and $q, k \in [p]$.

To show that this construction solves the modular addition problem analytically, we will analytically perform the inference step for two arbitrary inputs n, m where $x = (e_n, e_m)$. We denote $h = (Wx)^{\odot 2} \in \mathbb{R}^{8p}$ as the post-activations of the first layer, which is given by

$$
h_{8k:8(k+1)} = \begin{pmatrix}
\cos\left(\frac{2\pi k}{p}n\right) + \cos\left(\frac{2\pi k}{p}m\right) \\
\cos\left(\frac{2\pi k}{p}n\right) - \cos\left(\frac{2\pi k}{p}m\right) \\
\sin\left(\frac{2\pi k}{p}n\right) + \sin\left(\frac{2\pi k}{p}m\right) \\
\sin\left(\frac{2\pi k}{p}n\right) - \sin\left(\frac{2\pi k}{p}m\right) \\
\sin\left(\frac{2\pi k}{p}n\right) + \cos\left(\frac{2\pi k}{p}m\right) \\
\sin\left(\frac{2\pi k}{p}n\right) - \cos\left(\frac{2\pi k}{p}m\right) \\
\cos\left(\frac{2\pi k}{p}n\right) + \sin\left(\frac{2\pi k}{p}m\right) \\
\cos\left(\frac{2\pi k}{p}m\right) - \sin\left(\frac{2\pi k}{p}n\right)
\end{pmatrix}^{2}. \tag{F.3}
$$

Note that for each k, we have that (after dropping (e_n, e_m) for simplicity)

$$
h_{8k} - h_{8k+1} = 2\cos\left(\frac{2\pi k}{p}(n+m)\right) + 2\cos\left(\frac{2\pi k}{p}(n-m)\right) \tag{F.4}
$$

and

$$
h_{8k+2} - h_{8k+3} = 2\cos\left(\frac{2\pi k}{p}(n-m)\right) - 2\cos\left(\frac{2\pi k}{p}(n+m)\right) \tag{F.5}
$$

and

$$
h_{8k+4} - h_{8k+5} = 4\sin\left(\frac{2\pi k}{p}n\right)\cos\left(\frac{2\pi k}{p}m\right) \tag{F.6}
$$

and

$$
h_{8k+6} - h_{8k+7} = 4\cos\left(\frac{2\pi k}{p}n\right)\sin\left(\frac{2\pi k}{p}m\right). \tag{F.7}
$$

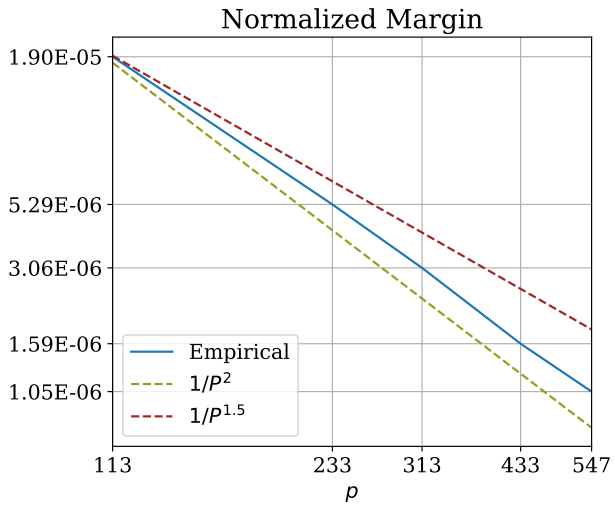

Figure 6: Normalized margin for different values of p after training with ce loss for 500,000 steps of normalized GD.

Hence,

$$h_{8k} - h_{8k+1} - h_{8k+2} + h_{8k+3} = 4\cos\left(\frac{2\pi k}{p}(n+m)\right) \tag{F.8}$$

and

$$h_{8k+4} - h_{8k+5} + h_{8k+6} - h_{8k+7} = 4\sin\left(\frac{2\pi k}{p}(n+m)\right). \tag{F.9}$$

Using the fact that $\cos(a-b) = \cos(a)\cos(b) - \sin(a)\sin(b)$, we can see that

$$\begin{aligned}
[f(e_n, e_m)]_q = \langle V_{q,:}, h\rangle &= 4\sum_{k=0}^{p-1} \cos\left(\frac{2\pi k}{p}q\right)\cos\left(\frac{2\pi k}{p}(n+m)\right) - \sin\left(\frac{2\pi k}{p}q\right)\sin\left(\frac{2\pi k}{p}(n+m)\right) \\
&= 4\sum_{k=0}^{p-1} \cos\left(\frac{2\pi k}{p}(m+n-q)\right) \\
&= 4p\,\mathbf{1}\big((m+n-q) \bmod p = 0\big)
\end{aligned} \tag{F.10}$$

where the last equality follows from Euler's identity and needs p to be odd. □

Remark F.1. *Assuming p is odd, we need at most $4p$ hidden neurons to interpolate the modular addition task.*

Observing the fact that $\cos(2\pi - a) = \cos(a)$, we can see that

$$\sum_{k=0}^{p-1} \cos\left(\frac{2\pi k}{p}(m+n-q)\right) = 1 + 2\sum_{k=1}^{\frac{p-1}{2}} \cos\left(\frac{2\pi k}{p}(m+n-q)\right) \tag{F.11}$$

where we replaced $\cos\left(\frac{2\pi 0}{p}(m+n-q)\right)$ with 1. Based on Equation (F.11), we can cut out half of the weights of the first and second layer, and only construct the frequencies up to $\frac{p-1}{2}$, which results in only needing $4p$ hidden neurons to construct the interpolating solution.

G. Margin-Based Generalization Bound for Classification

We begin by providing some background and notation on sub-exponential random variables, which will be later used in the proof of our margin-based generalization bound.

G.1. Background on sub-exponential variables

The following proofs rely heavily on concentration inequalities for sub-exponential random variables; we will first review some background on these quantities.

A real-valued random variable X with mean μ is called *sub-exponential* (see e.g. Wainwright, 2019) if there are non-negative parameters (ν, α) such that

$$\mathbb{E}[e^{\lambda(X-\mu)}] \leq e^{\frac{\nu^2\lambda^2}{2}} \quad \text{for all } |\lambda| < \frac{1}{\alpha}. \tag{G.1}$$

We use $X \sim SE(\nu, \alpha)$ to denote that X is a sub-exponential random variable with parameters (ν, α), but note that this is not a particular distribution.

One famous sub-exponential random variable is the product of the absolute value of two standard normal distributions, $z_i \sim \mathcal{N}(0, 1)$, such that the two factors are either independent ($X_1 = |z_1||z_2| \sim SE(\nu_p, \alpha_p)$ with mean $2/\pi$) or the same ($X_2 = z^2 \sim SE(2, 4)$ with mean 1). We now present a few lemmas regarding sub-exponential random variables that will come in handy in the later subsections of the appendix.

Lemma G.1. *Assume X is sub-exponential with parameters (ν, α). It holds that the random variable sX where $s \in \mathbb{R}^+$ is also sub-exponential, but with parameters $(s\nu, s\alpha)$.*

Proof. Consider $X \sim SE(\nu, \alpha)$ and $X' = sX$ with $\mathbb{E}[X'] = s\,\mathbb{E}[X]$. Based on the definition of sub-exponential random variables

$$\begin{aligned}
&\mathbb{E}\left[\exp\left(\lambda(X-\mu)\right)\right] \leq \exp(\frac{\nu^2\lambda^2}{2}) \quad \text{for all } |\lambda| < \frac{1}{\alpha} \\
\implies &\mathbb{E}\left[\exp\left(\frac{\lambda}{s}(sX - s\mu)\right)\right] \leq \exp(\frac{\nu^2 s^2 \frac{\lambda^2}{s^2}}{2}) \quad \text{for all } |\frac{\lambda}{s}| < \frac{1}{s\alpha} \\
\xrightarrow{\lambda' = \frac{\lambda}{s}} &\mathbb{E}\left[\exp\left(\lambda'(X' - \mu')\right)\right] \leq \exp(\frac{\nu^2 s^2 \lambda'^2}{2}) \quad \text{for all } |\lambda'| < \frac{1}{s\alpha}
\end{aligned} \tag{G.2}$$

Defining $\alpha' = s\alpha$ and $\nu' = s\nu$ we see that $X' \sim SE(s\nu, s\alpha)$. □

Proposition G.2. *If all of the random variables X_i for $i \in [N]$ for $N \in \mathbb{N}^+$ are sub-exponential with parameters (ν_i, α_i), and all of them are independent, then $\sum_{i=1}^N X_i \in SE(\sqrt{\sum_{i=1}^N \nu_i^2}, \max_i \alpha_i)$, and $\frac{1}{N}\sum_{i=1}^N X_i \sim SE\left(\frac{1}{\sqrt{N}}\sqrt{\frac{1}{N}\sum_{i=1}^N \nu_i^2}, \frac{1}{N}\max_i \alpha_i\right)$.*

Proof. This is a simplification of the discussion prior to equation 2.18 of Wainwright (2019). □

Proposition G.3. *For a random variable $X \sim SE(\nu, \alpha)$, the following concentration inequality holds:*

$$\Pr\left(|X - \mu| \geq t\right) \leq 2\exp\left(-\min\left(\frac{t^2}{2\nu^2}, \frac{t}{2\alpha}\right)\right).$$

Proof. The proof is straightforward from multiplying the result derived in Equation 2.18 of Wainwright (2019) by a scalar. □

Corollary G.4. *Consider $X \sim SE(\nu, \alpha)$, the following bound holds with probability at least $1 - \delta$:*

$$|X - \mu| < \max\left(\nu\sqrt{2\log\frac{2}{\delta}}, 2\alpha\log\frac{2}{\delta}\right).$$

A sub-Gaussian random variable, $SG(\nu)$, is one which satisfies (G.1) for all λ, i.e. it is the limit of $SE(\nu, \alpha)$ as $\alpha \to 0$.

Proposition G.5 (Chernoff bound)**.** *If X is $SG(\nu)$, then with probability at least $1 - \delta$, $|X - \mu| \leq \nu\sqrt{2\log\frac{2}{\delta}}$.*

Proposition G.6 (Hoeffding's inequality)**.** *If $X_1, \ldots, X_n$ are independent variables with means μ_i and each $SG(\nu_i)$, then $|\sum_{i=1}^n X_i - \sum_{i=1}^n \mu_i| \leq \sqrt{2\left(\sum_{i=1}^n \nu_i^2\right)\log\frac{2}{\delta}}$ with probability at least $1 - \delta$.*

G.2. Generalization Bound

We are now ready to state the main theorem for proving an upper bound on the number of training points needed to generalize.

Theorem G.7. *Let $f : \mathbb{R}^{2p} \to \mathbb{R}^p$ be a one-hidden-layer network with $h = \mathcal{O}(p)$ hidden neurons and quadratic activation, parameterized by $\theta = vec\,(W, V)$ where $W \in \mathbb{R}^{h\times 2p}$ and $V \in \mathbb{R}^{p\times h}$. For any $\delta > 0$, it holds with probability at least $1 - \delta$ over the choice of training set $\mathcal{D}_{train}$ of size m that, for all θ satisfying **1)** $\|W\|_\infty = \mathcal{O}(1)$, **2)** $\|V\|_\infty = \mathcal{O}(1)$ and **3)** $\sum_i^h V_{qi} = 0$ for all $q \in [p]$,*

$$L_0(f, \mathcal{D}) \le L_p(f, \mathcal{D}_{train}) + \tilde{\mathcal{O}}\left(\sqrt{\frac{p^{5/3}}{m}}\right). \tag{G.3}$$

Proof. We'll start by obtaining high probability bounds over the output logits of the network after perturbing each scalar weight with $\mathcal{N}(0, \sigma^2)$ noise. Let x represent the two-hot vector corresponding to inputs m and n and f_q represnet the q'th output logit of the neural network f. $\tilde{W}$ and $\tilde{V}$ denote the perturbation noises. Moreover, assume $C_V = \|V\|_\infty$ and $C_W = \|W\|_\infty$ are positive constants.

$$\begin{aligned}\tilde{f}_q\left(W + \tilde{W}, V + \tilde{V}; x\right) &= \left(V_q + \tilde{V}_q\right)\left((W + \tilde{W})\,x\right)^{\odot 2} \\ &= f_q\left(W, V; x\right) + V_q\left((\tilde{W}x)^{\odot 2} + 2\,Wx \odot \tilde{W}x\right) + \tilde{V}_q\left((W + \tilde{W})\,x\right)^{\odot 2}\end{aligned} \tag{G.4}$$

Starting with the first term, we can see that

$$\begin{aligned}V_q\left(\tilde{W}x\right)^{\odot 2} = \sum_{i=1}^h V_{qi}\left(\tilde{W}_{im} + \tilde{W}_{in}\right)^{\odot 2} &= 2\sigma^2 \sum_{i=1}^h V_{qi}\,\chi_i \\ &\sim \sqrt{2}\sigma^2 C_V SE(2\sqrt{h}, 4)\end{aligned} \tag{G.5}$$

where $\left(\tilde{W}_{im} + \tilde{W}_{in}\right)^{\odot 2}$ is distributed as a chi-squared random variable denoted by χ_i (as the second power of sum of two i.i.d Gaussian random variables each having a variance of σ^2). $\chi^2(1)$ is a sub-exponential random variable with parameters $SE(2, 4)$ with mean 1. Based on assumption **3**, we can see that the sum has a zero mean . We can further apply Corollary G.4 to show that with probability at least $1 - \delta_1$ over randomness of perturbation

$$\left|V_q\left(\tilde{W}x\right)^{\odot 2}\right| \le \sigma^2 \max\left(4C_V\sqrt{h\log\frac{2}{\delta_1}}, 4\sqrt{2}\log\frac{2}{\delta_1}\right). \tag{G.6}$$

For the second term, we can see that

$$2V_q\,Wx \odot \tilde{W}x = 2\sum_{i=1}^h V_{qi}(W_{im} + W_{in})(\tilde{W}_{im} + \tilde{W}_{in}) = 2\sqrt{2}\sigma\sum_{i=1}^h V_{qi}(W_{im} + W_{in})\,\mathcal{N}_i \tag{G.7}$$

where $(\tilde{W}_{im} + \tilde{W}_{in})$ is distributed as a Gaussian with $\mathcal{N}(0, 2)$ parameters and is replaced with $\sqrt{2}\mathcal{N}_i$ where $\mathcal{N}_i \sim \mathcal{N}(0, 1)$. Once again, we can see that this sum has a zero mean. Applying Proposition G.6 on this sum we can show that with probability at least $1 - \delta_2$ over randomness of perturbation

$$\left|2V_q\,Wx \odot \tilde{W}x\right| \le 2\sigma C_V C_W C_2\sqrt{2h\log\frac{2}{\delta_2}} \tag{G.8}$$

where C_2 is a positive constant.

Accordingly, we can decompose the second term into three sums. For the first component, we can see that

$$\tilde{V}_q\left(Wx\right)^2 \sim \mathcal{N}\left(0, \sigma^2\left\|(Wx)^2\right\|_2^2\right). \tag{G.9}$$

Applying the Proposition G.5 on this Gaussian random variable, one can see that with probability at least $1 - \delta_3$ over randomness of perturbation

$$\left|\tilde{V}_q \left(Wx\right)^2\right| \leq 4\sigma\sqrt{h \log \frac{2}{\delta_3}} \tag{G.10}$$

and for the second term in the decomposition we can see that

$$2\tilde{V}_q \, Wx \odot \tilde{W}x = 2\sum_{i=1}^{h} \tilde{V}_{qi}(W_{im} + W_{in})(\tilde{W}_{im} + \tilde{W}_{in}) \sim 2\sqrt{2}\sigma^2 \sum_{i=1}^{h}(W_{im} + W_{in})\mathcal{N}_i^{(1)}\mathcal{N}_i^{(2)} \tag{G.11}$$

where $\mathcal{N}_i^{(1)}$ and $\mathcal{N}_i^{(2)}$ are random variables distributed as $\mathcal{N}(0, 1)$. Note that the sum has a zero mean, the product of two independent Gaussian distributions can be written as the sum of two chi-squared distributions. Applying this technique and Corollary G.4 we can see that with probability at least $1 - \delta_4$ over randomness of perturbation

$$\left|2\tilde{V}_q \, Wx \odot \tilde{W}x\right| \leq C_3 C_W \sigma^2 \max\left(\sqrt{h \log \frac{2}{\delta_4}}, \log \frac{2}{\delta_4}\right) \tag{G.12}$$

where C_3 is a positive constant. Finally, for the last term, we can show that,

$$\tilde{V}_q \left(\tilde{W}x\right)^2 = \sum_{i=1}^{h} \tilde{V}_{qi} \left(\tilde{W}_{im} + \tilde{W}_{in}\right)^2 \sim \sqrt{2}\sigma^3 \sum_{i=1}^{h} \mathcal{N}_i \chi_i \tag{G.13}$$

where χ_i is a random variable distributed according to $\chi^2(1)$ and $\mathcal{N}_i$ is a random variable distributed according to $\mathcal{N}(0, 1)$. The sum has mean zero. To bound this sum, we can first treat the chi-squared variables as bounded random variables with high probability, and pull them out of the sum. Then, we can apply the Hoeffding's inequality to bound the sum of Gaussians. Note that for each $y_i \sim \chi^2(1)$, we have that

$$\Pr\left[|y_i - 1| < C_4 \max\left(\sqrt{2 \log \frac{2}{\delta}}, \log \frac{2}{\delta}\right)\right] \geq 1 - \delta \tag{G.14}$$

where C_4 is a positive constant. Applying a union bound on all y_i for $i \in [h]$ we have that

$$\Pr\left[\forall i \in [h]; |y_i - 1| < C_4 \max\left(\sqrt{2 \log \frac{2h}{\delta}}, \log \frac{2h}{\delta}\right)\right] \geq 1 - \delta. \tag{G.15}$$

Pulling this out of the sum, we can see that

$$\Pr\left[\left|\tilde{V}_q \left(\tilde{W}\right)^2 x\right| \leq \sqrt{2} C_4' \sigma^3 \log \frac{2h}{\delta} \sum_{i=1}^{h} \mathcal{N}(0, 1)\right] \geq 1 - \delta \tag{G.16}$$

where C_4' is a positive constant. Applying Proposition G.6 to bound the sum of Gaussians, we can show that

$$\Pr\left[\left|\tilde{V}_q \left(\tilde{W}\right)^2 x\right| \leq \sqrt{2} C_4' \sigma^3 \log \frac{2h}{\delta} \sqrt{2h \log \frac{2}{\delta'}}\right] \geq 1 - \delta - \delta'. \tag{G.17}$$

Combining the two high probability events, we can conclude that with probability at least $1 - \delta_5$ over perturbation noise

$$\left|\tilde{V}_q \left(\tilde{W}x\right)^2\right| \leq C_4'' \sigma^3 \sqrt{h} \left(\log \frac{2h}{\delta_5}\right)^{3/2} \tag{G.18}$$

where C_4'' is a positive constant.

Applying a union bound on $\delta_1, \cdots, \delta_5$, and then another union bound on each logit, we can show that for each x,

$$\max_q \left|\tilde{f}_q(x) - f_q(x)\right| \leq \tilde{\mathcal{O}}\left(\sqrt{h}\sigma^3 \left(\log \frac{2h}{\delta}\right)^{3/2}\right) \tag{G.19}$$

with probability at least $1 - \delta$ over the perturbation noise.

Since p is the margin of solution achieved with W, V, applying Lemma 1 from Neyshabur et al. (2018) with $\sigma^2 = h^{\frac{1}{3}}$ concludes the proof.

□

Proposition G.8. *Condition **3** from Theorem G.7 which implies that* $\forall q, \sum_{i=1}^{h} V_{qi} = 0$ *is not necessary.*

Proof. Assume we have a network with weights $\theta = (W, V)$ that satisfies all conditions of Theorem G.7 except $\forall q, \sum_{i=1}^{h} V_{qi} = 0$. We can construct a new network with weights $\theta' = (W', V')$ such that $W' = \begin{bmatrix} W \\ W \end{bmatrix}$ and $V' = \begin{bmatrix} V & -V \end{bmatrix}$. This network has the same outputs as the original one with parameters θ, while each row in V' has a zero sum. Hence, Theorem G.7 shows that the constructed network follows the provided genearlization bound, which subsequently shows that the original network with parameters θ does so too, since the outputs of these two networks are exactly identical.

□