EPFL

Thesis n° 11255

Deep Neural Networks: Large-Width Behavior and Generalization Bounds

Presented on 26th February 2025

School of Basic Sciences
Chair of Statistical Field Theory
Doctoral program in Mathematics

for the award of the degree of Docteur ès Sciences (PhD)

by

Evgenii GOLIKOV

Accepted on the jury's recommendation

Prof. D. Kressner, jury president
Prof. C. Hongler, thesis director
Prof. M. Mondelli, examiner
Prof. A. Terenin, examiner
Prof. L. Chizat, examiner

École
polytechnique
fédérale
de Lausanne

2025

Acknowledgements

As many of analytically-minded people, I have never paid much attention to humanity studies. However, there was one phrase from a university professor who gave us a course of philosophy, which fell deep in my memory:

> For the moment, you are immortal. Of course, on a rational level, you are aware of your mortality, but you are unable to really feel this. However, as you become older, the clock behind your ears keeps haunting you, its ticks get only louder.

If I knew back then how right he was. I am dedicating my thesis to the ticking clock, loud and ubiquitous.

I would like to thank my advisor who took the duty of a ticking clock when I got stunned by the music. I am grateful to my parents who gave me birth early enough for the clock to have enough time to get close enough to my ears. I am also grateful to a brave woman from ODS Random coffee who woke me up from a perennial hybernation (at least for a while). Last but not least, I am grateful to Alexander Terenin who introduced me to Greg Yang, with whom we had a very fruitful collaboration, the result of which constitutes a large part of the present thesis. I have learned a lot from Greg, and became much more mature as a researcher by working with him.

I would also like to cite a small piece of poetry that follows me since my early undergrad years (I apologize for my awkward translation from Russian):

With a slanting rain	Ливнем косым
They will knock on our door	Постучатся в нашу дверь
Angry springtimes, cheerful armies...	Гневные вёсны, весёлые войска...
Once,	Однажды,
Just believe in that!	Только ты поверь!
The pendulum will move to the *right* side,	Маятник качнётся в правильную сторону,
And the time will be over...	И времени больше не будет...[1]

Perhaps more seriously, I would like to acknowledge one more person for whom the time got over too soon. Without Maxim Kretov I would never start studying Deep Learning theory in all its depth. This *duty* has never made me exhausted, and I cannot exhaust it myself:

Our ultimate duty is like a last cartridge in a gun,	Наше дело последнее, словно патрон,
Like the ever last feat,	Словно вечно последний подвиг,
Like an every single *last time*...	Словно всякий последний раз...[2]

[1] A piece from *Солнцеворот* by Yegor Letov.
[2] Another piece from the same poem.

Abstract

During several decades, neural network architectures have undergone considerable evolution: from shallow to deep, from fully connected to structured (e.g. convolutional, recurrent, or residual), from unnormalized to normalized. The present thesis contributes to *Deep Learning Theory*: a field of science studying the behavior of neural networks.

An essential part of the present work is devoted to *Tensor Programs* first introduced by Yang (2019a), a unifying formalism covering both training and inference for a vast class of neural network architectures. The theory of Tensor Programs is tightly related to the theory of *infinitely wide networks*, which we exhaustively overview in one of the chapters. In a direction orthogonal to Tensor Programs, we also present results on exact minima of L_2-regularized objective functions. Finally, we state a novel *a-priori* generalization bound for neural networks with activation functions close to being linear.

The cornerstone of the theory of Tensor Programs is the *Master Theorem*. This theorem states that any scalar generated by a Tensor Program (i.e. a loss or accuracy value at a given training timestep) converges to a deterministic limit given by a certain recurrence formula. This theorem generalizes several existing results in the theory of infinitely wide neural nets (Yang, 2019b, 2020a; Yang and Littwin, 2021), and recovers classical results in random matrix theory (Yang, 2020b).

In the present work, we generalize this result to non-Gaussian weight initializations, thus proving that the Master Theorem is *universal* with respect to the initial weight distribution. We also provide tail bounds for scalars generated by a Tensor Program; these tail bounds are crucial for analyzing convergence rates.

In addition, we provide a comprehensive survey on the theory of networks in a specific limit of infinite width (namely, in the NTK limit).

Apart from infinitely wide nets, this thesis also analyzes the minima of L_2-regularized loss functions for networks of finite width. One of the results we obtained is that the regularized loss value cannot be improved by adding new neurons after a certain finite threshold value which depends on the training dataset size.

Finally, we prove a novel generalization bound for networks with activation functions close to being linear. To the best of our knowledge, this bound is the first to be *non-vacuous* (i.e. it guarantees the distribution risk to be below that of a random guess model) and *a-priori* (i.e. it can be computed before the actual training of the model) at the same time.

We hope our work (1) demonstrates that neural networks exhibit universality phenomena ubiquitous in natural sciences, (2) helps to understand the behavior of infinitely wide networks better, (3) connects them to networks of finite width, (4) suggests when making a network wider does not improve performance, and (5) provides better generalization guarantees for neural networks.

Keywords: machine learning, neural networks, deep learning, deep learning theory, infinitely-wide networks, tensor programs, neural tangent kernel, tail bounds, loss landscape, generalization.

Contents

1 Introduction

The present thesis consists of three published papers (Golikov and Yang, 2022; Jacot et al., 2022; Golikov, 2025), one preprint (Golikov et al., 2022), and one work in progress. The original texts of these works can be found in Chapters 2 to 6.

In this introductory chapter, we start with a general overview of the topic, and present the thesis contributions as we go along. We also put a short summary of the contributions (chapter by chapter) in Conclusions, Chapter 7.

1.1 Evolution of neural network architectures

Since the introduction of the Perceptron by Rosenblatt (1958), neural networks architectures have undergone a long way of evolution:

- In 1965, Ivakhnenko et al. (1966) proposed stacking several layers on top of each other, thus giving rise to a "deep Perceptron". Today, most of neural networks used in practice for highly expressive data such as images or texts have more than one hidden layer.
- Fukushima (1988) was the first to explore the idea of weight sharing, inspired by biological neural networks (Hubel et al., 1959). This idea later developed into convolutional neural networks (LeCun et al., 1989), which still remain a de-facto standard for image processing tasks.
- Recurrent neural networks (RNN) first introduced in Rumelhart et al. (1986), also exploit the idea of weight sharing, but with the weights shared across layers, not inside the layer as in CNNs. Long Short-term Memory Networks (LSTM) introduced in Schmidhuber et al. (1997), is one of the notable extensions of RNNs; networks of this type were sucessfully applied to sequence processing tasks (e.g. text processing, or time series).
- As architectures started getting larger, the problem of *overfitting* became more and more vital. In 2014, a surprisingly simple technique called Dropout (Srivastava et al., 2014) was introduced. This technique fights overfitting by randomly masking neurons during training.
- In 2015, two important architectural modifications were introduced: skip-connections (He et al., 2016a) and Batch Normalization (Ioffe and Szegedy, 2015) later developed into Layer Normalization (Ba et al., 2016). The purpose of both was to simplify and accelerate training of deep architectures: up to 1000 layers (He et al., 2016b), which was hardly possible before.

- In 2014, Bahdanau et al. (2014) suggested an extension for a sequence-to-sequence architecture based on LSTMs, called Attention mechanism. Vaswani et al. (2017) were first to build an architecture for sequence processing tasks based solely on the Attention mechanism. This architecture, coined Transformer, is in the core of all modern Large Language Models (LLM), in particular, the GPTs (Radford et al., 2018, 2019; Brown et al., 2020; Achiam et al., 2023).

What we list above are cornerstones on a long process of search for improved performance on real practical tasks. This search was mostly performed by trial and error, motivated by intuition, or sometimes, in an automated way (with Neural Arhitecture Search, see Baymurzina et al. (2022)), resulting in architectures with no solid theoretical grounding. The present thesis is a contribution to *Theory of Deep Learning*, a discipline that aims to provide such a grounding.

The architectures we mentioned above are very complex, thus making a goal of constructing a unified theory seem too ambitious. Indeed, most of the theoretical results we mention later in the present thesis, are formulated primarily for multilayer Perceptrons, the architecture dating back to 60s. However, as we shall see later, *Tensor Programs*, first introduced in Yang (2019a), provide a unified framework for computations (including learning and inference) for *all* of the architectures discussed above.

1.2 Obstacles to Theory of Deep Learning

One of the main obstacles to constructing a solid Theory of Deep Learning is that the result of training a neural network is quite difficult to predict, or even to characterize in a mathematically tractable way. Before demonstrating this statement, let us consider the most basic neural network architecture possible, a linear model: $f_w(x) = w^\top x$, where $w, x \in \mathbb{R}^d$. Suppose we are aiming to optimize cumulative square error on a dataset consisting of m training samples $(x_k, y_k)_{k=1}^m$; this results in the following loss function:

$$\mathcal{L}(w) = \frac{1}{2}\sum_{k=1}^{m}(y_k - f_w(x_k))^2 = \frac{1}{2}\|y - X^\top w\|^2, \tag{1.1}$$

where the matrix $X \in \mathbb{R}^{d\times m}$ has training inputs $(x_k)_{k=1}^m$ as columns, and the vector $y \in \mathbb{R}^m$ consists of the corresponding targets.

The above loss function $\mathcal{L}(w)$ indicates the mismatch between the model's predictions $f_w(X)$ and the targets y on the training set. Learning is usually done by minimization of this functional. Typically, we are looking for a minimum using a local learning rule as gradient descent, or one of its variants (e.g. Adam (Kingma and Ba, 2015) and many others). Gradient descent searches for a local minimum by performing the following iteration:

$$w_{t+1} = w_t - \eta\nabla_w\mathcal{L}(w_t) = w_t + \eta X(y - X^\top w_t) = \eta Xy + \left(I - \eta XX^\top\right) w_t, \tag{1.2}$$

where η is called *learning rate*. It is easy to solve the above iteration explicitly. Assuming $w_0 = 0$, we arrive at

$$w_t = \eta\sum_{s=0}^{t-1}\left(I - \eta XX^\top\right)^s Xy = \left(I - \left(I - \eta XX^\top\right)^t\right) X^+ y, \tag{1.3}$$

where we also assumed $XX^\top$ to be invertible, or equivalently, $\operatorname{rk} X = d$. Here $X^+ = (XX^\top)^{-1}X$ is Moore–Penrose pseudoinverse of X. When $\eta \in (0, 2/\lambda_{max}(XX^\top))$, where λ_{max} is the maximal

eigenvalue of a symmetric matrix, the above iteration converges to $X^+ y$ as $t \to \infty$, which is the global minimum of $\mathcal{L}(w)$.

Linear nets. In the above case, the result of learning could be obtained explicitly in terms of training data (X, y) in a mathematically tractable way. However, when we switch to non-trivial networks, which have more than one layer, getting the result of learning in a mathematically tractable way becomes tricky. Indeed, consider a network with two layers and no nonlinear activation functions: $f_{v,W}(x) = v^\top W x$. Here $v \in \mathbb{R}^n$, $W \in \mathbb{R}^{n \times d}$, and $x \in \mathbb{R}^d$. The number of hidden neurons n is often called *network width*. Of course, such a network has no advantage over the linear model above in terms of expressivity. However, it exhibits much more intricate learning dynamics compared to its one-layer sibling. Consider a gradient flow, a gradient descent with continuous time:

$$\dot{v}_t = W_t X \left(y - X^\top W_t^\top v_t\right), \qquad \dot{W}_t = v_t \left(y - X^\top W_t^\top v_t\right)^\top X^\top. \tag{1.4}$$

This is a system of two ODEs, a vector and a matrix one, of order three each; no general analytic solution is known for such systems.[1]

Nonlinear nets. The analysis of learning dynamics becomes even more intricate when nonlinear activation functions are present. This is because these functions are typically applied elementwise to hidden representations of neural networks. Applying a scalar function to each entry of a vector is an operation with no clear linear-algebraic interpretation. Indeed, if we consider $f_{v,W}(x) = v^\top \phi(Wx)$, where the map $\phi \in C^1(\mathbb{R})$ is applied elementwise, the gradient flow dynamics becomes

$$\dot{v}_t = \phi(W_t X) \left(y - \phi(X^\top W_t^\top) v_t\right), \qquad \dot{W}_t = \left[\phi'(W_t X) \odot v_t \left(y - \phi(X^\top W_t^\top) v_t\right)^\top\right] X^\top. \tag{1.5}$$

In contrast to the linear dynamics Eq. (1.4) exhibiting only matrix products covered by linear algebra, the above dynamics mixes matrix products with Hadamard products and elementwise function applications. We are not aware of any mathematical tools which are able to handle such expressions.

1.3 Neural networks in the limit of infinite width

As a ubiquitous phenomenon in mathematics, the analysis of certain objects simplifies when limits are taken. We have already taken a limit of infinitesimal gradient steps, which simplified a discrete dynamics of gradient descent into a continuous dynamics of gradient flow expressed with an ODE Eqs. (1.4) and (1.5). Another fruitful limit is proven to be a limit when the width n approaches infinity, while input and output dimensions stay fixed. Depending on the initialization and parameterization, they give rise to a variety of different limiting dynamics (Golikov, 2020c,a; Yang and Hu, 2021), with two being special and extensively studied: NTK (Jacot et al., 2018) and mean-field (Mei et al., 2018; Rotskoff and Vanden-Eijnden, 2018; Chizat and Bach, 2018; Yarotsky, 2018; Nguyen, 2019a; Araújo et al., 2019; Sirignano and Spiliopoulos, 2020, 2022; Nguyen and Pham, 2023).

[1] Analytic solutions for some special initializations (see e.g. Saxe et al. (2013)), as well as approximate solutions (Tu et al., 2024), as well as solutions in limit cases (Chizat et al., 2024), do exist, but this topic is out of the scope of the present thesis.

1.3.1 NTK limit

Our contribution: a survey on the NTK theory. In the first case, the training dynamics becomes equivalent to that of a kernel method, with a special kernel coined *Neural Tangent Kernel* (NTK). One of the contributions of the present thesis is a comprehensive survey on the NTK theory, Golikov et al. (2022), reproduced in Chapter 2. Below we give a brief introduction to this theory.

Setup. Consider a generic parametric model $f_\theta(x)$ with parameters $\theta \in \mathbb{R}^N$ and a scalar output.[2] If we optimize a square loss $\mathcal{L}(\theta) = \frac{1}{2}\sum_{k=1}^{m}\left(y_k - f_\theta(x_k)\right)^2$ on a dataset of size m with gradient flow, we get the following evolution of model outputs:

$$\frac{df_{\theta_t}(x)}{dt} = \nabla_\theta^\top f_{\theta_t}(x)\dot{\theta}_t = \nabla_\theta^\top f_{\theta_t}(x)\nabla_\theta f_{\theta_t}(X)\left(y - f_{\theta_t}(X)\right) = \Theta_t(x, X)\left(y - f_{\theta_t}(X)\right), \tag{1.6}$$

where we defined

$$\Theta_t(x, x') = \nabla_\theta^\top f_{\theta_t}(x)\nabla_\theta f_{\theta_t}(x'). \tag{1.7}$$

For any fixed t, Θ_t is a positive semi-definite (PSD) function; such functions are called kernels. Specifically, Θ_t is called *(empirical) Neural Tangent Kernel (NTK)* of the model f_{θ_t}.

The model output evolution Eq. (1.6) resembles that of a *kernel gradient flow*:

$$\frac{dg_t(x)}{dt} = K(x, X)\left(y - g_t(X)\right), \qquad g_0 \equiv 0, \tag{1.8}$$

where K is a kernel. It can be easily integrated as follows:

$$g_t(X) = \left(I - e^{-tK(X,X)}\right)y, \qquad g_t(x) = K(x, X)\left(K(X, X)\right)^{-1}\left(I - e^{-tK(X,X)}\right)y, \tag{1.9}$$

where we assumed $K(X, X)$ to be positive definite. The crucial feature of Eq. (1.6) is that the kernel Θ_t is not constant, but evolves over training time as the network learns. This prevents us from writing the exact solution immediately as we did for the kernel gradient flow.

Limit dynamics. The crucial advantage of considering the evolution in the model output space Eq. (1.6) over that in the parameter space $\dot{\theta}_t = -\nabla_\theta \mathcal{L}(\theta_t)$ is that the former gives a chance of formulating a clear limit as the number of parameters N grows to infinity, while the latter does not. In other words, $\lim_{N\to\infty}\theta_t$ makes little sense since $\theta \in \mathbb{R}^N$, however, $\lim_{N\to\infty} f_{\theta_t}(x)$ might make sense for any x since $f_{\theta_t}(x) \in \mathbb{R}$.

As demonstrated by Jacot et al. (2018), when f_θ is a fully-connected network under a specific (non-standard) parameterization with weights initialized from iid standard Gaussians, Θ_t converges pointwise in probability to a deterministic limit which *does not depend on* t, as the numbers of neurons in hidden layers $n_1, \ldots, n_L$ go to infinity sequentially:

$$\lim_{n_L\to\infty} \ldots \lim_{n_1\to\infty} \Theta_t(x, x') = \bar{\Theta}(x, x') \quad \text{in prob.} \quad \forall x, x' \in \mathbb{R}^d \quad \forall t \geq 0. \tag{1.10}$$

The kernel $\bar{\Theta}$ is called *(limiting) NTK*. Since it does not depend on the training time anymore, training the model becomes equivalent to a kernel gradient flow in the limit:

$$\lim_{n_L\to\infty} \ldots \lim_{n_1\to\infty} f_{\theta_t}(x) = \bar{\Theta}(x, X)\left(\bar{\Theta}(X, X)\right)^{-1}\left(I - e^{-t\bar{\Theta}(X,X)}\right)y \quad \text{in prob.} \quad \forall x \in \mathbb{R}^d \quad \forall t \geq 0. \tag{1.11}$$

[2] A large class of neural networks with scalar output falls into this category. A notable exception could be networks with Batch Norm (Ioffe and Szegedy, 2015), whose output during training depends only on the training batch.

Our contribution: a non-Gaussian Master Theorem. As we shall see later, one of the main results of the present thesis, *non-Gaussian Master Theorem*, Theorem 1.4.2, allows one to prove a stronger (almost sure) convergence for both the kernel and the model output, for virtually all modern neural network architectures (Yang, 2020a; Yang and Littwin, 2021), and a wide class of popular weight initializations (e.g. for the one of He et al. (2015)).

Transferring kernel method properties to neural nets. A rich literature concerning kernel methods makes the NTK limit especially compelling: one could potentially transfer results from kernel methods to neural networks. A high-level approach would be to first, characterize the deviation of a real, finite-width neural network from its NTK limit, and then, formulate the result for a kernel method with the limit NTK as a kernel, taking the finite-width deviation into account. As particular examples of this approach, Du et al. (2019a,b); Song and Yang (2019); Oymak and Soltanolkotabi (2020); Bombari et al. (2022) prove that gradient descent converges exponentially to a global minimum for wide enough networks; this property is far from obvious given non-convexity and high complexity of the loss landscape of neural nets (see e.g. Choromanska et al. (2015); Pennington and Bahri (2017)). Another property that could be derived from the NTK stability is generalization ability (Arora et al., 2019a); we are going to discuss our contributions to the generalization theory later in Section 1.8.

The downsides of this approach are that (a) it requires the network to be unrealistically wide (the width has to scale polynomially with the number of data points for the empirical NTK to be close enough to the limit one: e.g. Oymak and Soltanolkotabi (2020) guarantee that $nd = \Omega(m^2)$ is enough), and (b) in many real networks, the empirical NTK is demonstrated to be not stable (as in e.g. Fort et al. (2020)). There are two reasons for NTK instability: first of all, the parameterization required for the limit NTK to be stable differs from the one typically used in practice. Second, even if one applies the NTK parameterization, the width has to be really large for the NTK to stabilize. See our survey, Golikov et al. (2022), for details.

In addition, our current results on tail bounds which we discuss later in Section 1.5 suggest that width required for the NTK to converge grows with depth and the number of training steps. This, in its turn, suggests that a simultaneous infinite width and depth limit (as studied e.g. in Hanin and Nica (2020b)) could be a more adequate approximation for realisic neural nets, compared to an infinite width but fixed depth limit, which leads to the NTK analysis.

Disadvantages of a stable kernel. Looking from another side, the fact that the NTK of a finite-width model does evolve allows the model to learn features. Indeed, any kernel could be represented as a scalar product in some Hilbert space $\mathcal{H}$: $K(x, x') = \langle \Phi(x), \Phi(x') \rangle$, where $\Phi : \mathbb{R}^d \to \mathcal{H}$. When $\mathcal{H}$ is finite-dimensional, the kernel gradient flow is nothing but a normal gradient flow for a model defined as $g_\theta(x) = \langle \theta, \Phi(x) \rangle$. This is a linear model in the feature space $\mathcal{H}$. For example, $\Phi_t(x) = \nabla_\theta f_{\theta_t}(x)$ for $K = \Theta_t$. If Θ_t changes then Φ_t necessarily changes; hence the features are learned.

The process of *feature learning* is crucial for understanding the high empirical performance of neural networks. One of the simplest models exhibiting this phenomenon is a linear network (Saxe et al., 2013; Tu et al., 2024). As a step forward towards realistic nonlinear networks, Bordelon et al. (2025) study a linear network over fixed high-dimensional nonlinear features, thus mixing a random feature model and a linear network.

1.3.2 Mean-field limit

In the so-called mean-field case, the learning dynamics of a network is described by an evolution of measure. Consider again a network with one hidden layer and a scalar output, and scale its output by $1/n$:

$$f_{v,W}(x) = \frac{1}{n}\sum_{\alpha=1}^{n} v_\alpha \phi(w_\alpha^\top x) = \int v\phi(w^\top x)\, d\hat{\mu}(v,w), \tag{1.12}$$

where we expressed the output as an integral over the *neuronal measure* $\hat{\mu}(v,w) = \frac{1}{n}\sum_{\alpha=1}^{n} \delta(v - v_\alpha) \otimes \delta(w - w_\alpha)$ for δ being a Dirac measure. The above expression became possible for two reasons: (1) we scaled the output by $1/n$, and (2) hidden neurons were permutationally invariant.

Suppose we initialize the weights from iid standard Gaussians: $v_0 \sim \mathcal{N}(0, I_n)$, $W_0 \sim \mathcal{N}(0, I_n \otimes I_d)$. Then we get a meaningful limit of the model output at initialization as n grows to infinity:

$$\lim_{n\to\infty} f_{v_0,W_0}(x) = f_{\mu_0}(x), \qquad f_\mu(x) = \int v\phi(x^\top w)\, d\mu(v,w) \tag{1.13}$$

almost surely by the Law of Large Numbers; here $\mu_0(v,w) = \mathcal{N}(0, I_{d+1})$ is a Gaussian measure on a $d+1$-dimensional space.

As one could guess, the learning dynamics transforms into an evolution of measure μ_t in this limit. Indeed, consider the gradient flow for finite n:

$$\dot{v}_{t,\alpha} = \frac{\eta}{n}\phi(w_{t,\alpha}^\top X)(y - f_{v_t,W_t}(X)), \qquad \dot{w}_{t,\alpha} = \frac{\eta}{n} X\left[\phi'(X^\top w_{t,\alpha}) \odot (y - f_{v_t,W_t}(X))\right] v_{t,\alpha}. \tag{1.14}$$

As we see, the evolution of each neuron depends only on this neuron itself (as we have no indices except for α in the above ODEs), and on the whole neuronal measure $\hat{\mu}_t(v,w) = \frac{1}{n}\sum_{\alpha=1}^{n} \delta(v - v_{t,\alpha}) \otimes \delta(w - w_{t,\alpha})$ (hidden inside $f_{v_t,W_t}(X)$). This allows us to rewrite the above evolution in terms of the evolution of measure $\hat{\mu}_t(v,w)$:

$$\partial_t \hat{\mu}_t = \frac{\eta}{n}\Psi_*(\hat{\mu}_t; \hat{\mu}_t), \qquad \Psi(v,w;\mu) = \begin{pmatrix} \phi(w^\top X)\,(y - f_\mu(X)) \\ X\left[\phi'(X^\top w) \odot (y - f_\mu(X))\right] v \end{pmatrix}. \tag{1.15}$$

Here $\Psi_*(\hat{\mu}_t; \hat{\mu}_t)$ denotes a *push-forward measure*. That is, given a measurable function $g : \mathcal{X} \to \mathcal{Z}$ and a measure μ on $\mathcal{X}$, we define a push-forward measure $g_*(\mu)$ on $\mathcal{Z}$ as follows: for any measurable $A \subset \mathcal{Z}$, we take $g_*(\mu)(A) = \mu(g^{-1}(A))$.

Taking $\eta = n$ yields a meaningful limit evolution as $n \to \infty$:

$$\partial_t \mu_t = \Psi_*(\mu_t; \mu_t), \qquad \mu_0(v,w) = \mathcal{N}(0, I_{d+1}). \tag{1.16}$$

Advantages over the NTK limit. One of the advantages of the mean-field limit is that the associated NTK Θ_t evolves over time even in the limit of $n \to \infty$ in contrast to the NTK limit. As was noted before, the phenomenon of a moving NTK is called feature learning and associated to the advantage of neural nets over kernel methods. Indeed, the empirical NTK is given by

$$\Theta(x,x') = \frac{1}{n^2}\sum_{\alpha=1}^{n} \left(\phi(w_\alpha^\top x)\phi(w_\alpha^\top x') + v_\alpha^2 \phi'(w_\alpha^\top x)\phi'(w_\alpha^\top x') x^\top x'\right). \tag{1.17}$$

Since all weights stay finite as $n \to \infty$ (as they follow a measure evolution), $\Theta(x, x')$ scales as n^{-1}. However, note that the evolution of model outputs $f(x)$ is governed by $\eta\Theta(x, X)$, and $\eta = n$. This gives a finite limit for the scaled NTK:

$$\lim_{n\to\infty} (\eta\Theta_t(x, x')) = \int \left(\phi(w^\top x)\phi(w^\top x') + v^2\phi'(w^\top x)\phi'(w^\top x')x^\top x'\right)\, \mu_t(dv, dw). \tag{1.18}$$

As the measure μ_t moves following Eq. (1.16), the above limit also evolves over time.

Disadvantages. First of all, while kernel methods give an explicit mathematically tractable solution Eq. (1.9), it is not clear what Eq. (1.16) converges to (but there exist some results on global convergence of the loss function $\mathcal{L}$ (Mei et al., 2018; Chizat and Bach, 2018)). Second, while the NTK limit is very general (applicable to a wide class of architectures expressible with Tensor Programs (Yang, 2020a; Yang and Littwin, 2021)), generalizing the above construction even to fully-connected nets with more than one hidden layer is not straightforward. The difficulty stems from the fact that the hidden neurons for deep nets are no longer permutationally invariant. This property was crucial for expressing the limit as a measure. Nevertheless, several reasonable constructions have been proposed (Nguyen, 2019a; Araújo et al., 2019; Sirignano and Spiliopoulos, 2022; Nguyen and Pham, 2023).

Yang and Hu (2021) proposed another meaningful limit called μP, which stands for "μ-paramaterized" and conincides with the mean-field limit for two-layered fully-connected nets; therefore it could also be seen as its generalization. This limit comes naturally from expressing the training routine as a Tensor Program, by ensuring all appearing terms to neither vanish, nor diverge. A remarkable property of this limit is that it preserves the optimal hyperparameters, which makes it useful for hyperparameter transfer from narrow nets to wide ones (Yang et al., 2021). However, the μP limit does not allow for a convenient form of a measure evolution, and it is still hard to study analytically (see Yang et al. (2022) for an analytically computable simplification for deep fully-connected nets).

One of the cases when a μP limit does become tractable is *linear networks*, i.e. networks with activation functions being identity. As demonstrated by Chizat et al. (2024), the training dynamics of a μ-parameterized infinitely wide linear network with a Gaussian initialization is in a one-to-one correspondence with that of another infinitely wide linear network with a specific deterministic initialization. Using this correspondence, Chizat et al. (2024) proved that when trained on a linearly explicable dataset[3], such a network converges to a minimal l_2-norm interpolator. This phenomenon is called *implicit bias*, and is out of the scope of the present thesis.

1.4 Tensor Programs

As was mentioned in Section 1.1, neural network architectures have undergone a long evolution since simple fully-connected designs. Tensor programs, first introduced in Yang (2019a), are a framework for expressing computations, including training and inference, for a very wide class of neural network architectures, in a unified way. The cornerstone theoretical result for Tensor Programs is *the Master Theorem*, which provides explicit limits for all scalars generated by a Tensor Program (e.g. loss and accuracy values at any training step), when *width* grows to infinity. This theorem covers both the NTK and the mean-field limit discussed in Section 1.3, as special cases. One of the main

[3]I.e. a dataset (X, y) with targets induced by a linear map: $y = X^\top w$ for some vector w.

contributions of the present thesis is a generalzation of the Master Theorem to non-Gaussian weight initializations, see Chapter 3.

1.4.1 Motivation

We start with a motivation for the framework of Tensor Programs. On a high level, training a neural network with a gradient-based method is nothing but the following iteration, run in a loop until the loss on the training dataset stops decreasing:

1. Forward pass on the training dataset (computing loss values);
2. Backward pass on the training dataset (computing loss gradients with respect to weights);
3. Weight update.

In its turn, doing a forward pass for a fully-connected network is nothing but the following iteration:

1. Take an input x;
2. Multiply it by a matrix;
3. Apply an activation function elementwise to the resulting vector;
4. Go to step 2 unless convergence criteria (e.g. low loss) are met.

It is easy to check that the backward pass of a fully-connected network is also an alteration between matrix multiplications and applying elementwise maps. What is more intriguing is that any given number of gradient descent steps could also be expressed this way; we will give an illustrative example below.

1.4.2 Definition

The above motivates the following construction. A Tensor Program works with the following types of variables: $n \times n$ matrices, vectors with n entries, and scalars. It takes some of them as inputs, and generates new ones by applying the following operations:

1. **Trsp:** $W \in \mathbb{R}^{n\times n} \to W^\top \in \mathbb{R}^{n\times n}$;
2. **MatMul:** $W \in \mathbb{R}^{n\times n},\ x \in \mathbb{R}^n \to Wx \in \mathbb{R}^n$;
3. **Nonlin:** $\phi : \mathbb{R}^{k+l} \to \mathbb{R},\ g^{1:k} \in \mathbb{R}^n,\ c^{1:l} \in \mathbb{R} \to \phi(g^{1:k}, c^{1:l}) \in \mathbb{R}^n$;
4. **Moment:** $\phi : \mathbb{R}^{k+l} \to \mathbb{R},\ g^{1:k} \in \mathbb{R}^n,\ c^{1:l} \in \mathbb{R} \to \frac{1}{n}\sum_{\alpha=1}^n \phi(g_\alpha^{1:k}, c^{1:l}) \in \mathbb{R}$.

The first two operations are simply a matrix transposition and a matrix-vector multiplication. The third is a generalized version of an elementwise nonlinearity: it takes k vector arguments (instead of one), and scalar arguments. The result could be illustrated schematically as follows:

$$\begin{pmatrix} g_1^1 \\ g_2^1 \\ \dots \\ g_n^1 \end{pmatrix}, \begin{pmatrix} g_1^2 \\ g_2^2 \\ \dots \\ g_n^2 \end{pmatrix}, \dots, \begin{pmatrix} g_1^k \\ g_2^k \\ \dots \\ g_n^k \end{pmatrix}, c^1, c^2, \dots, c^l \to \begin{pmatrix} \phi(g_1^1 & g_1^2 & \dots & g_1^k & c^1 & c^2 & \dots & c^l) \\ \phi(g_2^1 & g_2^2 & \dots & g_2^k & c^1 & c^2 & \dots & c^l) \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \phi(g_n^1 & g_n^2 & \dots & g_n^k & c^1 & c^2 & \dots & c^l) \end{pmatrix}. \tag{1.19}$$

We will see why this generalization is necessary in the examples below. The last operation generates a scalar by taking the average entry of the vector illustrated above.

1.4.3 Examples

Forward pass. Consider a fully-connected network with two hidden layers and a scalar output:

$$h^1 = W^1 x + b^1, \quad x^1 = \sigma(h^1), \qquad h^2 = W^2 x^1 + b^2, \quad x^2 = \sigma(h^2), \qquad f(x) = \frac{v^\top x^2}{n} + u. \quad (1.20)$$

Note that we intentionally normalize the output over n in order for it to be expressed as a scalar in a Tensor Program: Moment is the only operation which generates scalars, and it averages vector components. This scaling is very relevant to the μP limit that we have already mentioned above.

A Tensor Program expressing its very first forward pass will look as follows:

- Input matrices: W^2;
- Input vectors: $W^1 x$, b^1, b^2, v;
- Input scalars: u.
- Body:
  1. $x^1 := \sigma(W^1 x + b^1)$ (Nonlin: $\phi(g^1, g^2) = \sigma(g^1 + g^2)$);
  2. $h^2 := W^2 x^1$ (MatMul);
  3. Return $v^\top \sigma(h^2 + b^2)/n + u$ (Moment: $\phi(g^1, g^2, g^3, c^1) = g^1 \sigma(g^2 + g^3) + c^1$).

Here we already see why we need multiple vector arguments, as well as scalar arguments, for nonlinearities. Multiple arguments are needed because we have neither a vector addition, nor a scalar product, among our set of operations. A scalar argument appears because the output layer bias is a scalar.

Note also that it is $W^1 x$ that appears as an input vector, not x. The reason is that all vectors in the program have to be of the same size n. We are going to set this n to infinity in the Master Theorem we will introduce shortly, while d, the dimensionality of network inputs x, naturally stays fixed[4].

Vector outputs and input batches. Generalizing the above setting to a vector output is straightforward: it suffices to consider k output bias scalars $u^1, \dots u^k$ instead of one, and k output weight vectors $v^1, \dots v^k$ instead of one. It is also easy to write a program that processes a batch of inputs $x^{(1)}, \dots, x^{(B)}$: it is going to be a program that is a union of B identical programs processing each input in the batch. As long as we process whole batches, we could apply *Batch Normalization* (Ioffe and Szegedy, 2015), a technique that normalizes the output of each neuron using the first and the second order statistics from the batch. As activation functions are usually placed after

[4]There is a huge body of research considering the situation when both d and n grow to infinity proportionally. This situation could be covered by the framework of Tensor Programs as well, but we do not discuss it in the present thesis.

batchnorms, we could express the composition of the two using a single Nonlin operation:

$$\phi(g, g^1, \ldots, g^B) = \sigma\left(g + \frac{g^b - \frac{1}{B}\sum_{b'=1}^B g^{b'}}{\sqrt{\frac{1}{B}\sum_{b'=1}^B (g^{b'})^2 - \left(\frac{1}{B}\sum_{b'=1}^B g^{b'}\right)^2}}\right). \tag{1.21}$$

Weights as higher-order tensors. In some architectures, tensors of higher order are used. A canonical example is a convolutional neural network for image processing, where hidden representations are order-three tensors of size $H \times W \times n$, where H and W are image dimensions. These hidden representations are processed with convolutional filters which are tensors of order four of size $p \times q \times n \times n$, where p and q are filter dimensions[5]. Fortunately, higher order tensors do not pose any issues as the extra dimensions do not grow with n; therefore each hidden representation could be expressed as a set of HW vectors of size n, while each convolutional filter could be expressed as a set of pq matrices of size n.

Note also that nothing precludes us from using the same matrix in a Tensor Program several times: this is necessary in architectures with shared weights, like Recurrent Neural Nets. We will also have to reuse matrices when expressing backward passes, as we will see shortly. See also Yang (2019b) for more examples of Tensor Programs expressing forward passes for different architectures.

Backward pass. Let us come back to our toy model with two hidden layers and a scalar output. Let y be the target and $\ell(\cdot, \cdot)$ be the loss function. A backward pass through this network is given by the following equations:

$$\delta^2 := n\frac{\partial \ell(f(x), y)}{\partial x^2} = v\ell'(f(x), y), \qquad g^2 := n\frac{\partial \ell(f(x), y)}{\partial h^2} = \delta^2 \odot \sigma'(h^2), \tag{1.22}$$

$$\delta^1 := n\frac{\partial \ell(f(x), y)}{\partial x^1} = (W^2)^\top g^2, \qquad g^1 := n\frac{\partial \ell(f(x), y)}{\partial h^1} = \delta^1 \odot \sigma'(h^1). \tag{1.23}$$

As we see, these are backward passes where matrix transpositions naturally appear. Here we scale all the gradients by n in order for the variables to be expressible as vectors in a Tensor Program, similarly to what we did for the forward pass. The corresponding Tensor Program is given below:

- Input matrices: W^2;
- Input vectors: $W^1 x$, b^1, b^2, v;
- Input scalars: u.
- Body:
  1. $\hat{y} := \{\text{forward pass}\}$;
  2. $g^2 := v\ell'(\hat{y}, y) \odot \sigma'(h^2)$ (Nonlin: $\phi(g^1, g^2, c^1) = g^1 \ell'(c^1, y)\sigma'(g^2)$);
  3. $\tilde{W}^2 := (W^2)^\top$ (Trsp);
  4. $\delta^1 := \tilde{W}^2 g^2$ (MatMul);
  5. $g^1 := \delta^1 \odot \sigma'(h^1)$ (Nonlin: $\phi(g^1, g^2) = g^1 \sigma'(g^2)$);
  6. Return g^1, δ^1, g^2.

[5] Usually, $p = q$.

Weight update. As the reader has probably noticed already, the only operation in our toolset which generates new matrices is Trsp, which obviously cannot generate a new matrix after one gradient descent step. However, at each step, the update for each weight matrix has at most rank B for B being the batch size. Therefore the update is always expressible as a finite sum of vector-vector outer products. As a result, the action of an updated weight matrix is always expressible with a single MatMul operation (the action of the initial weight matrix), and a finite number of Moment operations (used to express the actions of the update), as we will illustrate shortly.

If we update our weights using only one training sample (x, y) (this is known as *stochastic gradient descent (SGD)*) then the very first gradient update is given by

$$v' = v - \eta x^2, \qquad W^{2,\prime} = W^2 - \eta \frac{g^2 (x^1)^\top}{n}, \qquad W^{1,\prime} = W^1 - \eta g^1 x^\top, \tag{1.24}$$

where η is a learning rate, and we have not updated biases for the sake of simplicity. Clearly, both W^1 and W^2 experience a rank-one update. While we cannot generate the updated matrix $W^{2,\prime}$ in our program explicitly, we can still express a forward pass with the updated weights:

$$x^{1,\prime} = \sigma(W^{1,\prime} x + b^1), \qquad W^{1,\prime} x = W^1 x - \eta g^1 (x^\top x), \tag{1.25}$$

$$x^{2,\prime} = \sigma(W^{2,\prime} x^{1,\prime} + b^1), \qquad W^{2,\prime} x^{1,\prime} = \underbrace{W^2 x^{1,\prime}}_{\text{MatMul}} - \eta g^2 \underbrace{\frac{(x^1)^\top x^{1,\prime}}{n}}_{\text{Moment}}, \tag{1.26}$$

$$f'(x) = \frac{(v')^\top x^{2,\prime}}{n} + b = \frac{v^\top x^{2,\prime}}{n} - \eta \frac{(x^2)^\top x^{2,\prime}}{n} + b. \tag{1.27}$$

As a result, the model output after one weight update is still expressible as a Tensor Program; the same holds for any given number of gradient descent steps if we perform this reasoning inductively, as demonstrated in Yang and Littwin (2021).

From here, the reason why we decided to scale the learning rate η over n for W^2, becomes clear: this was for the forward pass with updated weights to become expressible as a Tensor Program. This is the only natural scaling for Tensor Programs for which all of the generated scalars (such as $\frac{(x^1)^\top x^{1,\prime}}{n}$) neither diverge, nor vanish, as n grows to infinity. The infinite n limit is called μP, and is given by the main limit theorem, the Master Theorem, which we discuss shortly.

1.4.4 Master Theorem

The following theorem gives limits of all scalars a program generates, in an explicit form:

Theorem 1.4.1 ((Gaussian) Master Theorem, Yang (2019a, 2020b))**.** *Consider a Tensor Program with M vectors $g^1, \dots, g^M \in \mathbb{R}^n$ and scalars $c^1, \dots, c^M$. Suppose*

1. *All initial vectors $g^1, \dots, g^{M_0}$ have iid entries from $\mathcal{N}(0, 1)$;*
2. *All matrices A^i have iid entries from $\mathcal{N}(0, n^{-1})$;*
3. *All the nonlinearities ϕ are pseudo-Lipschitz;*[6]

[6] A function $f : \mathbb{R}^n \to \mathbb{R}^m$ is called pseudo-Lipschitz if there exist $C, d > 0$ such that for any $x, y \in \mathbb{R}^n$, $\|f(x) - f(y)\| \le C\|x - y\|(1 + \|x\|^d + \|y\|^d)$.

4. All initial scalars $c^1, \dots, c^{M_0}$ have almost sure limits as $n \to \infty$.

Then, as $n \to \infty$, for any $i \in [M]$,

$$c^i \xrightarrow{a.s.} \mathring{c}^i, \tag{1.28}$$

where $\mathring{c}^i$ is a deterministic scalar given by an explicit recurrence formula.

Our contribution: a non-Gaussian Master Theorem. As we see, the above theorem assumes all matrices to be Gaussian which does not match with modern practice: for example, uniform weight initialization is much more common (He et al., 2015). In our paper Golikov and Yang (2022) reproduced in Chapter 3, we prove its generalization to non-Gaussian weight initializations:

Theorem 1.4.2 (Non-Gaussian Master Theorem, Golikov and Yang (2022))**.** *Consider a Tensor Program with M vectors $g^1, \dots, g^M \in \mathbb{R}^n$ and scalars $c^1, \dots, c^M$. Suppose*

1. *All initial vectors $g^1, \dots, g^{M_0}$ have iid entries from $\mathcal{N}(0, 1)$;*
2. *All matrices A^i have iid entries with zero mean, variance n^{-1}, and each k-th moment bounded by $\nu_k n^{-k/2}$;*
3. *All the nonlinearities ϕ are polynomially smooth[7];*
4. *All initial scalars $c^1, \dots, c^{M_0}$ have almost sure limits as $n \to \infty$ and all moments.*

Then, as $n \to \infty$, for any $i \in [M]$,

$$c^i \xrightarrow{\text{a.s. \& } L^p} \mathring{c}^i \quad \forall p \in [1, \infty) \tag{1.29}$$

for the same $\mathring{c}^i$ as in the Gaussian theorem, Theorem 1.4.1.

While our theorem works for a smaller class of nonlinearities (necessarily smooth, which was not the case for Theorem 1.4.1), it allows for non-Gaussian matrices, and proves a stronger notion of convergence: not only almost sure, but also in L^p for any $p \in [1, \infty)$, which covers most of meaningful notions of convergence of random variables.

Distribution of vectors affects the limit. Our new theorem states that the distribution of matrices does not affect the limit. In this sense, the limit is *universal*, which resembles universality phenomena in physics, e.g. *the Maxwell-Boltzmann distribution* (Boltzmann, 1872). However, the distribution of vector variables could affect the limit. This might seem be counter-intuitive on the first sight since making all matrix entries non-Gaussian requires changing $O(n^2)$ parameters, while making entries of some vector variables non-Gaussian requires changing only $O(n)$ of them.

In order to see this, take any distribution ζ with zero mean and unit variance whose fourth moment does not equal the fourth moment of $\mathcal{N}(0, 1)$, which is 3. Then the almost sure limit of $\frac{1}{n} \sum_{\alpha=1}^n \phi_\alpha(g^1)$ for $\phi(x) = x^4$ and $g^1_\alpha \sim \zeta$ is not 3, which would be if $g^1_\alpha \sim \mathcal{N}(0, 1)$.

[7] We call f polynomially smooth if it is smooth and each (possibly mixed) derivative of order $k \geq 0$ is polynomially bounded.

Relationship to the Law of Large Numbers. Since scalars are generated only with a Nonlin operation, one could express the Master Theorem claim as

$$c^i = \frac{1}{n}\sum_{\alpha=1}^{n} \phi^i(g_\alpha^1, \dots, g_\alpha^{i-1}, c^1, \dots, c^{i-1}) \xrightarrow{\text{a.s.}} \mathring{c}^i, \tag{1.30}$$

where we assumed the nonlinearity to depend on all previously generated vectors and scalars without loss of generality. The above expression clearly resembles the Law of Large Numbers (LLN) as it states that the average of some random variables almost surely converges to a number. However, while these random variables are marginally identically distributed, they are not independent. Moreover, even their marginal distribution depends on n.

On the other hand, the Master Theorem implies a form of LLN:

$$\frac{1}{n}\sum_{\alpha=1}^{n} \phi(g_\alpha^1) \xrightarrow{\text{a.s.}} \mathbb{E}_{\xi\sim\mathcal{N}(0,1)}\left[\phi(\xi)\right], \tag{1.31}$$

where ϕ is pseudo-Lipschitz. This form is strictly weaker than the traditional LLN since not all integrable random variables are expressible as images of a Gaussian under a pseudo-Lipschitz map. Specifically, finite-variance distributions with tails heavier than that of a Gaussian, e.g. exponential, cannot be expressed this way.

Relationship to the Central Limit Theorem. As mentioned in Section 1.4, the Gaussian Master Theorem, Theorem 1.4.1, resembles (and implies) the Law of Large Numbers (LLN) in some sense. As both Master Theorems, the Gaussian and the non-Gaussian, imply a form of the Law of Large Numbers, our non-Gaussian theorem also implies a form of the other cornerstone theorem of probability, the Central Limit Theorem (CLT).

Indeed, consider a Tensor Program that computes the following, and its corresponding limit:

$$\frac{1}{n}\sum_{\alpha=1}^{n} \psi\left(\left(W^1\phi(g^1)\right)_\alpha\right) \to \mathbb{E}_{\xi\sim\mathcal{N}(0,1)}\left[\psi\left(\sqrt{\frac{1}{n}\sum_{\beta=1}^{n}\phi^2(g_\beta^1)}\,\xi\right)\right] \tag{1.32}$$

almost surely and in L^p for any p. Convergence in L^1 implies convergence *of* expectations:

$$\mathbb{E}\left[\frac{1}{n}\sum_{\alpha=1}^{n} \psi\left(\left(W^1\phi(g^1)\right)_\alpha\right)\right] = \mathbb{E}\left[\psi\left(\sum_{\beta=1}^{n} W_{1\beta}^1\phi(g_\beta^1)\right)\right] \to \mathbb{E}_{\xi\sim\mathcal{N}(0,1)}\left[\psi\left(\sqrt{\frac{1}{n}\sum_{\beta=1}^{n}\phi^2(g_\beta^1)}\,\xi\right)\right]. \tag{1.33}$$

Now it suffices to take $\phi \equiv 1$ to conclude that

$$\mathbb{E}\left[\psi\left(\sum_{\beta=1}^{n} W_{1\beta}^1\right)\right] \to \mathbb{E}_{\xi\sim\mathcal{N}(0,1)}[\psi(\xi)]. \tag{1.34}$$

Since considering polynomially smooth test functions is enough to establish convergence in distribution, we conclude that $\sum_{\beta=1}^{n} W_{1\beta}^1$ converges to $\mathcal{N}(0,1)$ in distribution. This way, we get CLT for random variables with all moments existing.

The fact that the Gaussian and the non-Gaussian theorems give the same limit resembles Lindeberg's proof of CLT (see e.g. Chapter 9 of Dudley (2018)). Let $(\tilde{\xi}_\alpha)_{\alpha=1}^n$ be iid random variables with zero mean, unit variance, and a finite third moment. The goal is to show that their scaled sum, $Z_n := \frac{1}{\sqrt{n}} \sum_{\alpha=1}^n \tilde{\xi}_\alpha$, converges to $\mathcal{N}(0,1)$ in distribution. By definition, this means that $\lim_{n\to\infty} \mathbb{E}\left[\phi(Z_n)\right] = \mathbb{E}_{\xi\sim\mathcal{N}(0,1)}[\phi(\xi)]$ for any test function ϕ.

The idea behind Lindeberg's proof is to swap each $\tilde{\xi}_\alpha$ with $\xi_\alpha \sim \mathcal{N}(0,1)$. Since they differ only in the third moment, $\mathbb{E}\left[\phi(Z_n)\right]$ will change only by $O(n^{-3/2})$. Since we have to swap only n variables this way, the overall change is only $O(n^{-1/2})$, which goes to zero:

$$\mathbb{E}\left[\phi\left(\frac{1}{\sqrt{n}}\sum_{\alpha=1}^n \tilde{\xi}_\alpha\right)\right] = \mathbb{E}\left[\phi\left(\frac{1}{\sqrt{n}}\sum_{\alpha=1}^n \xi_\alpha\right)\right] + O(n^{-1/2}) = \mathbb{E}_{z\sim\mathcal{N}(0,1)}[\phi(z)] + O(n^{-1/2}). \quad (1.35)$$

Our Theorem 1.4.2 states a similar claim: swapping all matrix entries with samples from $\mathcal{N}(0, n^{-1})$ (as in the Gaussian theorem) does not affect the limit: they are the same for both Theorem 1.4.1 and Theorem 1.4.2. This hints that a plausible strategy of proving the latter might be by applying Lindeberg's priniple: altering matrix entries one by one, hoping that the cumulative error will vanish for large n. However, a naive application of this principle fails since we have many more variables to alter: $O(n^2)$ instead of $O(n)$.

1.4.5 Proof idea

The strategy we use in our work (Golikov and Yang, 2022) is to consider a parametric family of matrices:

$$A^i(t) = \tilde{A}^i \cos\left(\frac{\pi}{2}t\right) + A^i \sin\left(\frac{\pi}{2}t\right) \qquad \forall t \in [0,1], \quad (1.36)$$

for each i, where A^i is the matrix with non-Gaussian entries, as in the theorem claim, while $\tilde{A}^i$ is their Gaussian counterpart (having entries distributed iid from $\mathcal{N}(0, n^{-1})$). This parameterization induces parametric families of all objects generated by the Tensor Program at hand.

We proceed with proving the following bound:

$$\sup_{t\in[0,1]} \mathbb{E}\,|\dot{c}^i(t)|^p = O_{n\to\infty}(n^{-p/2}) \qquad \forall p \in \mathbb{N}, \quad (1.37)$$

valid for any $i \in [M]$, where the "dot" abbreviates d/dt. This bound informally reads as "$\dot{c}^i(t) = O(n^{-1/2})$ uniformly over t", which means that the scalars generated by a Tensor Program become affected only negligibly by altering the matrix entries for large n, as long as we preserve the first two moments.

The above bound is highly nontrivial to prove, and requires exploiting the inner structure of Tensor Programs. However, as soon as we have it, we can proceed as

$$\mathbb{E}\,|c^i(1) - c^i(0)|^p = \mathbb{E}\left|\int_0^1 \dot{c}^i(t)\,dt\right|^p \le \int_0^1 |\dot{c}^i(t)|^p\,dt = O(n^{-p/2}) \qquad \forall p \in \mathbb{N}, \quad (1.38)$$

which means that $c^i(1) - c^i(0)$ converges to zero in L^p for any p. In particular, the case of $p = 4$ implies also almost sure convergence by a standard application of the Borel-Cantelli lemma. Since we know that $c^i(0) = \tilde{c}^i \xrightarrow{\text{a.s.}} \mathring{c}^i$ by the Gaussian theorem, $c^i = c^i(1)$ has the same almost sure limit. The fact that $c^i(1) - \mathring{c}^i$ converges to zero in L^p follows from the fact it does almost surely, a standard truncation trick, and a moment bound for $c^i(t)$ we get as a by-product of our proof.

1.4.6 Applications

Both Master Theorems, the Gaussian and the non-Gaussian, have a number of applications. We list some of them below:

Neural Network Gaussian Process (NNGP) correspondence. In the seminal work of Neal (1995), it was shown that pre-activation outputs of hidden neurons in a shallow fully-connected neural network converge to a Gaussian process when width goes to infinity and weights are Gaussian (i.e. at initialization). This result was extended to deep networks in Lee et al. (2018), then to convolutional networks in Garriga-Alonso et al. (2019); Novak et al. (2019). The same result follows from the Master Theorem as shown in Yang (2019b).

Convergence to a kernel method. As we have previously discussed in Section 1.3, training a fully-connected network under gradient flow becomes equivalent to training a kernel method with kernel gradient flow (Jacot et al., 2018). The corresponding kernel is called Neural Tangent Kernel (NTK). Yang (2020a); Yang and Littwin (2021) demonstrated that the NTK evaluated at a pair of points is nothing but a scalar in a Tensor Program; therefore it converges by the Master Theorem. This result works for any architecture whose training process is expressible with a Tensor Program, and is stronger in a way that the Master Theorem states almost sure convergence, while Jacot et al. (2018) prove only convergence in probability. Note that the parameterization required for NTK to be stable over the course of training differs from μP — the natural parameterization for Tensor Programs. Under NTK parameterization, some of the scalars vanish in the limit; this requires a slight extension of the framework, see Yang and Littwin (2021) for details.

Limiting spectral distributions of random matrix ensembles. One could recover the well-known Wigner semi-cirle law (Wigner, 1958) and Marchenko–Pastur law (Marchenko and Pastur, 1967) for random matrix ensembles using the Master Theorem, as demonstrated in Yang (2020b).

Indeed, let the matrix A follow *Wishart ensemble*: $A = WW^\top$, where W is an $n \times n$ matrix with iid entries sampled from a zero-mean distribution of variance one. Marchenko and Pastur (1967) demonstrated that the spectrum of the random matrix A converges to a specific limit distribution as $n \to \infty$.

Since a compactly supported distribution is described uniquely with its moments, it suffices to compute the moments of the matrix A, $M_k = \frac{1}{n} \operatorname{tr} \mathbb{E}\left[A^k\right]$, in the limit of $n \to \infty$. To fit them into the framework of Tensor Programs, we use the following trick: $\operatorname{tr} C = \mathbb{E}_{v \sim \mathcal{N}(0, I_n)} v^\top C v$, which gives

$$M_k = \frac{1}{n} \mathbb{E}_{v \sim \mathcal{N}(0, I_n)} \mathbb{E}_W \left[v^\top (WW^\top)^k v \right]. \tag{1.39}$$

Expressing $\frac{1}{n}\left[v^\top (WW^\top)^k v\right]$ as a scalar in a Tensor Program is straightforward, and already its infinite n limit gives the correct moment. However, what we need is convergence of expectations, which does not a-priori given by the Master Theorem. Yang (2020b) proves an extension of the Master Theorem to L^1 convergence under stricter assumptions on nonlinearities (which hold in this case).

The Gaussian Master Theorem presented above works only for a Gaussian matrix W, while the original result of Marchenko and Pastur (1967) works for any distribution with zero mean and finite variance. Our non-Gaussian Master Theorem relaxes this limitation, as well as stating L^p convergence for any $p \in \mathbb{N}$ for a wider class of nonlinearities.

The strategy for proving the semi-circle law (Wigner, 1958) is similar: one has to take $A = \frac{1}{2}(W+W^\top)$ and proceed as above. This technique is potentially extensible to other matrix ensembles, like the one of (Pennington and Worah, 2017), $A = \frac{1}{n}Y^\top Y$ for $Y = \sigma(WX)$ and Gaussian W and X, as long as the moments are analytically computable and one could recover the distribution from its moments.

Free independence principle. In a number of works dealing with random matrices in the context of neural networks (e.g. in Pennington et al. (2017)), it was assumed that hidden representations become *freely independent* from weight matrices at initialization when n is large. Without this heuristic, the computations were barely doable. By exploiting the Master Theorem, Yang (2020b) was the first to mathematically validate this heuristic.

On advantages of non-Gaussianity. In addition to widening the scope of applicability of the Master Theorem to non-Gaussain weight initializations more commonly used in practice (He et al., 2015), our non-Gaussian Master Theorem also underlines the fact that the distribution of *vector-like variables*, i.e. those with only one dimension growing to infinity, such as bias vectors, or input and output layer weights, may affect the limit: e.g. their distribution may affect the limiting NTK. For all cases concerning random matrices, it relaxes the requirement for the matrices to have Gaussian entries.

1.4.7 Limitations and directions for future work

Smoothness requirement. The most crucial limitation of our theorem in the present form is the class of nonlinearities allowed. The most commonly-used activation function in neural networks is ReLU (Fukushima, 1969), $\phi(x) = \max(0, x)$, which is not smooth. Moreover, its derivative which is used to express the backward pass, is not even continuous, thus ruling out the applicability of even the Gaussian theorem, which does not require nonlinearities to lie in $C^\infty(\mathbb{R})$.

One of the plausible ways to generalize our Theorem 1.4.2 to pseudo-Lipschitz nonlinearities would be to consider a continuous family of polynomially smooth functions approximating the given nonlinearity ϕ: $(\phi^\delta)_{\delta>0}$ such that $\|\phi - \phi^\delta\|_\infty \leq \delta \ \forall \delta > 0$. Then one might try to prove the following bound:

$$\exists \delta_0 > 0: \quad \forall p \in [1, \infty) \ \lim_{n\to\infty} \sup_{\delta\in(0,\delta_0)} \mathbb{E}\left[\left|c^\delta(1) - c^\delta(0)\right|^p\right] = 0. \tag{1.40}$$

That is, the conjecture is that $\mathbb{E}\left[\left|c^\delta(1) - c^\delta(0)\right|^p\right]$ goes to zero as $n \to \infty$ uniformly over δ (note that pointwise convergence is what we already have: Eq. (1.38)). If one moreover proves that the same quantity has a finite limit as $\delta \to 0$ for any n (which seems likely as the family $(\phi^\delta)_{\delta>0}$ uniformly approximates ϕ), the n- and δ-limits are interchangeable by the Moore-Osgood theorem:

$$\lim_{n\to\infty} \mathbb{E}\left[|c(1) - c(0)|^p\right] = \lim_{n\to\infty}\lim_{\delta\to 0} \mathbb{E}\left[\left|c^\delta(1) - c^\delta(0)\right|^p\right] = \lim_{\delta\to 0}\lim_{n\to\infty} \mathbb{E}\left[\left|c^\delta(1) - c^\delta(0)\right|^p\right] = 0, \tag{1.41}$$

where one could validate interchanging the expectation and the δ-limit using dominated convergence, and the last limit is due to our "smooth" theorem.

Another plausible way to generalize our theorem to non-smooth nonlinearities could be to prove the following bound:

$$\left|c(t) - c^\delta(t)\right| \leq \delta P\left(\|A^1(t)\|_{op}, \ldots, \|A^L(t)\|_{op}\right) \quad \forall t \in [0, 1], \tag{1.42}$$

where P is a polynomial. This bound does hold for Lipschitz ϕ as demonstrated in our paper, see Chapter 3. The above bound has a finite almost sure limit thanks to concentration of operator norms; this limit is proportional to δ. We proceed with a triangular inequality:

$$|c(1) - c(0)| \leq |c(1) - c^\delta(1)| + |c^\delta(1) - c^\delta(0)| + |c^\delta(0) - c(0)|. \tag{1.43}$$

Then the almost sure $n \to \infty$ limit exists and bounded by something proportional to δ. Taking the infinum over δ gives zero on the right-hand side.

While the above strategy works for Lipschitz nonlinearities, we were not able to prove Eq. (1.42) for pseudo-Lipschitz ones.

Even if the conjecture of Eq. (1.40) holds, this does not allow us to generalize the theorem to non-continuous nonlinearities, like the derivative of ReLU. What gives us hope for this to be still possible is that there exists a variant of the Gaussian Master Theorem that works only for Tensor Programs with no matrix transpositions (thus not being able to express backward passes in neural nets), but allows for any nonlinearities ϕ for which $\mathbb{E}_{z\sim\mathcal{N}(0,1)}|\phi(z)| < \infty$, see Yang (2019b).

All moments of the matrix entries should exist. According to our preliminary experiments, this requirement could be relaxed to existence of only four moments. However, this will automatically take us back to only almost sure convergence, as the L^p one cannot generally hold in this case.

Weight matrices should be initialized in an iid way. While iid Gaussian or uniform initializations are arguably the most widely used ones, orthogonal initializations are aslo not uncommon in practice (see e.g. Hu et al. (2020)). The rationale behind using them is that they preserve signals during the first forward and backward passes, therefore achieving better trainability for very deep nets (Pennington et al., 2017). As matrix entries become dependent, we do not see any direct workaround that would allow applying our results to this setting. However, there are universality results for rotationally invariant matrices in the context of *Approximate Message Passing* (Dudeja et al., 2023; Wang et al., 2024), which is very related to Tensor Programs (but less general). We foresee the following roadmap for proving a Master Theorem in this setting: (1) prove it for uniform Haar measures on orthogonal matrices, (2) apply a perturbation argument, as we did for our Theorem 1.4.2, to generalize the setup to general rotationally invariant matrices.

Convergence speed. Another limitation of the present Master Theorems is that while providing the $n \to \infty$ limit, they do not tell us anything about the convergence speed. One could derive the convergence speed from a tail bound:

$$\mathcal{P}(c^l - \mathring{c}^l > \delta) \leq B(n, l, \delta), \tag{1.44}$$

where the bound B vanishes as $n \to \infty$.

One of the questions one might ask is how one should scale n with l for $B(n, l, \delta)$ to vanish as $l \to \infty$. This questions is particularly interesting since the depth of a Tensor Program expressing a neural network training process is roughly twice the number of gradient descent steps times the number of hidden layers. Since the number of gradient descent steps is usually $10^3 \sim 10^4$, while the depth is $10 \sim 10^2$, this number is large. How large should the width n be for c^l with that large l to be close to its limit $\mathring{c}^l$ predicted by the Master Theorem?

Hanin and Nica (2020b) proves a tail bound for the very first forward pass of a fully-connected neural network with ReLU nonlinearities. Their bound stays bounded when $l \to \infty$ whenever $n \propto l$. We discuss our contributions to tail bounds for Tensor Programs in the next section.

1.5 Tail bounds for Tensor Programs

Our contribution: tail bounds for TPs. The present section is devoted to a work in progress in collaboration with Greg Yang, on tail bounds for Tensor Programs. The main results are Theorems 1.5.1 and 1.5.3. We introduce and discuss these theorems in the present section, and reproduce them with their full proofs in Chapter 4.

Setup. We consider a reduced setup with the following simplifications:

1. Each MatMul operation uses a matrix which has not been used before;
2. Nonlin and Moment use nonlinearities that do not depend on scalar variables.

A TP under these restrictions is able to express a forward pass of a fully-connected network (probably, with skip connections), but not a backward pass (as it requires reusing the same matrices as for the forward pass), and not for convolutional or recurrent networks (as they exploit weight sharing). Without loss of generality a TP under the above restrictions could be expressed as the following iteration:

$$g^{l+1} = A^l \phi(g^1, \dots, g^l), \qquad c^l = \frac{1}{n} \sum_\alpha \psi(g^1_\alpha, \dots, g^l_\alpha) \qquad \forall l \in [L] \tag{1.45}$$

for some integer L.

Assume $\exists \lambda > 0$ such that ϕ is λ-Lipschitz and ψ is λ^2-order-2-pseudo-Lipschitz with a constant ν^l, i.e.

$$|\psi(z^1, \dots, z^l) - \psi(\tilde{z}^1, \dots, \tilde{z}^l)| \le \lambda^2 \left(\nu^l + \sum_{j=1}^{l} \left(|z^j| + |\tilde{z}^j| \right) \right) \sum_{k=1}^{l} \left| z^k - \tilde{z}^k \right|. \tag{1.46}$$

Assume also $\phi(0) = 0$, so that ϕ^2 is λ^2-order-2-pseudo-Lipschitz with a constant 0.

Define

$$v^l = \frac{1}{n} \sum_\beta \phi^2(g^1_\beta, \dots, g^l_\beta), \qquad d^l(\nu) = \frac{1}{n} \sum_\beta \left(\nu + 2 \sum_{j=1}^{l} |g^j_\beta| \right)^2, \tag{1.47}$$

and let $\mathring{v}^l$ and $\mathring{d}^l(\nu)$ be their respective a.s. limits. We were able to prove the following tail bound:

Theorem 1.5.1. *Let $\mathring{c}^l$ be an a.s. limit of c^l. Then $\forall l \in [L],\ n \in \mathbb{N},\ \tau^l > 0$*

$$\mathcal{P}(c^l - \mathring{c}^l > \tau^l) \le B^l \left(e^{-n \left(\frac{\tau^l}{C_2^l(\tau^l, \nu^l)} \right)^2} + n e^{-C_0 n} \right), \tag{1.48}$$

where $B^l = \frac{3^l - 1}{2}$, $C_0 = 8e^{-\frac{2}{\pi}} \approx 4.23$, while C_2^l is defined recursively:

$$\hat{C}_2(v, d) = \lambda^2 \sqrt{2 \left(\left(3 + 256 e^{\frac{2}{\pi}} \right) v^2 + \left(2 + 32\pi e^{\frac{2}{\pi}} \right) \sqrt{\frac{(2v)^3 d}{\pi}} + \left(1 + 8\pi e^{\frac{2}{\pi}} \right) v d \right)}; \tag{1.49}$$

$$C_2^1(\tau, \nu) = \hat{C}_2(1, \nu^2), \qquad \bar{C}_2^l(\tau, \nu) = \hat{C}_2 \left(\bar{v}^l(\tau), \bar{d}^l(\tau, \nu) \right); \tag{1.50}$$

$$C_2^{l+1}(\tau,\nu) = \bar{C}_2^l\left(\hat{\tau}^l(\tau,\nu),\nu\right) + \lambda^2 D^l(\nu) C_2^l\left(\hat{\tau}^l(\tau,\nu),\nu\right) + C_2^l\left(\tau,\bar{\nu}^l(\tau,\nu)\right), \tag{1.51}$$

where $D^l(\nu) = 1 + \sqrt{\frac{\mathring{d}^l(\nu)}{2\pi\mathring{v}^l}}$, $\hat{\tau}^l(\tau,\nu) = \frac{\tau}{\lambda^2 D^l(\nu)}$, $\bar{v}^l(\tau) = \mathring{v}^l + \tau$, $\bar{d}^l(\tau,\nu) = \mathring{d}^l(\nu) + \frac{4\tau}{\lambda^2}$, $\bar{\nu}^l(\tau,\nu) = \nu + \sqrt{\frac{8}{\pi}\bar{v}^l(\hat{\tau}^l(\tau,\nu))}$, *and we have the same bound for* $\mathcal{P}(c^l - \mathring{c}^l < -\tau)$.

The constants of the above bound are given as a recurrence which is quite difficult to resolve analytically. Fortunately, this recurrence simplifies in the limit of infinite λ. Indeed, intuitively, each g^l scales as λ^{l-1}, therefore only last arguments of each ϕ survive for large λ. We study this limit in the upcoming section.

1.5.1 Preserving the tail mass

As already discussed before, we would like to know how large n should be in order for c^l to be close to its limit $\mathring{c}^l$. Given the tail bound as above, we therefore interested in how fast n should grow to infinity when l does. In order to figure it out, we will be looking for the critical scaling, i.e. that one for which the tail bound stays finite as $l \to \infty$.

Note that the second term of the bound vanishes as $l \to \infty$ as long as $n(l) \geq \left(\frac{\ln 3}{C_0} + \epsilon\right) l$ for some fixed $\epsilon > 0$, so a linear scaling of n suffices. We will be concerned about the first term in the sequel.

Large-λ asymptotics. Dealing with an arbitrary Lipschitz constant λ turns out to be tricky. For this reason, we consider the limit case of infinite λ, as intuitively, since $g^l = \Theta_{\lambda\to\infty}(\lambda^{l-1})$, the last argument of ϕ always dominates, hence the program simplifies to

$$g^{l+1} = A^l\phi(g^l), \qquad c^l = \frac{1}{n}\sum_\alpha \psi(g_\alpha^l) \qquad \forall l \in [L]. \tag{1.52}$$

Proposition 1.5.2. *Assume* $C_2^l(\tau^{l+1}, \bar{\nu}(\tau^{l+1},\nu)) = o_{\lambda\to\infty}(\lambda^{2l+2})$. *Then under the above setting,* $C_2^l(\tau^l, 0) \sim q^l\lambda^{2l}$ *as* $\lambda \to \infty$, *where*

$$q^l = \sqrt{2\left(3 + 256e^{\frac{2}{\pi}}\right)}\left((l-1)t + \frac{1-\rho^{2l}}{1-\rho^2}\right). \tag{1.53}$$

Proof. We have $g^l = \Theta(\lambda^{l-1})$. Suppose $\nu^l = 0\ \forall l \geq 1$. Then $c^l = \Theta(\lambda^{2l})$ and $v^l = \Theta(\lambda^{2l})$, while $d^l(\nu) = \Theta\left(\left(\nu + 2\lambda^{l-1}\right)^2\right)$. Since c^l scales with λ, it makes sense to scale τ^l the same way: $\tau^l \sim tc^l \sim t\lambda^{2l}$.

We have $D^l(0) = 1$, hence $\hat{\tau}^l(\tau,0) = \tau/\lambda^2$, and in particular, $\hat{\tau}^l(\tau^{l+1},0) = \tau^l$. Suppose $v^l \sim (\rho\lambda)^{2l}$ and $d^l(0) \sim 4(\sigma\lambda)^{2l-2}$ for some $\rho, \sigma > 0$. Then $\bar{v}^l(\tau^l) \sim (\rho^{2l} + t)\lambda^{2l}$, $\bar{d}^l(\tau^l, 0) \sim 4(\sigma^{2l-2} + t)\lambda^{2l-2}$, and

$$\bar{C}_2^l(\tau^l,0) = \hat{C}_2(\bar{v}^l(\tau^l), \bar{d}^l(\tau^l,0)) \sim \lambda^{2l+2}(\rho^{2l} + t)\sqrt{2\left(3 + 256e^{\frac{2}{\pi}}\right)}. \tag{1.54}$$

We will look for $C_2^l(\tau^l,0) \sim q^l\lambda^{2l}$. If we assume $C_2^l(\tau^{l+1}, \bar{\nu}(\tau^{l+1},\nu)) = o(\lambda^{2l+2})$ then

$$q^{l+1} = (\rho^{2l} + t)\sqrt{2\left(3 + 256e^{\frac{2}{\pi}}\right)} + q^l. \tag{1.55}$$

Noting that $q^1 = \sqrt{2\left(3 + 256e^{\frac{2}{\pi}}\right)}$, we get q^l as stated in the proposition. □

Therefore if we would like to bound the tail as $\mathcal{P}(c^l - \mathring{c}^l > \tau) \leq \delta$ for some fixed $\delta \in (0,1)$, we need to have

$$n \geq 2\left(3 + 256e^{\frac{2}{\pi}}\right)\left(\frac{\frac{1-\rho^{2l}}{1-\rho^2} + (l-1)t}{t}\right)^2 \ln\left(\frac{3^l - 1}{2\delta}\right) \sim 2\left(3 + 256e^{\frac{2}{\pi}}\right)(\ln 3) l^3, \tag{1.56}$$

where the asymptotics is for large l. That is, our bound suggests that the width has to scale faster than cubically with depth of the program in order for the scalars to converge to their limits. In a similar setup, for fully-connected ReLU networks with no shared weights, Hanin and Nica (2020b) demonstrated the proportional scaling, i.e. $n \sim l$, to be relevant under this setting.

This indicates that the bound of our Theorem 1.5.1 is loose. Moreover, the assumption $C_2^l(\tau^{l+1}, \bar{\nu}(\tau^{l+1}, \nu)) = o(\lambda^{2l+2})$ we made in the above proposition turns out to be wrong: one could demonstrate that $\ln C_2^l(\tau^{l+1}, \bar{\nu}(\tau^{l+1}, \nu))$ grows quadratically in l. This fact also adds to the point that the present bound could be improved considerably.

1.5.2 TP with dependence on last vectors only

Assuming an even simpler class of TPs whose nonlinearities are allowed to depend only on the last generated vectors so far, allows for a cleaner and easier-to-prove version of Theorem 1.5.1. That is, consider the following iteration:

$$g^{l+1} = A^l \phi(g^l), \qquad c^l = \frac{1}{n}\sum_\alpha \chi^2(g_\alpha^l) \qquad \forall l \in [L]. \tag{1.57}$$

The above covers forward passes of fully-connected networks with no bias vectors (since bias vectors require multivariate nonlinearities, see our example in Section 1.4). In addition, assume ϕ and χ to be λ-Lipschitz with $\phi(0) = \chi(0) = 0$. We prove the following analogue of Theorem 1.5.1:

Theorem 1.5.3. *Let $\mathring{c}^l$ be an a.s. limit of c^l. Then $\forall l \in [L],\ n \in \mathbb{N},\ \tau^l > 0$*

$$\mathcal{P}(c^l - \mathring{c}^l > \tau^l) \leq B^l\left(e^{-n\left(\frac{\tau^l}{C_2^l(\tau^l)}\right)^2} + ne^{-C_0 n}\right), \tag{1.58}$$

where $B^l = l$, $C_0 = 8e^{-\frac{1}{2}} \approx 4.85$, while C_2^l is defined recursively:

$$C_2^1(\tau^1) = \lambda^2\sqrt{2\left(3 + 64e^{1/2}\right)}, \qquad \bar{C}_2^l(\tau^l) = \lambda^2(\mathring{v}^l + \tau^l)\sqrt{2\left(3 + 64e^{1/2}\right)}, \tag{1.59}$$

$$C_2^{l+1}(\tau^{l+1}) = \lambda^2 C_2^l(\tau^{l+1}/\lambda^2) + \bar{C}_2^l(\tau^{l+1}/\lambda^2), \tag{1.60}$$

and we have the same bound for $\mathcal{P}(c^l - \mathring{c}^l < -\tau)$.

From the positive side, the above theorem does not exhibit the same issues for large λ as Theorem 1.5.1 does. That is, we were able to compute the $\lambda \to \infty$ asymptotics of the bound of Theorem 1.5.3 and study its behavior for large l.

Large-l asymptotics for large λ. We prove an analogue of Proposition 1.5.2 for Theorem 1.5.3:

Proposition 1.5.4. *$C_2^l(t) = \sqrt{2\left(3+64e^{1/2}\right)}q^l(t)\lambda^{2l}$, where*

$$q^l(t) = \frac{1-\rho^{2l}}{1-\rho^2} + (l-1)t. \tag{1.61}$$

Proof. We have $g^l = \Theta(\lambda^{l-1})$, $\mathring{v}^l = \Theta(\lambda^{2l})$, and $\mathring{c}^l = \Theta(\lambda^{2l})$, therefore it makes sense to take $\tau^l = t\lambda^{2l}$ for some $t > 0$. In this case, $\tau^{l+1}/\lambda^2 = \tau^l \ \forall l$. Suppose also $\mathring{v}^l = (\rho\lambda)^{2l}$ for some $\rho > 0$. This allows us, with a slight abuse of notation, to consider C_2^l and Q^l as functions of t:

$$C_2^1(t) = \lambda^2\sqrt{2\left(3+64e^{1/2}\right)}, \quad \bar{C}_2^l(t) = \lambda^{2l+2}(\rho^{2l}+t)\sqrt{2\left(3+64e^{1/2}\right)}, \quad C_2^{l+1}(t) = \lambda^2 C_2^l(t) + \bar{C}_2^l(t). \tag{1.62}$$

We will look for $C_2^l(t) = \sqrt{2\left(3+64e^{1/2}\right)}q^l(t)\lambda^{2l}$:

$$q^1(t) = 1, \qquad q^{l+1}(t) = q^l(t) + \rho^{2l} + t. \tag{1.63}$$

The solution is $q^l(t) = \frac{1-\rho^{2l}}{1-\rho^2} + (l-1)t$. □

Therefore if we would like to bound the tail as $\mathcal{P}(c^l - \mathring{c}^l > \tau) \le \delta$ for some fixed $\delta \in (0,1)$, we need to have

$$n \ge 2\left(3+64e^{1/2}\right)\left(\frac{\frac{1-\rho^{2l}}{1-\rho^2}+(l-1)t}{t}\right)^2 \ln\left(\frac{l}{\delta}\right) \sim 2\left(3+64e^{1/2}\right)l^2\ln l \tag{1.64}$$

as $l \to \infty$. This is better than $n \sim l^3$ predicted by Theorem 1.5.1, but still worse than $n \sim l$ predicted by Hanin and Nica (2020b) in the same setup for ϕ being ReLU. We analyze the reason for this suboptimality in the sequel.

1.5.3 Comparison to Hanin and Nica (2020b)

The goal of the present section is to demonstrate what allowed Hanin and Nica (2020b) to obtain a tail bound which stays bounded when $n \sim l \to \infty$. Consider the following iteration:

$$g^{l+1} = A^l\phi(g^l), \qquad v^l = \frac{1}{n}\sum_\alpha \phi^2(g_\alpha^l) \qquad \forall l \in [L], \tag{1.65}$$

where ϕ is positively homogeneous (e.g. ReLU). Then using the fact that $g^1 \sim \mathcal{N}(0, I_n)$, while A_{ij}^l are sampled iid from $\mathcal{N}(0, n^{-1})$ $\forall i,j \in [n]$ $\forall l \in [L]$, we arrive at

$$v^L \sim \prod_{l=1}^L \left(\frac{1}{n}\sum_{\alpha=1}^n \phi^2(\xi_\alpha^l)\right), \tag{1.66}$$

where all ξ^l are sampled iid from $\mathcal{N}(0, I_n)$.

Suppose ϕ is normalized as follows: $\mathbb{E}_{z\sim\mathcal{N}(0,1)}[\phi^2(z)] = 1$. Then taking a logarithm, we get

$$\ln v^L \sim \sum_{l=1}^{L} \ln\left(1+\zeta^l\right), \qquad \zeta^l = \frac{1}{n}\sum_{\alpha=1}^{n} \phi^2(\xi_\alpha^l) - 1 \quad \forall l \in [L]. \tag{1.67}$$

Note that $\mathbb{E}[\zeta^l] = 0$, while $\mathbb{V}\mathrm{ar}[\zeta^l] \propto n^{-1}$ $\forall l \in [L]$. Since all ζ^l become small for large n, we can Taylor-expand the logarithm to get

$$\ln v^L \sim \sum_{l=1}^{L} \ln\left(1+\zeta^l\right) = \sum_{l=1}^{L} \left(\zeta^l - \frac{(\zeta^l)^2}{2} + O_{n\to\infty}\left((\zeta^l)^3\right)\right). \tag{1.68}$$

Since ζ^l's are independent random variables with zero mean, we have

$$\mathbb{E}\left[\sum_{l=1}^{L}\zeta^l\right] = 0, \qquad \mathbb{V}\mathrm{ar}\left[\sum_{l=1}^{L}\zeta^l\right] \propto \frac{L}{n}, \qquad \mathbb{E}\left[\sum_{l=1}^{L}(\zeta^l)^2\right] \propto \frac{L}{n}. \tag{1.69}$$

This indicates that the distribution of $\ln v^L$ (hence that of v^L) neither diverges, nor concentrates, when $n \propto L \to \infty$.

The reason that lead us to this conclusion is that we expressed $\ln v^l$ as a sum of independent random variables $\ln\left(1+\zeta^l\right)$, for which the leading term, ζ^l, had zero mean and variance $\Theta(n^{-1})$. The sum of this leading term then scaled as $\sqrt{L/n}$, not as $L/\sqrt{n}$, as the sum would normally do, if there were no independence and zero mean. If we did not exploit the independence of ζ^l, the result would suggest that $n \sim L^2$ is necessary for the distribution to not diverge.

Comparison to Theorem 1.5.3. As mentioned earlier, our Theorem 1.5.3 suggests $n \sim L^2 \ln L$, which is not too far from $n \sim L^2$. The idea behind the proof is to induct over l, and exploit a concentration bound at each step. By this manner, there is no clear way to exploit the fact that the variable v^L we are analyzing is a product of independent random variables with zero mean. As we do not do that, we naturally get nothing better than $n \sim L^2$.

Obstacles to extending the above technique to TPs. The technique demonstrated above does not allow for a straightforward extension to multivariate nonlinearities (as in our Theorem 1.5.1), and even to univariate but non-homogeneous ones (as in our Theorem 1.5.3). The reason is that $\ln v^l$ becomes no longer a sum of independent random variables in these cases. A proper extension of the above technique to TPs is one of the areas of our active research at the present moment.

1.6 Beyond Tensor Programs

As we aimed demonstrating above, Tensor Programs provide a very general and powerful framework for neural network computations. However, they have the following limitations.

Poor tractability. While the Master Theorem provides limits for scalars generated during training by a very wide class of architectures, there are solutions that are easier to interpret and analyze for some specific setups. One of the examples is linear networks already mentioned in Section 1.2. The training dynamics could be integrated analytically for some special class of initializations (Saxe

et al., 2013), see also Chizat et al. (2024). While the use of linear nets is limited, one could view nonlinear nets as perturbations of linear ones; this idea first appeared in Li et al. (2022). As one of the contributions of the present thesis, we propose a generalization bound based on this idea, see Section 1.8.

Under-generality. There are practical cases not covered by Tensor Programs as presented here. One of the notable examples is training with Adam optimizer (Kingma and Ba, 2015) instead of gradient descent. The issue comes from the fact that Adam uses normalized gradients to update the weights, and the gradients are normalized entrywise. In order to compute it, one needs elementwise division for matrices, which are not expressible in the framework.

However, a recent work of Yang and Littwin (2023) extends the formalism of Tensor Programs in such a way that training with adaptive optimizers like Adam gets also covered. They prove a corresponding Master Theorem for their extension, both in the Gaussian and non-Gaussian cases; the proof of the latter case is based on one of the works included in the present thesis (Golikov and Yang, 2022).

Other notable examples are exact minimizers. Instead of following the gradient descent which aims to minimize the loss function, one could skip the optimization procedure and consider the loss minimizers directly. In order to naively compute them, one needs an infinitely deep Tensor Program, as one needs infinitely many gradient steps. Moreover, gradient descent is not generally guaranteed to find a global minimum. At the same time, global minima of L_2-regularized loss are interesting objects on their own, and we study them in the present thesis; see Section 1.7.

1.7 Feature learning in L_2-regularized DNNs: Attraction/repulsion and sparsity

Our contribution: properties of minima of L_2-regularized loss. As mentioned in Section 1.6, Tensor Programs do cover gradient descent training of a large class of architectures, but do not cover some other interesting problem formulations related to deep learning. One of such problem formulations is directly studying the minima of L_2-regularized loss instead of considering a training process, which we do in our paper Jacot et al. (2022) reproduced in Chapter 5.

Setup. Consider a fully-connected network with L layers and an activation function σ. Its output $Z_L \in \mathbb{R}^{n_L \times N}$ on a dataset $X \in \mathbb{R}^{n_0 \times N}$ of size N is given by the following iteration:

$$Z_l^\sigma = \sigma(Z_l), \quad Z_l = W_l Z_{l-1}^\sigma \quad \forall l \in [L], \qquad Z_0^\sigma = X, \tag{1.70}$$

where σ is applied elementwise, and $W^l \in \mathbb{R}^{n_l \times n_{l-1}}$ for $l \in [L]$ are weight matrices. We will be studying the following L_2-regularized loss:

$$\mathcal{L}(W_{1:L}) = C(Z_L(X; W_{1:L})) + \lambda \sum_{l=1}^{L} \|W_l\|_F^2, \tag{1.71}$$

where C is an arbitrary cost function. We are going to consider two reformulations of the above loss function: one in terms of hidden representations and one in terms of covariances. Each reformulation gives its own insights, as we discuss below.

1.7.1 Reformulation in terms of hidden representations

Since $Z_l = W_l Z_{l-1}^\sigma$, any weight matrix W_l can be represented as $W_l = Z_l (Z_{l-1}^\sigma)^+ + \tilde{W}_l$, and Z_l does not depend on $\tilde{W}_l$. Since we penalize the Frobenius norms of weight matrices, it always makes sense to put $\tilde{W}_l$ to be zero. This remark suggests rewriting the above loss function in terms of hidden representation $(Z_l)_{l=1}^L$:

$$\mathcal{L}^r(Z_{1:L}) = C(Z_L) + \lambda \sum_{l=1}^{L} \left\| Z_l (Z_{l-1}^\sigma)^+ \right\|_F^2 . \tag{1.72}$$

Correspondence between global minima. Clearly, the two loss functions have global minima of the same value:

$$\inf_{W_{1:L}} \mathcal{L}(W_{1:L}) = \inf_{Z_{1:L} \in \mathcal{Z}} \mathcal{L}^r(Z_{1:L}), \tag{1.73}$$

where $\mathcal{Z}$ is the set of matrices $Z_{1:L}$ realizable as pre-activation representations in our neural network. To be precise, matrices from $\mathcal{Z}$ are those who satisfy $\operatorname{Im} Z_l^T \subseteq \operatorname{Im} Z_{l-1}^{\sigma,T}$ for any $l \in [L]$; in this and only in this case, there exists W_l such that $Z_l = W_l Z_{l-1}^\sigma$.

Correspondence between local minima. If $W_{1:L}$ is a local minimizer of $\mathcal{L}$ then $Z_{1:L}(W_1, \ldots, W_L)$ is a local minimizer of $\mathcal{L}^r$. Indeed, if we slightly perturb $Z_{1:L}$ inside $\mathcal{Z}$, this perturbation always translates into a sufficiently small perturbation of $W_{1:L}$ by definition of $\mathcal{Z}$. Conversely, if $Z_{1:L}$ is a local minimizer of $\mathcal{L}^r$ then $\{Z_l (Z_{l-1}^\sigma)^+\}_{l=1}^L$ is a local minimizer of $\mathcal{L}$. Indeed, any sufficiently small perturbation of weights results only in a small perturbation of hidden representations. Since adding $\tilde{W}_{1:L}$ to weights breaks local minima, the local minimizers of $\mathcal{L}$ and $\mathcal{L}^r$ are in a one-to-one correspondence.

Locality and attraction-repulsion mechanism. There is a few more interesting observations. First, minimizing $\mathcal{L}^r$ with gradient descent is *local* in a sense that $\nabla_{Z_l} \mathcal{L}^r$ depends only on Z_{l+1}, Z_l, and Z_{l-1}. Note that minimizing $\mathcal{L}$ with gradient descent is not local since $\nabla_{W_l} \mathcal{L}$ depends on all weight matrices. Local learning rules are considered more "biologically plausible" since non-local rules cannot be implemented by brain-like systems (Bengio et al., 2015; Illing et al., 2019).

Second, for each Z_l, the objective contains an "attractive term" $\left\| Z_l (Z_{l-1}^\sigma)^+ \right\|_F^2$, which drives it to zero, and a "repulsive term" $\| Z_{l+1} (Z_l^\sigma)^+ \|_F^2$, which drives it into infinity. This attraction-repulsion effect has a nice parallel with the *information bottleneck theory* (Tishby et al., 2000; Tishby and Zaslavsky, 2015; Shwartz-Ziv and Tishby, 2022; Saxe et al., 2019; Kawaguchi et al., 2023): we aim to erase as much information from the input (attraction to zero), while keeping as much information about the output (repulsion from zero).

1.7.2 Reformulation in terms of covariance matrices

Notice that the loss functional $\mathcal{L}^r$ depends on $Z_{1:L-1}$ only through covariance matrices $K_l = Z_l^T Z_l \in \mathbb{R}^{N \times N}$ and $K_l^\sigma = Z_l^{\sigma,T} Z_l^\sigma \in \mathbb{R}^{N \times N}$ for $l \in [L-1]$, and $K_0^\sigma = X^T X$. This motivates us considering another reformulation of the loss functional:

$$\mathcal{L}^k(K_{1:L-1}, K_{1:L-1}^\sigma, Z_L) = C(Z_L) + \lambda \sum_{l=1}^{L} \operatorname{tr}\left(K_l (K_{l-1}^\sigma)^+ \right). \tag{1.74}$$

Note that $\mathcal{L}^r(Z_{1:L}) = \mathcal{L}^k(Z_1^T Z_1, \ldots, Z_{L-1}^T Z_{L-1}, Z_1^{\sigma,T} Z_1^\sigma, \ldots, Z_{L-1}^{\sigma,T} Z_{L-1}^\sigma, Z_L)$ since

$$\left\| Z_l (Z_{l-1}^\sigma)^+ \right\|_F^2 = \operatorname{tr}\left(Z_l^T Z_l (Z_{l-1}^{\sigma,T} Z_{l-1}^\sigma)^+ \right) = \operatorname{tr}\left(K_l (K_{l-1}^\sigma)^+ \right). \tag{1.75}$$

Correspondence between global minima. Clearly,

$$\inf_{Z_{1:L} \in \mathcal{Z}} \mathcal{L}^r(Z_{1:L}) = \inf_{(K_{1:L-1}, K_{1:L-1}^\sigma, Z_L) \in \mathcal{K}_{\mathbf{n}}} \mathcal{L}^k(K_{1:L-1}, K_{1:L-1}^\sigma, Z_L) \tag{1.76}$$

for $\mathcal{K}_{\mathbf{n}}$ being a set of $K_{1:L-1}$, $K_{1:L-1}^\sigma$, and Z_L representable as $K_l = Z_l^T Z_l$, $K_l^\sigma = Z_l^{\sigma,T} Z_l^\sigma$, where $Z_{1:L} \in \mathcal{Z}$; we put a subscript $\mathbf{n} := n_{1:L-1}$ in order to explicitly mark the dependence on width.

Defining $\mathcal{K}_{\mathbf{n}}$ without relying on $\mathcal{Z}$. We could equivalently define the set of feasible correlation matrices $\mathcal{K}_{\mathbf{n}}$ without relying on the set of feasible hidden representations $\mathcal{Z}$ as follows. Define $\mathcal{K}_{\mathbf{n}}$ as a set of $K_{1:L-1}$, $K_{1:L-1}^\sigma$, and Z_L such that for any $l \in [L-1]$,

1. (K_l, K_l^σ) lies in S_{n_l} defined as

$$S_{n_l} = H_{n_l}^1 \left((xx^T, \sigma(x)\sigma(x)^T) : x \in \mathbb{R}^N \right), \tag{1.77}$$

   where $H_k^1(\Omega) := \left\{ \sum_{j=1}^k x_j : x_j \in \Omega \; \forall j \in [k] \right\}$ for Ω being a subset of a vector space,

2. $\operatorname{Im} K_l \subset \operatorname{Im} K_{l-1}^\sigma$,

3. $\operatorname{Im} Z_L^T \subset \operatorname{Im} K_{L-1}^\sigma$.

$\mathcal{K}_{\mathbf{n}}$ saturates when width exceeds critical. When σ is positively homogeneous (i.e. $\sigma(\alpha x) = \alpha\sigma(x)$ for any $\alpha > 0$ and any x), S_{n_l} coincides with a conical hull:

$$S_{n_l} = H_{n_l}^+ \left((xx^T, \sigma(x)\sigma(x)^T) : x \in \mathbb{R}^N \right), \tag{1.78}$$

where a k-conical hull is defined as $H_k^+(\Omega) := \left\{ \sum_{j=1}^k \alpha_j x_j : x_j \in \Omega, \, \alpha_j \geq 0 \, \forall j \in [k] \right\}$.

Define $H^+(\Omega) = \cup_{k=1}^\infty H_k^+(\Omega)$. By Caratheodory's theorem on conical hulls, if $\Omega \subset \mathbb{R}^d$ then $H_k^+(\Omega) = H^+(\Omega)$ for any $k \geq d$. Since the dimension of symmetric $N \times N$ matrices is $N(N+1)/2$ and we have a pair of them in our Ω, this means that whenever $n_l \geq N(N+1)$ for each $l \in [L-1]$, $\mathcal{K}_{\mathbf{n}} = \mathcal{K}$ defined as a set of $K_{1:L-1}$, $K_{1:L-1}^\sigma$, and Z_L such that for any $l \in [L-1]$,

1. (K_l, K_l^σ) lies in S defined as

$$S = H^+ \left((xx^T, \sigma(x)\sigma(x)^T) : x \in \mathbb{R}^N \right), \tag{1.79}$$

2. $\operatorname{Im} K_l \subset \operatorname{Im} K_{l-1}^\sigma$,

and $\operatorname{Im} Z_L^T \subset \operatorname{Im} K_{L-1}^\sigma$.

The set $\mathcal{K}$ is width-independent, meaning that as long as $n_l \geq N(N+1)$ for each $l \in [L-1]$, the problem $\mathcal{L}^k(K_{1:L-1}, K_{1:L-1}^\sigma, Z_L) \to \min_{\mathcal{K}_{\mathbf{n}}}$ does not depend on $\mathbf{n}$.

The minimal loss plateaus when exceeding the critical width. Clearly, $\inf_{W_{1:L}} \mathcal{L}(W_{1:L})$ is non-increasing as a function of each n_l for $l \in [L-1]$. Indeed, if some weight configuration gives infinum plus epsilon loss, we can add neurons with zero inbound and outbound weights, which will affect neither the cost, nor the penalty. Since $\inf_{W_{1:L}} \mathcal{L}(W_{1:L}) = \inf_{\mathcal{K}_\mathbf{n}} \mathcal{L}^k(\ldots)$ and the latter stops changing when all $n_l \geq N(N+1)$, the former behaves the same way. In other words, when the network width surpasses some threshold (which is at most quadratic wrt the number of training points), one cannot improve the representation (in terms of penalized train loss) by adding new neurons.

Let us define the rank of a pair of covariance matrices $(K, K^\sigma) \in S$ as the minimal k such that $(K, K^\sigma) \in S_k$. We have the following sandwich bound: $\operatorname{rk} K \leq \operatorname{rk}(K, K^\sigma) \leq N(N+1)$. The plateau starts when each n_l reaches $\operatorname{rk}(K_l, K_l^\sigma)$ for some $(K_{1:L-1}, K^\sigma_{1:L-1}, Z_L) \in \mathcal{K}$ being a global minimizer of $\mathcal{L}^k$.

Tightness of the plateau asymptotics. The asymptotics of the upper-bound for the start of the plateau is tight up to a constant factor. Indeed, consider a full bipartite graph with $N/2$ vertices in each part; it has $N^2/4$ edges. Consider its incidence matrix $E_N \in \mathbb{R}^{N^2/4 \times N}$. Then the matrix $B_N = E_N^T E_N \in \mathbb{R}^{N \times N}$ indicates the number of common edges for any pair of vertices:

$$B_N = \begin{pmatrix} \frac{N}{2} I_{N/2} & 1_{N/2} \\ 1_{N/2} & \frac{N}{2} I_{N/2} \end{pmatrix}. \tag{1.80}$$

Consider a dataset (X_N, Y_N) where $X_N = I_N$ and $Y_N = B_N$, and a ReLU network with one hidden layer of width n: $Z_N = V^T \sigma(U) X_N$ for $U, V \in \mathbb{R}^{n \times N}$. Let us first minimize the representation cost $\|U\|_F^2 + \|V\|_F^2$ over U, V such that $Z_N = V^T \sigma(U) X_N = B_N$. Introducing a matrix of Lagrange multipliers $H \in \mathbb{R}^{n \times N}$,

$$V + H \odot \sigma(U) = 0, \qquad U + H \odot V \odot \sigma'(U) = 0. \tag{1.81}$$

The first equation implies that for each non-positive entry U, V should have a zero corresponding entry. The second one then implies that that entry of U should also be zero. For all positive entries of U, they should equal the corresponding entries of V. Therefore U should equal V and both should have non-negative entries at any minimum of the representation cost perfectly fitting the data.

Note that the matrix $Y_N = B_N$ cannot be expressed as $B_N = U^T U$ for any $U \in \mathbb{R}^{n \times N}$ if $n < N^2/4$ since the above graph is full bipartite. Therefore a global minimum of the representation cost could fit the data only if $n \geq N^2/4$. Hence any global minimum of the representation cost as a function of (K, K^σ) is given by $K^\sigma = K = U^T U = V^T \sigma(U) = B_N$. At this point, $\operatorname{rk}(K, K^\sigma) = N^2/4$.

If we consider the penalized loss $\mathcal{L}$ with small enough λ, one of its global minima should be not far from global minima of the representation cost. Since correlation matrices are continuous as functions of weights, both K and K^σ could be perturbed only slightly when λ is small, hence $\operatorname{rk}(K, K^\sigma)$ could not decrease. Therefore for small enough λ, the plateau of $\mathcal{L}$ cannot start before $N^2/4$.

1.8 Generalization bound for nearly-linear networks

1.8.1 Generalization problem in supervised learning

Even though Tensor Programs provide the result of learning (in particular, loss and accuracy on train and validation sets) in the limit of infinite width, they do so in terms of nested expectations, which are often not analytically tractable, and even not analytically computable. In other words, while the machinery of Tensor Programs could predict the performance on held-out data, it does not do so in a tractable way.

At the same time, the question of *generalization ability* is arguably the most important question of *supervised learning*, not only of deep learning theory which we discuss in the present thesis. The question is: *"Why would a model learned to minimize a loss on a finite training set, have a low loss on the whole data distribution?"*

Distribution risk. Indeed, in the most general scenario of supervised learning, what we aim for is to minimize a *distribution risk functional*:

$$R[f] = \mathbb{E}_{(x,y)\sim\mathcal{D}} r(y, f(x)) \to \min_{f\in\mathcal{F}}, \tag{1.82}$$

where $\mathcal{F}$ is a model class (e.g. a class of all models realizable by neural networks of a given architecture), $\mathcal{D}$ is the data distribution, and $r(y, \hat{y})$ is a risk function. For binary classification with labels ± 1, a typical choice of the risk function is misclassification risk: $r(y, \hat{y}) = [y\hat{y} < 0]$, where "$[P]$" denotes the indicator of a condition P.

Empirical risk. One cannot approach the problem of Eq. (1.82) directly unless one is able to sample from $\mathcal{D}$ infinitely. However, in the most common scenario, only a finite set of samples $(x_i, y_i)_{i=1}^m$ from $\mathcal{D}$ is available. In this case, the problem we *could* approach is minimization of *empirical risk functional*:

$$\hat{R}[f] = \frac{1}{m}\sum_{i=1}^{m} r(y_i, f(x_i)) \to \min_{f\in\mathcal{F}}. \tag{1.83}$$

The problem of Eq. (1.83) could be still not tractable for several reasons. First, many risk functions (e.g. misclassification risk) are not differentiable, which precludes us from applying gradient-based minimization algorithms. A typical workaround for this is to substitute the risk function r with a *differentiable surrogate* ℓ. Common surrogates for misclassification risk include *logistic loss* $\ell(y, \hat{y}) = \ln\left(1 + e^{-y\hat{y}}\right)$ and *quadratic loss* $\ell(y, \hat{y}) = \frac{1}{2}(y - \hat{y})^2$. However, a minimizer of $\hat{\mathcal{L}}[f] = \frac{1}{m}\sum_{i=1}^m \ell(y_i, f(x_i))$ does not have to be a minimizer of $\hat{R}[f]$. Moreover, gradient-based optimization methods are not guaranteed to find global minima, but only local ones.

Generalization bounds. Nevertheless, if the output $\hat{f}$ of our learning algorithm is already known, we can compute its empirical risk $\hat{R}[\hat{f}]$, even though $\hat{f}$ is not guaranteed to be a minimizer. However, what is more interesting is its distribution risk $R[\hat{f}]$, which we cannot compute since we do not have an access to the data distribution $\mathcal{D}$. Starting from early 70's with a pioneering work of Vapnik and Chervonenkis (Vapnik and Chervonenkis, 1971), researchers were trying to construct tight *generalization bounds*, i.e. bounds for *generalization gap*:

$$R[\hat{f}] - \hat{R}[\hat{f}] \le B(\mathcal{F}, \hat{f}, m, \delta) \quad \text{w.p. } \ge 1 - \delta \text{ over } (x_i, y_i)_{i=1}^m \sim \mathcal{D}^{\otimes m}. \tag{1.84}$$

Such bounds have to be probabilistic since it is possible to sample a "bad" dataset (say, exactly the same point m times), trained on which the model will generalize poorly. If the bound had to work for all possible datasets, it would have to account for these cases, hence to be too pessimistic. However, the probability of sampling such datasets is low, and we can account for it by introducing a "failure probability" δ.

One should expect the bound $B(\mathcal{F}, \hat{f}, m, \delta)$ to decrease with δ (tighter bounds are possible with higher failure probability) and with m (the more training points we have, the better the empirical risk approximates the distribution one).

Our contribution: a non-vacuous a-priori generalization bound for neural nets. In our paper Golikov (2025) reproduced in Chapter 6, we propose a generalization bound (Theorem 1.8.2 below) which becomes *non-vacuous* (i.e. guarantees the distribution risk of the learned model to be better than that of random guess) for neural networks with activation functions close to being linear. To the best of our knowledge, our bound is the first non-vacuous *a-priori* generalization bound available in the literature, meaning that it could be computed before training the neural network. Before moving forward to our bound, we start with a brief review on existing generalization bounds.

1.8.2 Generalization bounds for neural networks

Uniform bounds. The pioneering bound of Vapnik and Chervonenkis (1971) falls into a class of bounds that bound the generalization gap uniformly over the model class $\mathcal{F}$:

$$R[\hat{f}] - \hat{R}[\hat{f}] \leq \sup_{f \in \mathcal{F}} \left(R[f] - \hat{R}[f] \right). \tag{1.85}$$

The resulting bound does not depend on the specific learned model $\hat{f}$, but only on the whole model class $\mathcal{F}$. This could be seen as an advantage as one does not have to learn the model first in order to bound the gap (which could be computationally expensive). However, such bounds are too pessimistic for most practically used neural networks, as the class of models realizable by a given architecture often contains "bad" models that fit the training dataset well, but not the whole data distribution. Zhang et al. (2017) obtained such *overfit* models explicitly by modifying the training procedure.

If one could somehow guarantee that the output of the learning procedure $\hat{f}$ lies in a subset of the whole model class $\mathcal{F}$, one could restrict a uniform bound to this class thus making it stronger. For example, some works (e.g. Bartlett et al. (2017); Dziugaite and Roy (2017); Neyshabur et al. (2018)) assume that gradient descent is *biased* towards solutions of small weight norms. This allows one to prove a uniform bound that depends on the weight norm of the solution (Bartlett et al., 2017): the lower the norm, the smaller the class, the better the bound.

This however transforms an *a-priori* bound (i.e. one that could computed before the actual training procedure) of Eq. (1.85), to an *a-posteriori* bound (computable only as soon as $\hat{f}$ is known). We will discuss both classes later in this section.

PAC-Bayesian bounds. Uniform bounds do not allow to take *preferences* of the learning algorihm into account in a natural way. For example, if we assume that gradient descent is biased towards minimizers of small weight norm, we would like to have a bound which is valid for any weight norm (and not just restrict it as for uniform bounds), but who would also become better (i.e. smaller) when actual weight norms are small.

PAC-Bayesian bounds first introduced in McAllester (1999) provide a natural mechanism for imposing such preferences: one starts with a *prior* distribution P over models in $\mathcal{F}$. This distribution should be chosen independently on the training dataset. The learning algorithm is supposed to output another distribution Q over models in $\mathcal{F}$, called the *posterior*[8]. The resulting bound depends on Kullback-Leibler divergence of the posterior from the prior. That is, if our guess of preferred models was correct, the divergence is lower and the bound is better.

Some examples of PAC-Bayesian bounds for neural networks include Dziugaite and Roy (2017); Neyshabur et al. (2018); Zhou et al. (2019); Biggs and Guedj (2022). All of these bounds are naturally a-posteriori as they require an output of the learning algorithm (a posterior distribution) in order to be evaluated.

A-priori and a-posteriori bounds. Most of the generalization bounds we are aware of are a-posteriori, i.e. they cannot be computed before one gets the actual trained model $\hat{f}$. For example, many PAC-Bayesian bounds (Dziugaite and Roy, 2017; Neyshabur et al., 2018; Biggs and Guedj, 2022), as well as uniform bounds based on Rademacher complexity (Bartlett et al., 2017), depend on the parameter norms. The bound of Zhou et al. (2019), also of PAC-Bayesian nature, depends on the trained model size after compression and quantization (and also applies only to this compressed and quantized model). The *comparative* bound of Galanti et al. (2023) depends on the *effective depth* of the learned model, which also cannot be computed in advance.

An obvious disadvantage of a-posteriori bounds is that whenever the learned model $\hat{f}$ is available, one could reliably estimate its performance using a *held-out* dataset, i.e. a set of samples from $\mathcal{D}$ which have not been used for training. This puts doubt on the practical usefulness of such bounds.

At the same time, an a-priori bound, even a loose one, could be potentially used in *Neural Architecture Search* for model selection. That is, such a bound could rule out poorly-performing architectures without having to explictly evaluate them after the actual training, saving a lot of computational time.

In order to construct an a-priori generalization bound, one should take the *implicit bias* of the learning rule into account in order to gain some information about $\hat{f}$ without having it explicitly. This information should be *sure*, not in a form of a guess, as for PAC-Bayesian bounds.

To the best of our knowledge, the bound we propose in our work Golikov (2025) is the first a-priori bound which could be non-vacuous for neural networks of realistic size: the bound of Vapnik and Chervonenkis (1971), besides being a-priori, becomes vacuous as soon as the number of parameters gets larger than the dataset size[9]. The former spans $10^7 \sim 10^{10}$ which is considerably larger than sizes of available datasets.

Non-vacuous bounds. While researchers have been proposing various generalization bounds since early 70's, generalization bounds who are *non-vacuous* for neural networks (i.e. ones predicting the distribution risk to be better than that of random guess) started to appear only recently. Dziugaite and Roy (2017) constructed a bound which was non-vacuous for small fully-connected networks trained on MNIST and FashionMNIST. This bound, being PAC-Bayesian, relied on injecting noise to learned weights; therefore it was applicable to "noised" models, but not the original ones. The bound of Zhou et al. (2019) turned out to be non-vacuous for a much larger convolutional VGG-type architecture (Simonyan and Zisserman, 2015) trained on ImageNet (Russakovsky et al., 2015), but

[8]Note that these prior and posterior distribution are not related to each other as in Bayesian statistics. Also, these notions of prior and posterior are not related to the notions of a-priori and a-posteriori bounds.

[9]Actually, even earlier due to a multiplicative constant.

relied on compression and quantization; it was also not directly applicably to the original trained model. The first non-vacuous bound applicable to the original model itself was that of Biggs and Guedj (2022). However, their construction works only for fully-connected networks with one hidden layer; it is not clear how to extend it even to multilayer fully-connected nets. All of the bounds mentioned in this paragraph are a-posteriori.

1.8.3 Generalization bounds based on proxy-models

We introduce a generalization bound (see Theorem 1.8.2 below) for fully-connected networks with LeakyReLU activation functions. Our bound increases with training time, and decreases as LeakyReLU gets closer to identity. This allows us to get a non-vacuous bound at the beginning of training for nearly-linear networks in some settings.

Our bound is a-priori because it exploits the fact that a network with nonlinearities close to identity stays close to a "proxy" model which has a small number of parameters. For such parametric models, even simple counting-based bounds (similar to the one of Vapnik and Chervonenkis (1971)) are small enough to be non-vacuous.

Before proceeding to our bound, we describe the general idea of generalization bounds based on proxy-models.

General idea. Let f be our trained model (we omit the "hat" from now on). The idea is to come up with a "proxy" model g such that (1) g is close to f in some sense, and (2) the generalization gap of g could be well-bounded. Given this, we bound the distribution risk of f as follows:

$$R[f] \le \hat{R}_\gamma[f] + \left|R^C_\gamma[g] - \hat{R}^C_\gamma[g]\right| + \frac{1}{\gamma}\mathbb{E}_{x\sim\mathcal{D}}|f(x)-g(x)| + \frac{1}{\gamma}\frac{1}{m}\sum_{i=1}^{m}|f(x_i)-g(x_i)|, \tag{1.86}$$

where $\gamma > 0$ and we used three different risk functions:

1. Misclassification risk: $r(y,\hat{y}) = [y\hat{y} < 0]$,
2. Margin risk: $r_\gamma(y,\hat{y}) = [y\hat{y} < \gamma]$,
3. Continuous margin risk: $r^C_\gamma(y,\hat{y}) = [y\hat{y} < 0] + \left(1 - \frac{y\hat{y}}{\gamma}\right)[y\hat{y} \in [0,\gamma]]$,

giving rise to R, R_γ, R^C_γ, and their empirical counterparts.

We derived Eq. (1.86) simply from the fact that $r^C_\gamma(\cdot, y)$ is $1/\gamma$-Lipschitz and bounds $r_\gamma(\cdot, y)$ from below. The first term on the right hand side of Eq. (1.86) empirical margin risk of the original model f. The second term is the generalization gap of the proxy-model g (with respect to continuous margin risk) which we assume to be easy to bound. The two remaining terms are deviations of g from f averaged over the whole data distribution $\mathcal{D}$ and the dataset, divided by γ. The parameter γ controls the trade-off between the first term and the other three: the first term increases with γ (eventually reaching 1), while the other three decrease.

1.8.4 Our bound, informal derivation

Model. The model we consider is a fully-connected LeakyReLU network with L layers:

$$f^\epsilon_\theta(x) = W_L x^\epsilon_{\theta,L-1}(x), \qquad x^\epsilon_{\theta,0}(x) = x, \qquad x^\epsilon_{\theta,l}(x) = \phi^\epsilon\left(W_l x^\epsilon_{\theta,l-1}(x)\right) \quad \forall l \in [L-1], \tag{1.87}$$

where $\theta \in \mathbb{R}^N$ denotes the vector of all weights, i.e. $\theta = \text{cat}(\{\text{vec}(W_l)\}_{l=1}^L)$, $W_l \in \mathbb{R}^{n_l \times n_{l-1}}$ $\forall l \in [L]$, and ϕ^ϵ is a Leaky ReLU with a negative slope $1-\epsilon$, that is:

$$\phi^\epsilon(x) = x - \epsilon \min(0, x). \tag{1.88}$$

Since we are going to consider only binary classification in the present work, we take $n_L = 1$. We also define $d = n_0$ to denote the input dimension.

Training. We train our model with gradient flow on a dataset (X, Y) of size m. We denote the weights θ resulted from training the model f_θ^ϵ for time t by $\theta^\epsilon(t)$. That is, $f^1_{\theta^1(t)}$ denotes a ReLU network trained for time t, while $f^0_{\theta^0(t)}$ denotes a linear network trained for the same time.

We assume the weight initialization $\theta^\epsilon(0)$ to be random, but the same for all ϵ. That is, we assume $\theta^1(t)$ and $\theta^0(t)$ be the results of gradient flow for models with different epsilons, but starting from the same initialization $\theta^1(0) = \theta^0(0)$.

Proxy model, Option 0. What one could propose as a proxy model for $f^\epsilon_{\theta^\epsilon(t)}$ is the corresponding trained linear network $f^0_{\theta^0(t)}$. It is easy to see that the deviation on any datapoint x is of the order of ϵ:

$$\left| f^0_{\theta^0(t)}(x) - f^\epsilon_{\theta^\epsilon(t)}(x) \right| = O_{\epsilon \to 0}(\epsilon), \tag{1.89}$$

where we hid the dependence on x and t on the right-hand side: intuitively, the above deviation should grow with training time.

Since $f^0_{\theta^0(t)}(x)$ is linear in $x \in \mathbb{R}^d$, its generalization gap could be bounded uniformly over the class of linear models on $\mathbb{R}^d$ (which is much smaller than the class of all models realizable by f^ϵ_θ). For this class, the classical bound of Vapnik and Chervonenkis (1971) becomes non-vacuous as long as the input dimension d is small enough compared to the dataset size m:

$$R^C_\gamma \left[f^0_{\theta^0(t)} \right] - \hat{R}^C_\gamma \left[f^0_{\theta^0(t)} \right] \le \sqrt{\frac{8d}{m}\left(1 - \ln\left(\frac{d}{m}\right)\right)} + \sqrt{\frac{\ln(1/\delta)}{2m}} \qquad \text{w.p. } \ge 1-\delta \text{ over } (X, Y). \tag{1.90}$$

Plugging this into Eq. (1.86) already gives a bound for $R\left[f^\epsilon_{\theta^\epsilon(t)} \right]$. However, the resulting bound could be improved by choosing a different proxy-model g.

Proxy model, Option 1. Instead of the linear model, we could consider a nonlinear model with weights from the trained linear model $f^\epsilon_{\theta^0(t)}$. Its deviation from the original model is also linear in ϵ and grows with t, but the constant is smaller.

Naively, since the new proxy-model $f^\epsilon_{\theta^0(t)}(x)$ is not linear in x, the bound similar to Eq. (1.90) no longer holds. However, one could show that whenever the loss function is square, $\theta^0(t)$ is fully determined by (1) the initialization $\theta(0)$, (2) $YX^\top \in \mathbb{R}^{1\times d}$, and (3) $XX^\top \in \mathbb{R}^{d\times d}$; not by $X \in \mathbb{R}^{d\times m}$ and $Y \in \mathbb{R}^{1\times m}$ separately. If we assume $\theta(0)$ to be fixed,[10] and the data to be *whitened*, i.e. $XX^\top = I$, $\theta^0(t)$ is determined only by $YX^\top$ which has d parameters. Therefore the class of models realizable by $f^\epsilon_{\theta^0(t)}$ for a fixed t and $\theta(0)$ is determined by d parameters, same as we had for the linear proxy-model above. Therefore Eq. (1.90) holds also for $f^\epsilon_{\theta^0(t)}$.

[10] It could still be random.

The new proxy results in a slightly better bound since the deviation terms are smaller. This bound, however, still depends linearly on ϵ. In the sequel, we present a proxy that deviates quadratically from the original model.

Proxy model, Option 2. Consider the same proxy as before, but "correct" it on the training dataset:

$$g_t^\epsilon(x) = f^\epsilon_{\theta^0(t)}(x) + \left(f^\epsilon_{\theta^\epsilon(t)}(X) - f^\epsilon_{\theta^0(t)}(X)\right) X^+ x. \tag{1.91}$$

That is, we add a correction term which is a linear model that aims to fit the deviation between the original model and the previous proxy-model on the training dataset X. In our paper, we prove that the deviation becomes quadratic in ϵ:

$$\left|g_t^\epsilon(x) - f^\epsilon_{\theta^\epsilon(t)}(x)\right| = O_{\epsilon\to 0}(\epsilon^2). \tag{1.92}$$

As explained above, the first term of the new proxy could be described with d parameters. Since the second term is linear in x, it could also be described with d parameters. Therefore a bound similar to that of Eq. (1.90) holds, but with $2d$ instead of d:

$$R^C_\gamma\left[g_t^\epsilon\right] - \hat{R}^C_\gamma\left[f^0_{\theta^0(t)}\right] \le \sqrt{\frac{16d}{m}\left(1 - \ln\left(\frac{2d}{m}\right)\right)} + \sqrt{\frac{\ln(1/\delta)}{2m}} \qquad \text{w.p. } \ge 1-\delta \text{ over } (X,Y). \tag{1.93}$$

Our bound. The resulting bound for a fully-connected network trained with gradient flow takes the following form:

$$R\left[f^\epsilon_{\theta^\epsilon(t)}\right] \le \hat{R}_\gamma\left[f^\epsilon_{\theta^\epsilon(t)}\right] + \Upsilon\left(\frac{\kappa d}{m}, \frac{\ln(1/\delta)}{m}\right) + \frac{\Delta_\kappa(t)\epsilon^\kappa}{\gamma} \qquad \text{w.p. } \ge 1-\delta \text{ over } (X,Y), \tag{1.94}$$

where κ is either one or two, corresponding to proxy models of Options 1 and 2, respectively. Here the Υ-term abbreviates the bound for the proxy-model, while $\Delta_\kappa(t)\epsilon^\kappa$ bounds the deviation. We use a slightly different counting-based bound than that of Vapnik and Chervonenkis (1971) for Υ in our final theorem, as it turns out to be numerically smaller.

Advantages.

1. The bound is a-priori: the difference $R\left[f^\epsilon_{\theta^\epsilon(t)}\right] - \hat{R}_\gamma\left[f^\epsilon_{\theta^\epsilon(t)}\right]$ can be evaluated without training f^ϵ_θ.
2. The above bound does not depend on network width $n_{1:L-1}$. Whereas the only non-vacuous bound we are aware of that (probably) does not depend on width is the one of Zhou et al. (2019).
3. It holds for the original trained model, not a proxy. The only non-vacuous bound we are aware of that posesses this property is the one Biggs and Guedj (2022).
4. To the best of our knowledge, our bound is the first to take the implicit bias of the learning rule into account. We do it by exploiting the fact that a network with small ϵ stays close to the proxy-model we consider, while the latter has few "effective" parameters.
5. Lastly, our bound rehabilitates classical parameter-counting bounds, besides them being deemed to be useless for neural nets of realistic size.

Disadvantages.

1. The deviation term $\Delta_\kappa(t)$ grows with time t really fast: $\log\log\Delta_\kappa(t) = O(t)$. Therefore our bound could be non-vacuous only at the beginning of training. In our paper, we demonstrate both experimentally and theoretically that the bound eventually becomes non-vacuous, before exploding due to Δ_κ.
2. The Υ term gets too large even for full-sized MNIST. We believe this term could be improved with tighter bounds for linear models; we hope, these bounds could also be adapted to our proxy-models that use linear model's weights.

Directions for improvement.

1. According to our experiments, the deviation bound $\Delta_\kappa(t)$, while the real deviation indeed grows with t, is quite loose. We believe it could be improved with a more thorough analysis.
2. The proxy-model generalization bound Υ is also loose. Since the linear model could be integrated analytically in some scenarios (Saxe et al., 2013), the $\kappa = 1$ proxy-model $f^\epsilon_{\theta^0}$ is analytically computable. This suggests that its generalization gap could also be computed analytically.
3. Whether higher-order proxy-models with reasonable generalization gap bounds could be constructed, is an interesting open question.

1.8.5 Our bound, formal statement

Now we are ready to state our bound formally. We start with describing the setup we work with.

Model. The model is a fully-connected LeakyReLU network with L layers, as already described above, with ϵ measuring the deviation from linearity.

Data. Data points (x, y) come from a distribution $\mathcal{D}$. We assume all x from $\mathcal{D}$ to lie on a unit ball, $\|x\| \le 1$, and all y to equal ± 1. During the training phase, we sample a set of m points iid from $\mathcal{D}$ to form a dataset (X, Y), where $X \in \mathbb{R}^{d\times m}$ and $Y \in \mathbb{R}^{1\times m}$.

Training. Assuming $\operatorname{rk} X = d$ (which implies $m \ge d$, i.e. the data is abundant), we train our model on (X, Y) with gradient flow to optimize square loss on whitened data, i.e. on $(\tilde{X}, Y)$ for $\tilde{X} = \Sigma_X^{-1/2} X$, where $\Sigma_X = \frac{1}{m} XX^\top \in \mathbb{R}^{d\times d}$ is an empirical feature correlation matrix. That is,

$$\frac{dW_l^\epsilon(t)}{dt} = -\frac{\partial\left(\frac{1}{2m}\left\|Y - f^\epsilon_{\theta^\epsilon(t)}(\tilde{X})\right\|_F^2\right)}{\partial W_l} \quad \forall l \in [L]. \tag{1.95}$$

Note that $\tilde{X}\tilde{X}^\top = mI_d$.

Inference. To conform with the above training procedure, we take the model output at a point x to be $f^\epsilon_\theta\left(\Sigma^{-1/2}x\right)$, where Σ is a (distribution) feature correlation matrix: $\Sigma = \mathbb{E}_{(x,y)\sim\mathcal{D}}(xx^\top) \in \mathbb{R}^{d\times d}$. We assume this matrix to be known; in practice, we could substitute it with Σ_X.

Performance measure. With a slight abuse of notation, we denote the risk of the model at training time t as follows:

$$\hat{R}^\epsilon(t) = \frac{1}{m}\sum_{k=1}^{m} r\left(f^\epsilon_{\theta^\epsilon(t)}(\Sigma^{-1/2}x_k), y_k\right), \qquad R^\epsilon(t) = \mathbb{E}_{(x,y)\sim\mathcal{D}} r\left(f^\epsilon_{\theta^\epsilon(t)}(\Sigma^{-1/2}x), y\right), \tag{1.96}$$

where $r(\cdot,\cdot)$ is misclassification risk as before. We define $R^\epsilon_\gamma(t)$ and $R^{C,\epsilon}_\gamma(t)$ analogously to $R^\epsilon(t)$, and $\hat{R}^\epsilon_\gamma$ and $\hat{R}^{C,\epsilon}_\gamma$ analogously to $\hat{R}^\epsilon$.

The bound. We will need the following assumption on the training process:

Assumption 1.8.1. $\forall t \geq 0 \left\|\left(Y - f^\epsilon_{\theta^\epsilon(t)}(\tilde{X})\right)\tilde{X}^\top\right\|_F^2 \leq \left\|\left(Y - f^\epsilon_{\theta^\epsilon(0)}(\tilde{X})\right)\tilde{X}^\top\right\|_F^2$.

Note that $\left\|Y - f^\epsilon_{\theta^\epsilon(t)}(\tilde{X})\right\|_F^2 \leq \left\|Y - f^\epsilon_{\theta^\epsilon(0)}(\tilde{X})\right\|_F^2$ since we minimize the loss monotonically with gradient flow. This implies that the above assumption holds automatically, whenever $\tilde{X}$ is a (scaled) orthogonal matrix (which happens when $m = d$). It is easy to show that it provably holds for a linear network ($\epsilon = 0$), and we found this assumption to hold empirically for all of our experiments with nonlinear networks too.

We are now ready to formulate our main result:

Theorem 1.8.2 (Golikov (2025))**.** *Fix $\beta, \gamma > 0$, $t \geq 0$, $\delta \in (0,1)$, $\epsilon \in [0,1]$, and $\kappa \in \{1,2\}$. Let p be the floating point arithmetic precision (32 by default). Under the above setting and Assumption 1.8.1, for any weight initialization satisfying $\|W^\epsilon_l(0)\| \leq \beta\ \forall l \in [L]$, w.p. $\geq 1-\delta$ over sampling the dataset (X,Y),*

$$R^\epsilon(t) \leq \hat{R}^\epsilon_\gamma(t) + \Upsilon_\kappa + \frac{\Delta_{\kappa,\beta}(t)\epsilon^\kappa}{\gamma}, \tag{1.97}$$

where

$$\Upsilon_\kappa = \sqrt{\frac{\kappa p d \ln 2 + \ln(1/\delta)}{2m}}, \tag{1.98}$$

and

$$\Delta_{\kappa,\beta}(t) = \Phi_\kappa v_\beta(t) u_\beta^{L-1}(t), \tag{1.99}$$

where

$$\Phi_\kappa = \begin{cases} L\sqrt{d} + 1 + (L-1)\rho, & \kappa = 1; \\ (L-1)\left[(L+1+(L-1)\rho)\sqrt{d} + 2(1+(L-1)\rho)\right], & \kappa = 2, \end{cases} \tag{1.100}$$

where $\rho = \frac{\|W^\epsilon_1(0)\|_F}{\|W^\epsilon_1(0)\|}$ is the square root of the stable rank of the input layer at initialization, and the definitions of u_β and v_β are given below.

Define

- $s = \left\|YX^\top\Sigma_X^{-1/2}\right\|$; $\quad \bar{1} = 1 + \rho\beta^L$;
- *For a given $r \geq 0$, $\bar{s}_r = (1-\epsilon)(s + \rho\beta^L) + \epsilon\sqrt{r}(L-1)(1+\rho\beta^L)$;*
- $\bar{s} = \bar{s}_1$; $\quad \hat{s} = \frac{L}{1+(L-1)\rho}\bar{s}$; $\quad \bar{\beta} = \beta\frac{\bar{s}_\rho}{\bar{s}_1}$.

We have

$$u_\beta(t) = \begin{cases} \bar{\beta} e^{\bar{s}t}, & L = 2; \\ \left(\bar{\beta}^{2-L} - (L-2)\bar{s}t\right)^{\frac{1}{2-L}}, & L \geq 3; \end{cases} \tag{1.101}$$

$$v_\beta(t) = \frac{L-1}{L} \hat{s}^{\frac{2-L}{L}} u_\beta^{L-1}(t)[w(u_\beta(t)) - w(\beta)] e^{\frac{u_\beta^L(t)}{\hat{s}}}, \tag{1.102}$$

where [11]

$$w(u) = -\frac{\bar{1}}{\bar{s}} \Gamma\left(\frac{2-L}{L}, \frac{u^L}{\hat{s}}\right) - \rho \frac{\hat{s}}{\bar{s}} \Gamma\left(\frac{2}{L}, \frac{u^L}{\hat{s}}\right). \tag{1.103}$$

[11] Γ is an upper-incomplete gamma-function defined as $\Gamma(s,x) = \int_x^\infty t^{s-1} e^{-t}\, dt$.

2 Neural tangent kernel: a survey

Authors: Eugene Golikov, Eduard Pokonechniy, Vladimir Korviakov.

Link and reference: Golikov et al. (2022), accessible on arXiv: https://arxiv.org/abs/2208.13614.

Contribution: everything except for the experiments in Section 2.9 was done by the author of the present thesis.

2.1 Definition and the explicit solution for square loss

Consider a generic parametric model $f(x;\theta) : \mathcal{X} \times \mathbb{R}^N \to \mathbb{R}$ differentiable with respect to weights θ. We aim to minimize square loss over a dataset $(\vec{x}, \vec{y})$ of size m: $\frac{1}{2}\sum_{j=1}^{m}(y_j - f(x_j;\theta))^2 \to \min_\theta$. A continuous-time gradient descent dynamics (gradient flow) corresponds to the following ordinary differential equation (ODE):

$$\dot{\theta}_t = -\nabla_\theta \left(\frac{1}{2}\sum_{j=1}^{m}(y_j - f(x_j;\theta_t))^2 \right) = \sum_{j=1}^{m}(y_j - f(x_j;\theta_t))\nabla_\theta f(x_j;\theta_t). \tag{2.1}$$

Let us abbreviate the prediction at a given data point x at time t, $f(x;\theta_t)$, as $f_t(x)$. Under the dynamics above, this quantity evolves as

$$\dot{f}_t(x) = \dot{\theta}_t^T \nabla f_t(x) = \sum_{j=1}^{m}(y_j - f_t(x_j))\nabla_\theta^T f_t(x_j)\nabla_\theta f_t(x). \tag{2.2}$$

If we perceive $\nabla_\theta f_t(x)$ as a feature map $\Phi_t : \mathcal{X} \to \mathbb{R}^N$, the scalar product above becomes a kernel evaluated at a pair (x_j, x). This kernel is called an empirical neural tangent kernel (NTK) and is denoted by $\hat{\Theta}_t$:

$$\hat{\Theta}_t(x, x') = \nabla_\theta^T f_t(x)\nabla_\theta f_t(x'). \tag{2.3}$$

This definition allows for a shorter representation of the prediction dynamics (2.2):

$$\dot{f}_t(x) = \hat{\Theta}_t(x, \vec{x})(\vec{y} - f_t(\vec{x})), \tag{2.4}$$

where by convention, $\hat{\Theta}_t(x, \vec{x}) \in \mathbb{R}^{1\times m}$.

Assume that the empirical NTK does not evolve with time, i.e $\hat{\Theta}_t(x, x') = \hat{\Theta}_0(x, x')\ \forall x, x' \in \mathcal{X}$. This assumption is equivalent to assuming the model $f(x;\theta)$ to be linear as a function of its weights:

$$f(x;\theta) = f(x;\theta_0) + \nabla_\theta^T f(x;\theta_0)(\theta - \theta_0). \tag{2.5}$$

When the kernel is constant, Eq.(2.4) is easily integrable. Indeed, on the train dataset,

$$\dot{f}_t(\vec{x}) = \hat{\Theta}_0(\vec{x}, \vec{x})(\vec{y} - f_t(\vec{x})), \tag{2.6}$$

which gives

$$f_t(\vec{x}) = f_0(\vec{x}) - \left(I - e^{-\hat{\Theta}_0(\vec{x},\vec{x})t}\right)(f_0(\vec{x}) - \vec{y}). \tag{2.7}$$

Plugging it back to Eq.(2.4) gives

$$\dot{f}_t(x) = \hat{\Theta}_t(x, \vec{x}) e^{-\hat{\Theta}_0(\vec{x},\vec{x})t}(\vec{y} - f_0(\vec{x})), \tag{2.8}$$

and finally,

$$f_t(x) = f_0(x) - \hat{\Theta}_0(x, \vec{x})\hat{\Theta}_0^{-1}(\vec{x}, \vec{x})\left(I - e^{-\hat{\Theta}_0(\vec{x},\vec{x})t}\right)(f_0(\vec{x}) - \vec{y}). \tag{2.9}$$

While the exact solution above is based on the constant kernel assumption, one can prove that the kernel is indeed nearly constant in certain settings, see Section 2.2. This allows one to transfer results that hold for linearized models to original ones.

For example, $f_t(\vec{x})$ converges to $\vec{y}$ (i.e. the model learns the dataset) as long as the Gram matrix is positive definite: $\hat{\Theta}_0(\vec{x}, \vec{x}) \geq \lambda_0$ for some $\lambda_0 > 0$, see Eq.(2.6). The same result holds without the constant kernel assumption, as long as $\hat{\Theta}_t(\vec{x}, \vec{x})$ stays sufficiently close to $\hat{\Theta}_0(\vec{x}, \vec{x})$, and therefore, say, $\hat{\Theta}_t(\vec{x}, \vec{x}) \geq \lambda_0/2$. Indeed,

$$\frac{d}{dt}\left(\frac{1}{2}\|\vec{y} - f_t(\vec{x})\|_2^2\right) = -(\vec{y} - f_t(\vec{x}))^T \hat{\Theta}_t(\vec{x}, \vec{x})(\vec{y} - f_t(\vec{x})) \leq -\frac{\lambda_0}{2}\|\vec{y} - f_t(\vec{x})\|_2^2, \tag{2.10}$$

which gives

$$\|\vec{y} - f_t(\vec{x})\|_2^2 \leq e^{-\lambda_0 t}\|\vec{y} - f_0(\vec{x})\|_2^2 \to 0 \quad \text{as } t \to \infty; \tag{2.11}$$

see Du et al. (2019b) for the formal result. This result is not trivial, since loss surfaces of generic neural nets are non-convex, and therefore any local optimization method (e.g. the gradient flow) may get stuck in a spurious local minimum. See Arora et al. (2019a) for other results of a similar kind.

Also, if one assumes the kernel to be nearly constant, one can identify certain pathologies affecting the learning process by analyzing the initial kernel: see Martens et al. (2021) discussing trainability of very deep nets and Dupuis and Jacot (2021); Tancik et al. (2020) fixing blurry results of image regression.

Finally, the exact solution (2.9) can be used as a substitute for the usual gradient descent training routine. A naive approach for evaluating Eq.(2.9) would be to compute the initial kernel $\hat{\Theta}_0(\vec{x}, \vec{x})$ and then to invert it. Naively computing the kernel requires $O(Nm^2)$ time and $O(m^2)$ memory, while inverting it takes $O(m^3)$ more time. Such an approach is infeasible for datasets of realistic sizes (i.e. $m \gtrsim 10^5$), asking for major optimizations, see Novak et al. (2020, 2021); Meanti et al. (2020). Nevertheless, for $m \lesssim 10^4$, the direct approach is feasible and gives promising results, see Arora et al. (2020). Also, in certain scenarios, the kernel can be efficiently scaled from small m to larger ones, see Radhakrishnan et al. (2022).

2.2 Kernel convergence

The goal of this section is to validate the constant kernel assumption: $\hat{\Theta}_t(x, x') = \hat{\Theta}_0(x, x')\ \forall x, x' \in \mathcal{X}$. The main result is: under certain parameterization, the empirical NTK of a neural network becomes constant as width goes to infinity. Before stating this result formally, we provide an illustrative example.

Consider a neural network with one hidden layer, scalar input, and Gaussian-initialized weights:

$$f(x; a_{1:n}, w_{1:n}) = \sum_{i=1}^{n} a_i \phi(w_i x), \quad a_{1:n} \sim \mathcal{N}(0, n^{-1} I), \quad w_{1:n} \sim \mathcal{N}(0, I). \tag{2.12}$$

Here n is width of the hidden layer; following a standard initialization scheme (He et al., 2015), initialization variance of each layer is inversely proportional to the number of input neurons.

The above parameterization of the network is the one typically used in practice; we shall refer it as standard. However, the parameterization we need is a different one:

$$f(x; a_{1:n}, w_{1:n}) = \frac{1}{\sqrt{n}} \sum_{i=1}^{n} a_i \phi(w_i x), \quad a_{1:n} \sim \mathcal{N}(0, I), \quad w_{1:n} \sim \mathcal{N}(0, I). \tag{2.13}$$

We shall refer it as NTK-parameterization. Note that it does not alter the distribution of neurons, both hidden and output, at initialization but it does alter the gradient flow:

$$\dot{a}_k = \frac{1}{\sqrt{n}} \sum_{j=1}^{m} \phi(w_k x_j), \quad \dot{w}_k = \frac{1}{\sqrt{n}} \sum_{j=1}^{m} a_k \phi'(w_k x_j) x_j. \tag{2.14}$$

Here input and output weights receive $O(n^{-1/2})$ increments, while both of them are $O(1)$ at initialization. Hence $a_k(t) \to a_k(0)$ and $w_k(t) \to w_k(0)$ as $n \to \infty$ for any fixed $k \in \mathbb{N}$ and $t \in \mathbb{R}_+$.

Compare with gradient flow under standard parameterization:

$$\dot{a}_k = \sum_{j=1}^{m} \phi(w_k x_j), \quad \dot{w}_k = \sum_{j=1}^{m} a_k \phi'(w_k x_j) x_j. \tag{2.15}$$

Here the output weights are $O(n^{-1/2})$ at initialization but receive $O(1)$ increments for $t = 0$, while the input weights are $O(1)$ at initialization but receive $O(n^{-1/2})$ increments for $t = 0$.

Let us write the NTK under NTK parameterization:

$$\begin{aligned} \hat{\Theta}_t(x, x') &= \sum_{i=1}^{n} \left(\partial_{a_i} f(x) \partial_{a_i} f(x') + \partial_{w_i} f(x) \partial_{w_i} f(x') \right) \\ &= \frac{1}{n} \sum_{i=1}^{n} \left(\phi(w_i(t)x) \phi(w_i(t)x') + a_i^2(t) \phi'(w_i(t)x) \phi'(w_i(t)x') x x' \right). \end{aligned} \tag{2.16}$$

Since $a_k(t) \to a_k(0)$ and $w_k(t) \to w_k(0)$ as $n \to \infty$ for any fixed $k \in \mathbb{N}$ and $t \in \mathbb{R}_+$, the above expression is asymptotically equivalent to

$$\hat{\Theta}_0(x, x') = \frac{1}{n} \sum_{i=1}^{n} \left(\phi(w_i(0)x) \phi(w_i(0)x') + a_i^2(0) \phi'(w_i(0)x) \phi'(w_i(0)x') x x' \right), \tag{2.17}$$

which converges (almost surely) to

$$\Theta(x, x') = \mathbb{E}_{a,w\sim\mathcal{N}(0,1)} \left(\phi(wx)\phi(wx') + a^2\phi'(wx)\phi'(wx')xx'\right) \tag{2.18}$$

as $n \to \infty$ due to the (strong) Law of Large Numbers. The limit kernel $\Theta(x, x')$ depends neither on a timestep t, nor on initialization. This kernel is typically referred as NTK, contrasting to the empirical NTK $\hat{\Theta}_t$.

Since under standard parameterization the weights receive increments asymptotically at least comparable to initialization, one cannot expect that the empirical NTK stops evolving as $n \to \infty$ in this setting. Moreover, the initial empirical NTK diverges with width:

$$\hat{\Theta}_0(x, x') = \sum_{i=1}^{n} \left(\phi(w_i(0)x)\phi(w_i(0)x') + a_i^2(0)\phi'(w_i(0)x)\phi'(w_i(0)x')xx'\right) \sim$$

$$\sim n \times \mathbb{E}_{w\sim\mathcal{N}(0,1)}\phi(wx)\phi(wx'). \tag{2.19}$$

The above kernel convergence result holds in more general settings. Consider a fully-connected network with L layers under NTK parameterization:

$$f(x) = h_L(x), \quad h_l(x) = \frac{1}{\sqrt{n_{l-1}}} W_l x_{l-1}(x), \quad x_{l-1}(x) = \phi(h_{l-1}(x)), \quad x_0(x) = x, \tag{2.20}$$

where $W_1 \in \mathbb{R}^{n_1\times n_0}$, $W_L \in \mathbb{R}^{1\times n_{L-1}}$, and $W_l \in \mathbb{R}^{n_l\times n_{l-1}}$ for all other l. Here all weights are initialized with independent standard Gaussians. Suppose we aim to optimize a generic differentiable loss ℓ instead of the quadratic one:

$$\dot{\theta}_t = -\nabla_\theta \left(\sum_{j=1}^{m} \ell(y_j, f(x_j; \theta_t))\right) = \sum_{j=1}^{m} \left.\frac{\partial \ell(y_j, z)}{\partial z}\right|_{z=f(x_j;\theta_t)} \nabla_\theta f(x_j; \theta_t), \tag{2.21}$$

where θ now is a concatenation of all weights $W_{1:L}$. The seminal work of Jacot et al. (2018) proves the following:

Theorem 2.2.1 (Jacot et al. (2018))**.** *Under the conditions above, for ϕ being C^2 and Lipschitz and ℓ being C^1 and Lipschitz, $\hat{\Theta}_t(x, x') \to \Theta(x, x')$ in probability as $n_{1:L-1} \to \infty$ sequentially $\forall x, x' \in \mathcal{X}$ $\forall t \geq 0$.*

In fact, the theorem above can be generalized far from fully-connected nets with smooth activation functions. Define a tensor program as a set of initial variables of certain types and a sequence of operations. Each of the operations generates a new variable by acting on previously generated ones. The variable types are

1. A: $n \times n$ matrices with iid $\mathcal{N}(0, 1)$ entries;
2. G: vectors of size n with asymptotically iid Gaussian entries;
3. H: images of G-vars by coordinatewise nonlinearities.

The operations are

1. Trsp: $W : \mathsf{A} \to W^\top : \mathsf{A}$;

2. MatMul: $(W : \mathsf{A},\ x : \mathsf{H}) \to \frac{1}{\sqrt{n}} Wx : \mathsf{G}$;

3. LinComb: $(\{x_i : \mathsf{G},\ a_i \in \mathbb{R}\}_{i=1}^k) \to \sum_{i=1}^k a_i x_i : \mathsf{G}$;

4. Nonlin: $(\{x_i : \mathsf{G}\}_{i=1}^k,\ \phi : \mathbb{R}^k \to \mathbb{R}) \to \phi(x_{1:k}) : \mathsf{H}$.

The set of initial variables consists of variables of A-type and G-type. As for input G-vars, we sample $\{x_\alpha : x \text{ is an input G-var}\} \sim \mathcal{N}(\mu^{in}, \Sigma^{in})\ \forall \alpha \in [n]$.

The above formalism allows to express forward and backward passes of a very wide class of neural nets (including RNNs, ResNets, and Transformers). Besides none of the operations above generates new A-vars (new weights), the whole gradient descent training process can be expressed as a single tensor program by backtracking the gradient steps. The real power of tensor programs comes from the following theorem:

Theorem 2.2.2 ("Master theorem", Yang (2020b))**.** *Consider a tensor program with M G-vars, under above assumptions. Suppose all the nonlinearities ϕ and a function $\psi : \mathbb{R}^M \to \mathbb{R}$ are polynomially bounded. Then the following holds:*

$$\frac{1}{n} \sum_{\alpha=1}^{n} \psi(g_\alpha^1, \dots, g_\alpha^M) \to \mathbb{E}_{Z \sim \mathcal{N}(\mu, \Sigma)} \psi(Z) \tag{2.22}$$

a.s. as $n \to \infty$, where μ and Σ can be computed using certain recurrent rules.

It is possible to define the empirical NTK of a tensor program and express it in the form $\frac{1}{n} \sum_{\alpha=1}^n \psi(g_\alpha^1, \dots, g_\alpha^M)$ for a certain function ψ. Then the kernel converges by virtue of the above theorem. See Yang (2020a) for the proof of initial kernel convergence and Yang and Littwin (2021) for the proof of kernel convergence for any timestep.

As an illustration, recall the two-layered net considered at the beginning of the present section. Its empirical NTK is given by

$$\hat{\Theta}_0(x, x') = \frac{1}{n} \sum_{i=1}^{n} \left(\phi(w_i(0)x)\phi(w_i(0)x') + a_i^2(0)\phi'(w_i(0)x)\phi'(w_i(0)x')xx' \right). \tag{2.23}$$

Here G-vars are $g^1 = w(0)x$, $g^2 = w(0)x'$, $g^3 = a(0)x$, $g^4 = a(0)x'$. Taking $\psi(g_\alpha^1, \dots, g_\alpha^4) = \phi(g_\alpha^1)\phi(g_\alpha^2) + \phi'(g_\alpha^1)\phi'(g_\alpha^2)g_\alpha^3 g_\alpha^4$ allows for explicit application of Theorem 2.2.2.

2.3 Finite-width corrections

While the results discussed in Section 2.2 hold in the limit of infinite width, they are not directly applicable to real-life finite-width nets for obvious reasons. This motivates one to introduce finite-width corrections for the limit NTK.

First, define a higher-order kernel:

$$O_{s,t}(x_{1:s}) = \nabla_\theta^T O_{s-1,t}(x_{1:s-1}) \nabla_\theta f_t(x_s). \tag{2.24}$$

Put $O_{1,t}(x_1) = f_t(x_1)$; this gives $O_{2,t}(x_1, x_2) = \hat{\Theta}_t(x_1, x_2)$.

Consider a gradient flow optimization process under square loss:

$$\dot{\theta}_t = \sum_{j=1}^{m} (y_j - f_t(x_j)) \nabla_\theta f_t(x_j). \tag{2.25}$$

Under this process, the s-order kernel evolves as

$$\dot{O}_{s,t}(x_{1:s}) = \nabla_\theta^T O_{s,t}(x_{1:s}) \dot{\theta} = O_{s+1,t}(x_{1:s}, \vec{x})(\vec{y} - f_t(\vec{x})). \tag{2.26}$$

This gives an infinite system of ODE's governing the evolution of the kernels.

If our goal is to obtain a solution only up to the order of n^{-1}, will it allow us to truncate the initially infinite system? How many equations should we keep? In order to answer these questions, let us estimate the order of growth for $O_{s,t}$.

Following Dyer and Gur-Ari (2020), we start with a definition of a correlation function. Let us fix $t = 0$ and omit the corresponding subscript for now. Define a rank-k derivative tensor $T_{\mu_1 \dots \mu_k}$ as follows:

$$T_{\mu_1 \dots \mu_k}(x; f) = \frac{\partial^k f(x)}{\partial \theta^{\mu_1} \dots \partial \theta^{\mu_k}}. \tag{2.27}$$

For $k = 0$ we define $T(x; f) = f(x)$. We are now ready to define a correlation function C:

$$C(x_1, \dots, x_m) = \sum_{\mu_1, \dots, \mu_{k_m}} \Delta^{(\pi)}_{\mu_1 \dots \mu_{k_m}} \mathbb{E}_\theta \left(T_{\mu_1 \dots \mu_{k_1}}(x_1) T_{\mu_{k_1+1} \dots \mu_{k_2}}(x_2) \dots T_{\mu_{k_{m-1}+1} \dots \mu_{k_m}}(x_m) \right). \tag{2.28}$$

Here $0 \le k_1 \le \dots \le k_m$, k_m and m are even, $\pi \in S_{k_m}$ is a permutation, and $\Delta^{(\pi)}_{\mu_1 \dots \mu_{k_m}} = \delta_{\mu_{\pi(1)} \mu_{\pi(2)}} \dots \delta_{\mu_{\pi(k_m - 1)} \mu_{\pi(k_m)}}$. For example,

$$\begin{aligned} \mathbb{E}_\theta (f(x) \nabla_\theta^T f(x) \nabla_\theta \nabla_\theta^T f(x_1) \nabla_\theta f(x_2)) &= \sum_{\mu,\nu} \mathbb{E}_\theta (f(x) \partial_\mu f(x) \partial^2_{\mu,\nu} f(x_1) \partial_\nu f(x_2)) \\ &= \sum_{\mu_1, \mu_2, \mu_3, \mu_4} \delta_{\mu_1 \mu_2} \delta_{\mu_3 \mu_4} \mathbb{E}_\theta (f(x) \partial_{\mu_1} f(x) \partial^2_{\mu_2, \mu_3} f(x_1) \partial_{\mu_4} f(x_2)) = C(x, x, x_1, x_2) \end{aligned} \tag{2.29}$$

is a correlation function with $m = 4$, $k_1 = 0$, $k_2 = 1$, $k_3 = 3$, $k_4 = 4$, and $\pi(j) = j$.

If two derivative tensors have two indices that are summed over, we say that they are contracted. Formally, we say that $T_{\mu_{k_{i-1}+1} \dots \mu_{k_i}}(x_i)$ is contracted with $T_{\mu_{k_{j-1}+1} \dots \mu_{k_j}}(x_j)$ for $1 \le i, j \le m$ if there exists an even $s \le k_m$ such that $k_{i-1} < \pi(s-1) \le k_i$, while $k_{j-1} < \pi(s) \le k_j$, or vice versa.

Define the cluster graph $G_C(V, E)$ as a non-oriented non-weighted graph with vertices $V = \{v_1, \dots, v_m\}$ and edges $E = \{(v_i, v_j) \,|\, T(x_i) \text{ and } T(x_j) \text{ are contracted in } C\}$. Let n_e be the number of even-sized connected components of $G_C(V, E)$ and n_o be the number of odd-sized components. We are going to use the following conjecture, which is proven in certain scenarios:

Conjecture 2.3.1 (Dyer and Gur-Ari (2020))**.** *If m is even, $C(x_1, \dots, x_m) = O_{n \to \infty}(n^{s_C})$, where $s_C = n_e + n_o/2 - m/2$. If m is odd, $C(x_1, \dots, x_m) = 0$.*

We are also going to use the following lemma:

Lemma 2.3.2 (Dyer and Gur-Ari (2020))**.** *Suppose Conjecture 2.3.1 holds. Let $C(\vec{x}) = \mathbb{E}_\theta F(\vec{x}; \theta)$ be a correlation function and suppose $C(\vec{x}) = O(n^{s_C})$ for s_C defined in Conjecture 2.3.1. Then $\mathbb{E}_\theta d^k F(\vec{x}; \theta)/dt^k = O(n^{s_C})$ $\forall k \ge 1$.*

Proof. Consider the first derivative:

$$\mathbb{E}_\theta \frac{dF(\vec{x})}{dt} = \mathbb{E}_\theta(\dot{\theta}^T \nabla_\theta F(\vec{x})) = \mathbb{E}_{x,y}\mathbb{E}_\theta(\eta(y - f(x))\nabla_\theta^T f(x)\nabla_\theta F(\vec{x}))$$
$$= \eta\mathbb{E}_{x,y}\mathbb{E}_\theta(y\nabla_\theta^T f(x)\nabla_\theta F(\vec{x})) - \eta\mathbb{E}_{x,y}\mathbb{E}_\theta(f(x)\nabla_\theta^T f(x)\nabla_\theta F(\vec{x})). \quad (2.30)$$

This is a sum of a linear combination of correlation functions. By Conjecture 2.3.1, the first sum evaluates to zero, while the second one has $m' = m + 2$, n'_e even clusters, and n'_o odd clusters. If $\nabla_\theta f(x)$ is contracted with an even cluster of C, we have $n'_e = n_e - 1$, $n'_o = n_o + 2$. In contrast, if $\nabla_\theta f(x)$ is contracted with an odd cluster of C, we have $n'_e = n_e + 1$, $n'_o = n_o$.

In the first case, we have $s'_C = n'_e + n'_o/2 - m'/2 = s_C - 1$, while for the second $s'_C = s_C$. In any case, the result is a linear combination of correlation functions with $s'_C \le s_C$ for each. □

Let us return the t-subscript. Since O_s has s derivative tensors and a single cluster, by virtue of Conjecture 2.3.1, $\mathbb{E}_\theta O_{s,0} = O(n^{1-s/2})$ for even s and $\mathbb{E}_\theta O_{s,0} = 0$ for odd s. At the same time, $\mathbb{E}_\theta \dot{O}_{s,0} = O(n^{1-(s+2)/2}) = O(n^{-s/2})$ for even s and $\mathbb{E}_\theta \dot{O}_{s,0} = O(n^{1-(s+1)/2}) = O(n^{1/2-s/2})$ for odd s.

As for the second moments, we have $\mathbb{E}_\theta(O_{s,0})^2 = O(n^{2-s})$ for even s and $\mathbb{E}_\theta(O_{s,0})^2 = O(n^{1-s})$ for odd s. Similarly, we have $\mathbb{E}_\theta(\dot{O}_{s,0})^2 = O(n^{2/2-(2s+2)/2}) = O(n^{-s})$ for even s and $\mathbb{E}_\theta(\dot{O}_{s,0})^2 = O(n^{2-(2s+2)/2}) = O(n^{1-s})$ for odd s.

The asymptotics for the first two moments implies the asymptotic for a random variable itself:

$$O_{s,0}(x_{1:s}) = \begin{cases} O(n^{1-s/2}) & \text{for even } s; \\ O(n^{1/2-s/2}) & \text{for odd } s; \end{cases} \qquad \dot{O}_{s,0}(x_{1:s}) = \begin{cases} O(n^{-s/2}) & \text{for even } s; \\ O(n^{1/2-s/2}) & \text{for odd } s. \end{cases} \quad (2.31)$$

Lemma 2.3.2 gives $\forall k \ge 1$:

$$\frac{d^k O_{s,t}}{dt^k}(x_{1:s})\bigg|_{t=0} = \begin{cases} O(n^{-s/2}) & \text{for even } s; \\ O(n^{1/2-s/2}) & \text{for odd } s. \end{cases} \quad (2.32)$$

Then given an analytic activation function, we have $\forall t \ge 0$:

$$\dot{O}_{s,t}(x_{1:s}) = \sum_{k=1}^{\infty} \frac{d^k O_{s,t}}{dt^k}(x_{1:s})\bigg|_{t=0} \frac{t^k}{k!} = \begin{cases} O(n^{-s/2}) & \text{for even } s; \\ O(n^{1/2-s/2}) & \text{for odd } s. \end{cases} \quad (2.33)$$

This allows us to write a finite system of ODE for the model evolution up to $O(n^{-1})$ terms:

$$\dot{f}_t(x_1) = O_{2,t}(x_1, \vec{x})(\vec{y} - f_t(\vec{x})), \qquad f_0(x_1) = f(x_1; \theta), \quad \theta \sim \mathcal{N}(0, I), \quad (2.34)$$

$$\dot{O}_{2,t}(x_1, x_2) = O_{3,t}(x_1, x_2, \vec{x})(\vec{y} - f_t(\vec{x})), \qquad O_{2,0}(x_1, x_2) = \nabla_\theta^T f_0(x_1)\nabla_\theta f_0(x_2), \quad (2.35)$$

$$\dot{O}_{3,t}(x_1, x_2, x_3) = O_{4,t}(x_1, x_2, x_3, \vec{x})(\vec{y} - f_t(\vec{x})), \qquad O_{3,0}(x_1, x_2, x_3) = \nabla_\theta^T O_{2,0}(x_1, x_2)\nabla_\theta f_0(x_3), \quad (2.36)$$

$$\dot{O}_{4,t}(x_1, x_2, x_3, x_4) = O(n^{-2}), \qquad O_{4,0}(x_1, x_2, x_3, x_4) = \nabla_\theta^T O_{3,0}(x_1, x_2, x_3)\nabla_\theta f_0(x_4). \quad (2.37)$$

Let us expand all the quantities wrt n^{-1}:

$$O_{s,t}(x_{1:s}) = O^{(0)}_{s,t}(x_{1:s}) + n^{-1}O^{(1)}_{s,t}(x_{1:s}) + O(n^{-2}), \quad (2.38)$$

where $O^{(k)}_{s,t}(x_{1:s}) = \Theta_{n\to\infty}(1)$. Then the system above transforms into the following:

$$\dot{f}^{(0)}_t(x_1) = O^{(0)}_{2,t}(x_1,\vec{x})(\vec{y} - f^{(0)}_t(\vec{x})), \tag{2.39}$$

$$\dot{f}^{(1)}_t(x_1) = O^{(1)}_{2,t}(x_1,\vec{x})(\vec{y} - f^{(0)}_t(\vec{x})) - O^{(0)}_{2,t}(x_1,\vec{x})f^{(1)}_t(\vec{x}), \tag{2.40}$$

$$O^{(0)}_{2,t}(x_1,x_2) = \nabla^T_\theta f^{(0)}_0(x_1)\nabla_\theta f^{(0)}_0(x_2), \tag{2.41}$$

$$\dot{O}^{(1)}_{2,t}(x_1,x_2) = O^{(1)}_{3,t}(x_1,x_2,\vec{x})(\vec{y} - f^{(0)}_t(\vec{x})), \tag{2.42}$$

$$\dot{O}^{(1)}_{3,t}(x_1,x_2,x_3) = O^{(1)}_{4,t}(x_1,x_2,x_3,\vec{x})(\vec{y} - f^{(0)}_t(\vec{x})), \tag{2.43}$$

$$O^{(1)}_{4,t}(x_1,x_2,x_3,x_4) = \nabla^T_\theta O^{(0)}_{3,0}(x_1,x_2,x_3)\nabla_\theta f^{(0)}_0(x_4), \tag{2.44}$$

where we have ignored the initial conditions for the time being. Integrating this system is straightforward:

$$f^{(0)}_t(\vec{x}) = \vec{y} + e^{-O^{(0)}_{2,0}(\vec{x},\vec{x})t}(f^{(0)}_0(\vec{x}) - \vec{y}), \tag{2.45}$$

For brevity, let us introduce the following definition:

$$\Delta f^{(0)}_t(x) = e^{-O^{(0)}_{2,0}(x,\vec{x})t}(f^{(0)}_0(\vec{x}) - \vec{y}). \tag{2.46}$$

This gives:

$$O^{(1)}_{3,t}(x_1,x_2,x_3) = O^{(1)}_{3,0}(x_1,x_2,x_3) - \int_0^t O^{(1)}_{4,0}(x_1,x_2,x_3,\vec{x})\Delta f^{(0)}_{t'}(\vec{x})\,dt'. \tag{2.47}$$

$$\begin{aligned} O^{(1)}_{2,t}(x_1,x_2) = O^{(1)}_{2,0}(x_1,x_2) &- \int_0^t O^{(1)}_{3,0}(x_1,x_2,\vec{x})\Delta f^{(0)}_{t'}(\vec{x})\,dt' \\ &+ \int_0^t \int_0^{t''} \Delta f^{(0),T}_{t''}(\vec{x})O^{(1)}_{4,0}(x_1,x_2,\vec{x},\vec{x}')\Delta f^{(0)}_{t'}(\vec{x}')\,dt'\,dt''. \end{aligned} \tag{2.48}$$

Let us elaborate the terms:

$$\int_0^t O^{(1)}_{3,0}(x_1,x_2,\vec{x})\Delta f^{(0)}_{t'}(\vec{x})\,dt' = O^{(1)}_{3,0}(x_1,x_2,\vec{x})\left(O^{(0)}_{2,0}(\vec{x},\vec{x})\right)^{-1}\left(I - e^{-O^{(0)}_{2,0}(\vec{x},\vec{x})t}\right)(f^{(0)}_0(\vec{x}) - \vec{y}). \tag{2.49}$$

$$\begin{aligned} &\int_0^t \int_0^{t''} \Delta f^{(0),T}_{t''}(\vec{x})O^{(1)}_{4,0}(x_1,x_2,\vec{x},\vec{x}')\Delta f^{(0)}_{t'}(\vec{x}')\,dt'\,dt'' \\ &= \int_0^t (f^{(0)}_0(\vec{x}) - \vec{y})^T\left(I - e^{-O^{(0)}_{2,0}(\vec{x},\vec{x})t'}\right)\left(O^{(0)}_{2,0}(\vec{x},\vec{x})\right)^{-1}O^{(1)}_{4,0}(x_1,x_2,\vec{x},\vec{x}')e^{-O^{(0)}_{2,0}(\vec{x},\vec{x})t'}(f^{(0)}_0(\vec{x}) - \vec{y})\,dt' \\ &= (f^{(0)}_0(\vec{x}) - \vec{y})^T\left(O^{(0)}_{2,0}(\vec{x},\vec{x})\right)^{-1}O^{(1)}_{4,0}(x_1,x_2,\vec{x},\vec{x}')\left(O^{(0)}_{2,0}(\vec{x},\vec{x})\right)^{-1}\left(I - e^{-O^{(0)}_{2,0}(\vec{x},\vec{x})t}\right)(f^{(0)}_0(\vec{x}) - \vec{y}) \\ &\quad - \int_0^t (f^{(0)}_0(\vec{x}) - \vec{y})^T e^{-O^{(0)}_{2,0}(\vec{x},\vec{x})t'}\left(O^{(0)}_{2,0}(\vec{x},\vec{x})\right)^{-1}O^{(1)}_{4,0}(x_1,x_2,\vec{x},\vec{x}')e^{-O^{(0)}_{2,0}(\vec{x},\vec{x})t'}(f^{(0)}_0(\vec{x}) - \vec{y})\,dt'. \end{aligned} \tag{2.50}$$

Consider the eigenvalue-eigenvector decomposition of $O_{2,0}^{(0)}(\vec{x},\vec{x})$: $O_{2,0}^{(0)}(\vec{x},\vec{x}) = \sum_{k=1}^{m} \lambda_1 v_k v_k^T$. This helps us integrating the last term:

$$\begin{aligned}
&\int_0^t (f_0^{(0)}(\vec{x}) - \vec{y})^T e^{-O_{2,0}^{(0)}(\vec{x},\vec{x})t'} \left(O_{2,0}^{(0)}(\vec{x},\vec{x})\right)^{-1} O_{4,0}^{(1)}(x_1,x_2,\vec{x},\vec{x}') e^{-O_{2,0}^{(0)}(\vec{x},\vec{x})t'} (f_0^{(0)}(\vec{x}) - \vec{y})\, dt' \\
&= \sum_{k,l=1}^{m} \int_0^t e^{-(\lambda_k+\lambda_l)t'} (f_0^{(0)}(\vec{x}) - \vec{y})^T v_k v_k^T \left(O_{2,0}^{(0)}(\vec{x},\vec{x})\right)^{-1} O_{4,0}^{(1)}(x_1,x_2,\vec{x},\vec{x}') v_l v_l^T (f_0^{(0)}(\vec{x}) - \vec{y})\, dt' \\
&= \sum_{k,l=1}^{m} \frac{1}{\lambda_k+\lambda_l} \left(1 - e^{-(\lambda_k+\lambda_l)t}\right) (f_0^{(0)}(\vec{x}) - \vec{y})^T v_k v_k^T \left(O_{2,0}^{(0)}(\vec{x},\vec{x})\right)^{-1} O_{4,0}^{(1)}(x_1,x_2,\vec{x},\vec{x}') v_l v_l^T (f_0^{(0)}(\vec{x}) - \vec{y}).
\end{aligned} \tag{2.51}$$

Recall $\hat{\Theta}_t(x_1,x_2) = O_{2,t}(x_1,x_2) = O_{2,t}^{(0)}(x_1,x_2) + n^{-1} O_{2,t}^{(1)}(x_1,x_2) + O(n^{-2})$. The first term (the limit NTK) does not depend on t, $O_{2,t}^{(0)}(x_1,x_2) = O_{2,0}^{(0)}(x_1,x_2) = \Theta(x_1,x_2)$, while the second one (the correction) does. Note that computing the second term invokes $O_{4,0}^{(1)}$, the fourth-order tensor, therefore approaching it directly requires $O(m^4)$ memory. Integrating the above system further gives the first-order correction for the limit model $f_t^{(1)}$.

As we shall see in Section 2.8, the kernel $\Theta^{NTH}(x_1,x_2) = O_{2,0}^{(0)}(x_1,x_2) + n^{-1}\mathbb{E}\, O_{2,\infty}^{(1)}(x_1,x_2)$ can be considered as a label-aware alternative to the usual NTK $\Theta(x_1,x_2) = O_{2,0}^{(0)}(x_1,x_2)$. Let us write its explicit definition and refer it later in Section 2.8:

$$\begin{aligned}
\Theta^{NTH}(x_1,x_2) &= O_{2,0}^{(0)}(x_1,x_2) + n^{-1}\mathbb{E}\, O_{2,\infty}^{(1)}(x_1,x_2) \\
&= \Theta(x_1,x_2) + n^{-1}\mathbb{E}\left[O_{2,0}^{(1)}(x_1,x_2)\right] - n^{-1}\mathbb{E}\left[O_{3,0}^{(1)}(x_1,x_2,\vec{x})\Theta^{-1}(\vec{x},\vec{x}) f_0^{(0)}(\vec{x})\right] \\
&\quad + n^{-1}\vec{y}^T\Theta^{-1}(\vec{x},\vec{x})\mathbb{E}\left[O_{4,0}^{(1)}(x_1,x_2,\vec{x},\vec{x})\right]\Theta^{-1}(\vec{x},\vec{x})\vec{y} \\
&\quad + n^{-1}\mathbb{E}\left[f_0^{(0),T}(\vec{x})\Theta^{-1}(\vec{x},\vec{x}) O_{4,0}^{(1)}(x_1,x_2,\vec{x},\vec{x})\Theta^{-1}(\vec{x},\vec{x}) f_0^{(0)}(\vec{x})\right] \\
&\quad - n^{-1}\sum_{k,l=1}^{m} \frac{1}{\lambda_k(\lambda_k+\lambda_l)} \vec{y}^T\vec{v}_k\vec{v}_k^T\mathbb{E}\left[O_{4,0}^{(1)}(x_1,x_2,\vec{x},\vec{x})\right]\vec{v}_l\vec{v}_l^T\vec{y} \\
&\quad - n^{-1}\sum_{k,l=1}^{m} \frac{1}{\lambda_k(\lambda_k+\lambda_l)} \mathbb{E}\left[f_0^{(0),T}(\vec{x})\vec{v}_k\vec{v}_k^T O_{4,0}^{(1)}(x_1,x_2,\vec{x},\vec{x})\vec{v}_l\vec{v}_l^T f_0^{(0)}(\vec{x})\right].
\end{aligned} \tag{2.52}$$

While the above result, Lemma 2.3.2, is valid under a conjecture, Conjecture 2.3.1, it can be proven rigorously under specific assumptions on data and activation functions, see Huang and Yau (2020). We decided to keep the original logic of Dyer and Gur-Ari (2020), i.e. Conjecture 2.3.1 $\Rightarrow$ Lemma 2.3.2, since it is more intuitive and easier to absorb to our opinion.

2.4 Computing the limit kernel

It is not obvious how to compute the limit kernel Θ predicted by the theorems discussed in Section 2.2. Fortunately, one can compute the limit kernel exactly for certain classes of models.

2.4.1 Fully-connected nets

Consider an L-layer fully-connected network under NTK parameterization:

$$f(x) = h_L(x), \quad h_l(x) = \frac{1}{\sqrt{n_{l-1}}} W_l x_{l-1}(x), \quad x_{l-1}(x) = \phi(h_{l-1}(x)), \quad x_0(x) = x, \tag{2.53}$$

where $W_l \in \mathbb{R}^{n_l \times n_{l-1}}$ $\forall l \in [L]$. For simplicity, we assume $n_L = 1$, i.e. the output is scalar.

Since we already know (see Section 2.2) that the kernel does not depend on t under NTK parameterization, we consider the case $t = 0$ only and omit the t-subscript. The empirical NTK is given by

$$\hat{\Theta}(x, x') = \nabla_\theta^T f(x;\theta) \nabla_\theta f(x';\theta) = \sum_{l=1}^{L} \operatorname{tr}\left(\nabla_{W_l}^T f(x; W_{1:L}) \nabla_{W_l} f(x; W_{1:L})\right). \tag{2.54}$$

By chain rule,

$$\nabla_{W_l} f(x) = \sum_{i=1}^{n_l} \partial_{h_l^i} f(x) \nabla_{W_l} h_l^i(x) = \frac{1}{\sqrt{n_{l-1}}} \sum_{i=1}^{n_l} \sum_{j=1}^{n_{l-1}} \partial_{h_l^i} f(x) E_{ij} x_{l-1}^j(x) = \frac{1}{\sqrt{n_{l-1}}} \nabla_{h_l} f(x) x_{l-1}^T(x). \tag{2.55}$$

Therefore,

$$\hat{\Theta}(x, x') = \sum_{l=1}^{L} \operatorname{tr}\left(\nabla_{W_l}^T f(x) \nabla_{W_l} f(x)\right) = \sum_{l=1}^{L} \frac{1}{n_{l-1}} \left(\nabla_{h_l}^T f(x') \nabla_{h_l} f(x)\right) \times \left(x_{l-1}^T(x) x_{l-1}(x')\right). \tag{2.56}$$

If x_{l-1} had iid components with zero mean, $\frac{1}{n_{l-1}} x_{l-1}^T(x) x_{l-1}(x')$ would be an empirical covariance estimated with n_{l-1} samples. In fact, when all weights are iid standard Gaussians, components of h_{l-1} become iid Gaussian with zero mean as $n_{1:l-2} \to \infty$ sequentially. Hence their images under elementwise maps ϕ are also iid.

Proof by induction. $h_1(x) = \frac{1}{\sqrt{n_0}} W_1 x$ has iid Gaussian components with zero mean and variance $q_1(x) = x^T x$. Suppose components of $h_{l-1}(x)$ become iid Gaussian with zero mean and $q_{l-1}(x)$ variance as $n_{1:l-2} \to \infty$ sequentially. Then $h_l(x) = \frac{1}{\sqrt{n_{l-1}}} W_l \phi(h_{l-1}(x))$ converges (in distribution) to a vector of Gaussians with zero mean and variance $q_l(x) = \mathbb{E}_{z \sim \mathcal{N}(0, q_{l-1}(x))} \phi^2(z)$ as $n_{1:l-1} \to \infty$ sequentially by the Central Limit Theorem (CLT).

One can easily generalize the above proof to any finite set of inputs. In particular, $[h_l^i(x), h_l^i(x')]^T$ converges to a Gaussian with zero mean and covariance $\Sigma_l(x, x') = \begin{pmatrix} q_l(x) & q_l(x, x') \\ q_l(x, x') & q_l(x') \end{pmatrix}$, where $q_l(x, x') = \mathbb{E}_{[z, z']^T \sim \mathcal{N}(0, \Sigma_{l-1}(x, x'))} \phi(z) \phi(z')$. Hence as $n_{1:l-2} \to \infty$ sequentially, $\frac{1}{n_{l-1}} x_{l-1}^T(x) x_{l-1}(x')$ converges to $q_l(x, x')$.

Let $g_l(x) = \sqrt{n_l} \nabla_{h_l} f(x)$. Since

$$\nabla_{h_l^j} f(x) = \sum_{i=1}^{n_{l+1}} \nabla_{h_{l+1}^i} f(x) \nabla_{h_l^j} h_{l+1}^i(x) = \frac{1}{\sqrt{n_l}} \sum_{i=1}^{n_{l+1}} \nabla_{h_{l+1}^i} f(x) W_{l+1}^{ij} \phi'(h_l^j(x)), \tag{2.57}$$

we have $g_l(x) = \frac{1}{\sqrt{n_{l+1}}} D_l(x) W_{l+1}^T g_{l+1}(x)$, where $D_l(x) = \operatorname{diag}(\phi'(h_l(x)))$.

There are two obstacles that prevent us from following the same lines for g_l as for h_l. First, g_{l+1} depends on D_{l+1} that depends on h_{l+1} that depends on W_{l+1}. Since W_{l+1} and g_{l+1} are dependent, we cannot guarantee that components of g_l become iid. Second, we know the distribution of h_l as all the layers from the input side become infinitely wide sequentially, while induction for g_l should be performed starting from the head. Nevertheless, it can be proven rigorously that ignoring these two obstacles still leads to a correct result (Yang, 2020a): $g_l(x)$ converges to a vector of iid Gaussians with zero mean and variance $\dot{q}_l(x) = \dot{q}_{l+1}(x)\mathbb{E}_{z\sim\mathcal{N}(0,q_l(x))}(\phi')^2(z)$ as $n_{1:L-1} \to \infty$. A similar result holds for a pair of inputs: $[g_l^i(x), g_l^i(x')]^T$ converges to a Gaussian with zero mean and covariance $\dot{\Sigma}_l(x,x') = \begin{pmatrix} \dot{q}_l(x) & \dot{q}_l(x,x') \\ \dot{q}_l(x,x') & \dot{q}_l(x') \end{pmatrix}$, where $\dot{q}_l(x,x') = \dot{q}_{l+1}(x,x')\mathbb{E}_{[z,z']^T\sim\mathcal{N}(0,\Sigma_l(x,x'))}\phi'(z)\phi'(z')$. Hence $\nabla_{h_l}^T f(x')\nabla_{h_l} f(x) = \frac{1}{n_l} g_l^T(x')g_l(x)$ converges to $\dot{q}_l(x,x')$.

Putting all together, $\hat{\Theta}(x,x')$ converges to $\Theta(x,x') = \sum_{l=1}^L \dot{q}_l(x,x')q_l(x,x')$, where

$$q_1(x,x') = x^T x', \quad q_l(x,x') = \mathbb{E}_{[z,z']^T\sim\mathcal{N}(0,\Sigma_{l-1}(x,x'))}\phi(z)\phi(z'), \tag{2.58}$$

$$\dot{q}_L(x,x') = 1, \quad \dot{q}_l(x,x') = \dot{q}_{l+1}(x,x')\mathbb{E}_{[z,z']^T\sim\mathcal{N}(0,\Sigma_l(x,x'))}\phi'(z)\phi'(z'), \tag{2.59}$$

and $\Sigma_l(x,x') = \begin{pmatrix} q_l(x,x) & q_l(x,x') \\ q_l(x,x') & q_l(x',x') \end{pmatrix}$. Note that the Master theorem of Yang (2020a) gives similar recurrent formulas for NTK of any architecture expressible by a tensor program and makes them mathematically rigorous.

In fact, computing the NTK can be performed in a convenient sequential layer-wise manner, as implemented in Neural Tangents[1] (Novak et al., 2020). Define the NTK for the first l layers as $\Theta_{:l}(x,x') = \sum_{l'=1}^l \operatorname{tr}(\nabla_{W_{l'}}^T h_l^i(x)\nabla_{W_{l'}} h_l^i(x'))$; in this case $\Theta_{:L}(x,x') = \Theta(x,x')$. Suppose $\Theta_{:l-1}(x,x')$ and $q_{l-1}(x,x')$ are already computed. Adding a nonlinearity and a linear layer with weights W_l gives q_l as listed above:

$$q_l(x,x') = \mathbb{E}_{[z,z']^T\sim\mathcal{N}(0,\Sigma_{l-1}(x,x'))}\phi(z)\phi(z'), \quad \text{where } \Sigma_{l-1}(x,x') = \begin{pmatrix} q_{l-1}(x,x) & q_{l-1}(x,x') \\ q_{l-1}(x,x') & q_{l-1}(x',x') \end{pmatrix}. \tag{2.60}$$

However, according to a formula above, $\dot{q}_l$ is computed using $\dot{q}_{l+1}$, which requires a sequential layer-wise "forward pass" to compute all q_l and a "backward pass" to compute $\dot{q}_l$. In fact, one forward pass is enough:

$$\begin{aligned}\Theta_{:l}(x,x') = \sum_{l'=1}^{l} \operatorname{tr}(\nabla_{W_{l'}}^T h_l^i(x)\nabla_{W_{l'}} h_l^i(x')) &= q_l(x,x') + \sum_{l'=1}^{l-1} \operatorname{tr}(\nabla_{W_{l'}}^T h_l^i(x)\nabla_{W_{l'}} h_l^i(x')) \\ &= q_l(x,x') + \Theta_{:l-1}(x,x')\mathbb{E}_{[z,z']^T\sim\mathcal{N}(0,\Sigma_{l-1}(x,x'))}\phi'(z)\phi'(z'). \end{aligned} \tag{2.61}$$

In Neural Tangents, each operation in a neural network is mapped to a corresponding kernel transform.

2.4.2 Convolutional nets

The same idea can be applied for convolutional nets as well. Consider 1d-convolutions for simplicity. In this case, we are dealing with 1d "images" with d pixels: $x \in \mathbb{R}^{n_0\times d}$. Consider a network with L

[1] https://github.com/google/neural-tangents

convolutions under NTK parameterization and an average pooling at the end:

$$f^i = \frac{1}{d}\sum_{s=1}^{d} x_L^{i,s}, \quad h_l^{i,s} = \frac{1}{\sqrt{n_{l-1}}}\sum_{j=1}^{n_{l-1}}\sum_{r\in \ker} W_l^{ijr} x_{l-1}^{j,s+r}, \quad x_{l-1}^{i,s} = \phi(h_{l-1}^{i,s}), \quad x_0^{i,s} = x^{i,s}, \tag{2.62}$$

where we omitted the argument x for brevity, $W_l \in \mathbb{R}^{n_l \times n_{l-1} \times |\ker|}$ with $W_l^{ijr} \sim \mathcal{N}(0,1)$ iid $\forall l \in [L]$, and ker denotes the convolution filter; e.g. $\ker = [-1,0,1]$ for a convolution of size 3. For simplicity, we assume $n_L = 1$, i.e. the output is scalar.

As before, the empirical NTK is given as

$$\hat{\Theta}(x,x') = \nabla_\theta^T f(x;\theta)\nabla_\theta f(x';\theta) = \sum_{l=1}^{L}\sum_{i=1}^{n_l}\sum_{j=1}^{n_{l-1}}\sum_{r\in\ker} \partial_{W_l^{ijr}} f(x)\partial_{W_l^{ijr}} f(x'). \tag{2.63}$$

By chain rule,

$$\partial_{W_l^{ijr}} f = \sum_{s=1}^{d} \partial_{h_l^{i,s}} f \partial_{W_l^{ijr}} h_l^{i,s} = \frac{1}{\sqrt{n_{l-1}}}\sum_{s=1}^{d} \partial_{h_l^{i,s}} f x_{l-1}^{j,s+r}. \tag{2.64}$$

Therefore,

$$\hat{\Theta}(x,x') = \sum_{l=1}^{L}\frac{1}{n_{l-1}}\sum_{i=1}^{n_l}\sum_{j=1}^{n_{l-1}}\sum_{r\in\ker}\sum_{s,s'=1}^{d} \partial_{h_l^{i,s}} f(x)\partial_{h_l^{i,s'}} f(x') x_{l-1}^{j,s+r}(x) x_{l-1}^{j,s'+r}(x'). \tag{2.65}$$

As for the fully-connected case, we are going to prove that $h^{i,s}$ become Gaussian with zero mean and variance given by a certain recurrent formula as $n_{1:l-1} \to \infty$ sequentially. However for the convolutional case, not all $h^{i,s}$ become independent: they become independent for different i's but not for different s.

Let us induct on l. $h_1^{i,s} = \frac{1}{\sqrt{n_0}}\sum_{j=1}^{n_0}\sum_{r\in\ker} W_1^{ijr} x^{j,s+r}$ are independent for any two different i's. For a fixed i, $h_1^{i,\cdot}$ is a Gaussian vector with zero mean and covariance $q_1^{s,s'} = \frac{1}{n_0}\sum_{j=1}^{n_0}\sum_{r\in\ker} x^{j,s+r}x^{j,s'+r}$. Suppose $h_{l-1}^{i,s}$ becomes Gaussian with zero mean, independent for any two different i's, and $q_{l-1}^{s,s'}$ is its covariance as $n_{1:l-2} \to \infty$ sequentially. Then $h_l^{i,s} = \frac{1}{\sqrt{n_{l-1}}}\sum_{j=1}^{n_{l-1}}\sum_{r\in\ker} W_l^{ijr} x_{l-1}^{j,s+r}$ converges (in distribution) to a random variable with similar properties but with covariance $q_l^{s,s'} = \mathbb{E}_{z\sim\mathcal{N}(0,q_{l-1})}\sum_{r\in\ker}\phi(z^{s+r})\phi(z^{s'+r})$ as $n_{1:l-1} \to \infty$ sequentially by the Central Limit Theorem (CLT).

One can easily generalize the above proof to any finite set of inputs. In particular, $[h_l^{i,\cdot}(x), h_l^{i,\cdot}(x')]^T \in \mathbb{R}^{2d}$ converges to a Gaussian with zero mean and covariance $\Sigma_l(x,x') = \begin{pmatrix} q_l(x) & q_l(x,x') \\ q_l(x,x') & q_l(x') \end{pmatrix} \in \mathbb{R}^{2d\times 2d}$, where $q_l^{s,s'}(x,x') = \mathbb{E}_{[z,z']^T\sim\mathcal{N}(0,\Sigma_{l-1}(x,x'))}\sum_{r\in\ker}\phi(z^{s+r})\phi(z'^{,s'+r})$. Hence as $n_{1:l-2} \to \infty$ sequentially, $\frac{1}{n_{l-1}}\sum_{j=1}^{n_{l-1}}\sum_{r\in\ker} x_{l-1}^{j,s+r}(x)x_{l-1}^{j,s'+r}(x')$ converges to $q_l^{s,s'}(x,x')$.

Let $g_l^{j,p} = \sqrt{n_l}\nabla_{h_l^{j,p}} f$. Since

$$\begin{aligned}\partial_{h_l^{j,p}} f &= \sum_{i=1}^{n_{l+1}}\sum_{s=1}^{d} \partial_{h_{l+1}^{i,s}} f \partial_{h_l^{j,p}} h_{l+1}^{i,s} \\ &= \frac{1}{\sqrt{n_l}}\sum_{i=1}^{n_{l+1}}\sum_{s=1}^{d}\partial_{h_{l+1}^{i,s}} f \sum_{r\in\ker} W_{l+1}^{ijr} 1_{s+r=p}\phi'(h_l^{j,p}) = \frac{1}{\sqrt{n_l}}\sum_{i=1}^{n_{l+1}}\sum_{r\in\ker}\partial_{h_{l+1}^{i,p-r}} f W_{l+1}^{ijr}\phi'(h_l^{j,p}),\end{aligned} \tag{2.66}$$

$\partial_{h_L^{j,p}} f = \frac{1}{d}\phi'(h_L^{j,p})$, and $n_L = 1$, we have

$$g_L^{j,p} = \frac{1}{d}\phi'(h_L^{j,p}), \quad g_l^{j,p} = \frac{1}{\sqrt{n_{l+1}}} \sum_{i=1}^{n_{l+1}} \sum_{r \in \ker} g_{l+1}^{i,p-r} W_{l+1}^{ijr} \phi'(h_l^{j,p}). \tag{2.67}$$

With the same correctness remark as for convolutional nets, it is possible to show that $g_l^{j,p}$ become independent for different j's and $g_l^{j,\cdot}$ become Gaussian with covariance $\dot{q}_l^{p,p'}$ as $n_{1:L-1} \to \infty$. Covariance is given by the following recurrence: $\dot{q}_L^{p,p'} = \frac{1}{d^2}\mathbb{E}_{[z,z']^T \sim \mathcal{N}(0,\Sigma_L)} \phi'(z^p)\phi'(z^{p'})$, $\dot{q}_l^{p,p'} = \mathbb{E}_{z \sim \mathcal{N}(0,q_l)} \phi'(z^p)\phi'(z^{p'}) \sum_{r \in \ker} \dot{q}_{l+1}^{p-r,p'-r}$.

A similar result holds for a pair of inputs: $[g_l^{i,\cdot}(x), g_l^{i,\cdot}(x')]^T \in \mathbb{R}^{2d}$ converges to a Gaussian with zero mean and covariance $\dot{\Sigma}_l(x,x') = \begin{pmatrix} \dot{q}_l(x) & \dot{q}_l(x,x') \\ \dot{q}_l(x,x') & \dot{q}_l(x') \end{pmatrix} \in \mathbb{R}^{2d \times 2d}$, where $\dot{q}_l^{s,s'}(x,x') = \mathbb{E}_{[z,z']^T \sim \mathcal{N}(0,\Sigma_l(x,x'))} \phi'(z^s)\phi'(z'^{,s'}) \sum_{r \in \ker} \dot{q}_{l+1}^{s-r,s'-r}(x,x')$. Hence

$$\sum_{i=1}^{n_l} \partial_{h_l^{i,s}} f(x) \partial_{h_l^{i,s'}} f(x') = \frac{1}{n_l} \sum_{i=1}^{n_l} g_l^{i,s}(x) g_l^{i,s'}(x') \to \dot{q}_l^{s,s'}(x,x'). \tag{2.68}$$

Putting all together, $\hat{\Theta}(x,x')$ converges to $\Theta(x,x') = \sum_{l=1}^{L} \sum_{s,s'=1}^{d} \dot{q}_l^{s,s'}(x,x') q_l^{s,s'}(x,x')$, where

$$q_1^{s,s'}(x,x') = \frac{1}{n_0} \sum_{j=1}^{n_0} \sum_{r \in \ker} x^{j,s+r} x'^{,j,s'+r}, \tag{2.69}$$

$$q_l^{s,s'}(x,x') = \mathbb{E}_{[z,z']^T \sim \mathcal{N}(0,\Sigma_{l-1}(x,x'))} \sum_{r \in \ker} \phi(z^{s+r})\phi(z'^{,s'+r}), \tag{2.70}$$

$$\dot{q}_L^{s,s'}(x,x') = \frac{1}{d^2}\mathbb{E}_{[z,z']^T \sim \mathcal{N}(0,\Sigma_L(x,x'))} \phi'(z^s)\phi'(z'^{,s'}), \tag{2.71}$$

$$\dot{q}_l^{s,s'}(x,x') = \mathbb{E}_{[z,z']^T \sim \mathcal{N}(0,\Sigma_l(x,x'))} \phi'(z^s)\phi'(z'^{,s'}) \sum_{r \in \ker} \dot{q}_{l+1}^{s-r,s'-r}(x,x'), \tag{2.72}$$

and $\Sigma_l(x,x') = \begin{pmatrix} q_l(x,x) & q_l(x,x') \\ q_l(x,x') & q_l(x',x') \end{pmatrix}$.

Same as for fully-connected nets, computing the NTK can be performed in a convenient sequential layer-wise manner. Define the empirical NTK for the first l layers as

$$\hat{\Theta}_{:l}^{s,s'}(x,x') = \sum_{l'=1}^{l} \sum_{i=1}^{n_{l'}} \sum_{j=1}^{n_{l'-1}} \sum_{r \in \ker} \partial_{W_{l'}^{ijr}} h_l^{1,s}(x) \partial_{W_{l'}^{ijr}} h_l^{1,s'}(x'); \tag{2.73}$$

in this case, by chain rule,

$$\begin{aligned}\hat{\Theta}(x,x') &= \sum_{l=1}^{L}\sum_{i=1}^{n_l}\sum_{j=1}^{n_{l-1}}\sum_{r\in\ker}\partial_{W_l^{ijr}}f(x)\partial_{W_l^{ijr}}f(x')\\ &= \sum_{l=1}^{L}\sum_{i=1}^{n_l}\sum_{j=1}^{n_{l-1}}\sum_{s,s'=1}^{d}\sum_{r\in\ker}\partial_{W_l^{ijr}}h_L^{1,s}(x)\partial_{W_l^{ijr}}h_L^{1,s'}(x')\partial_{h_L^{1,s}}f(x)\partial_{h_L^{1,s'}}f(x')\\ &= \frac{1}{d^2}\sum_{s,s'=1}^{d}\phi'(h_L^{1,s}(x))\phi'(h_L^{1,s'}(x'))\hat{\Theta}_{:L}^{s,s'}(x,x'), \quad (2.74)\end{aligned}$$

and therefore,

$$\Theta(x,x') = \frac{1}{d^2}\sum_{s,s'=1}^{d}\dot{q}_L^{s,s'}(x,x')\Theta_{:L}^{s,s'}(x,x'). \quad (2.75)$$

Suppose $\hat{\Theta}_{:l-1}(x,x')$ and $q_{l-1}(x,x')$ are already computed. Adding a nonlinearity and a convolutional layer with weights W_l gives q_l as listed above:

$$q_l^{s,s'}(x,x') = \mathbb{E}_{[z,z']\sim\mathcal{N}(0,\Sigma_{l-1}(x,x'))}\sum_{r\in\ker}\phi(z^{s+r})\phi(z'^{,s'+r}), \quad (2.76)$$

where $\Sigma_{l-1}(x,x') = \begin{pmatrix} q_{l-1}(x,x) & q_{l-1}(x,x') \\ q_{l-1}(x,x') & q_{l-1}(x',x')\end{pmatrix}$. We can compute $\hat{\Theta}_{:L}$ in a single forward pass using the following recurrence:

$$\begin{aligned}\hat{\Theta}_{:l}^{s,s'}(x,x') &= \sum_{l'=1}^{l}\sum_{i=1}^{n_{l'}}\sum_{j=1}^{n_{l'-1}}\sum_{\tilde{r}\in\ker}\partial_{W_{l'}^{ij\tilde{r}}}h_l^{1,s}(x)\partial_{W_{l'}^{ij\tilde{r}}}h_l^{1,s'}(x')\\ &= \frac{1}{n_{l-1}}\sum_{j=1}^{n_{l-1}}\sum_{\tilde{r}\in\ker}x^{j,s+\tilde{r}}(x)x^{j,s'+\tilde{r}}(x')\\ &+ \sum_{l'=1}^{l-1}\sum_{i=1}^{n_{l'}}\sum_{j=1}^{n_{l'-1}}\sum_{\tilde{r}\in\ker}\sum_{k,k'=1}^{n_{l-1}}\sum_{p,p'=1}^{d}\partial_{W_{l'}^{ij\tilde{r}}}h_{l-1}^{k,p}(x)\partial_{W_{l'}^{ij\tilde{r}}}h_{l-1}^{k',p'}(x')\partial_{h_{l-1}^{k,p}}h_l^{1,s}(x)\partial_{h_{l-1}^{k',p'}}h_l^{1,s'}(x')\\ &= \frac{1}{n_{l-1}}\sum_{j=1}^{n_{l-1}}\sum_{\tilde{r}\in\ker}x^{j,s+\tilde{r}}(x)x^{j,s'+\tilde{r}}(x')\\ &+ \frac{1}{n_{l-1}}\sum_{l'=1}^{l-1}\sum_{i=1}^{n_{l'}}\sum_{j=1}^{n_{l'-1}}\sum_{\tilde{r},r,r'\in\ker}\sum_{k,k'=1}^{n_{l-1}}\partial_{W_{l'}^{ij\tilde{r}}}h_{l-1}^{k,s+r}(x)\partial_{W_{l'}^{ij\tilde{r}}}h_{l-1}^{k',s'+r'}(x')\\ &\times W_l^{1kr}\phi'(h_{l-1}^{k,s+r}(x))W_l^{1k'r'}\phi'(h_{l-1}^{k',s'+r'}(x')). \quad (2.77)\end{aligned}$$

A limit then gives

$$\Theta_{:l}^{s,s'}(x,x') = q_l^{s,s'}(x,x') + \sum_{r,r'\in\ker}\Theta_{:l-1}^{s+r,s'+r'}(x,x')\mathbb{E}_{[z,z']^T\sim\mathcal{N}(0,\Sigma_{l-1}(x,x'))}\phi'(z^{s+r})\phi'(z'^{,s'+r'}), \quad (2.78)$$

which resembles the corresponding result for fully-connected nets when $\ker = [0]$.

2.4.3 Computing the expectations

The only obstacle that prevents explicit computation here is expectations over $[z, z']^T \sim \mathcal{N}(0, \Sigma_l(x, x'))$. Fortunately, these expectations can be computed analytically for certain ϕ: in particular, for ReLU and the error function.

We cover only the case of ReLU here as it is more widely used in practice. Let us omit the l-subscript and the arguments (x, x') for brevity: $\Sigma = \begin{pmatrix} q_{11} & q_{12} \\ q_{12} & q_{22} \end{pmatrix}$, and we are interested in $\mathbb{E}_{[u,v]^T \sim \mathcal{N}(0,\Sigma)} [u]_+ [v]_+$ and $\mathbb{E}_{[u,v]^T \sim \mathcal{N}(0,\Sigma)} 1_{u>0} 1_{v>0}$.

Following Arora et al. (2019b), we start with assuming $q_{11} = q_{22} = 1$ and $q_{12} = \lambda$; $\Sigma \geq 0$ implies $|\lambda| \leq 1$. Then

$$\begin{aligned}
\mathbb{E}_{[u,v]^T \sim \mathcal{N}(0,\Sigma)} [u]_+ [v]_+ &= \mathbb{E}_{[u,\tilde{v}]^T \sim \mathcal{N}(0,I)} [u]_+ \left[\lambda u + \sqrt{1-\lambda^2}\tilde{v}\right]_+ \\
&= \mathbb{E}_{u \sim \mathcal{N}(0,1)} \left([u]_+ \int_{-\frac{\lambda}{\sqrt{1-\lambda^2}} u}^{\infty} \left(\lambda u + \sqrt{1-\lambda^2}\tilde{v}\right) \frac{1}{\sqrt{2\pi}} e^{-\tilde{v}^2/2} \, d\tilde{v} \right) \\
&= \mathbb{E}_{u \sim \mathcal{N}(0,1)} \left([u]_+ \left(\lambda u \frac{1}{2} \left(1 - \operatorname{erf}\left(-\frac{\lambda}{\sqrt{2-2\lambda^2}} u \right) \right) + \sqrt{\frac{1-\lambda^2}{2\pi}} e^{-\frac{\lambda^2}{2-2\lambda^2} u^2} \right) \right) \\
&= \int_0^{\infty} u \left(\lambda u \frac{1}{2} \left(1 - \operatorname{erf}\left(-\frac{\lambda}{\sqrt{2-2\lambda^2}} u \right) \right) + \sqrt{\frac{1-\lambda^2}{2\pi}} e^{-\frac{\lambda^2}{2-2\lambda^2} u^2} \right) \frac{1}{\sqrt{2\pi}} e^{-u^2/2} \, du \\
&= \frac{\lambda}{4} + \int_0^{\infty} u \left(\lambda u \frac{1}{2} \operatorname{erf}\left(\frac{\lambda}{\sqrt{2-2\lambda^2}} u \right) + \sqrt{\frac{1-\lambda^2}{2\pi}} e^{-\frac{\lambda^2}{2-2\lambda^2} u^2} \right) \frac{1}{\sqrt{2\pi}} e^{-u^2/2} \, du \\
&= \frac{\lambda}{4} + \frac{\lambda}{2} A + \sqrt{\frac{1-\lambda^2}{2\pi}} B. \qquad (2.79)
\end{aligned}$$

$$\begin{aligned}
A &= \int_0^{\infty} u^2 \operatorname{erf}\left(\frac{\lambda}{\sqrt{2-2\lambda^2}} u \right) \frac{1}{\sqrt{2\pi}} e^{-u^2/2} \, du = -\int_0^{\infty} u \operatorname{erf}\left(\frac{\lambda}{\sqrt{2-2\lambda^2}} u \right) \frac{1}{\sqrt{2\pi}} \, d\left(e^{-u^2/2} \right) \\
&= \int_0^{\infty} \left(\operatorname{erf}\left(\frac{\lambda}{\sqrt{2-2\lambda^2}} u \right) + u \frac{\lambda}{\sqrt{2-2\lambda^2}} \frac{2}{\sqrt{\pi}} e^{-\frac{\lambda^2}{2-2\lambda^2} u^2} \right) \frac{1}{\sqrt{2\pi}} e^{-u^2/2} \, du = C + \frac{\lambda}{\sqrt{2-2\lambda^2}} \frac{2}{\sqrt{\pi}} B. \qquad (2.80)
\end{aligned}$$

$$C = \int_0^{\infty} \operatorname{erf}\left(\frac{\lambda}{\sqrt{2-2\lambda^2}} u \right) \frac{1}{\sqrt{2\pi}} e^{-u^2/2} \, du = \frac{1}{\pi} \arctan\left(\frac{\lambda}{\sqrt{1-\lambda^2}} \right) = \frac{1}{\pi} \arcsin \lambda. \qquad (2.81)$$

$$B = \int_0^{\infty} u e^{-\frac{\lambda^2}{2-2\lambda^2} u^2} \frac{1}{\sqrt{2\pi}} e^{-u^2/2} \, du = \frac{1}{\sqrt{2\pi}} \int_0^{\infty} u e^{-\frac{1}{2-2\lambda^2} u^2} \, du = \frac{1-\lambda^2}{\sqrt{2\pi}}. \qquad (2.82)$$

Putting all together,

$$\begin{aligned}\mathbb{E}_{[u,v]^T\sim\mathcal{N}(0,\Sigma)}[u]_+[v]_+ &= \frac{\lambda}{4}+\frac{\lambda}{2}A+\sqrt{\frac{1-\lambda^2}{2\pi}}B = \frac{\lambda}{4}+\frac{\lambda}{2}C+\frac{\lambda^2}{\sqrt{1-\lambda^2}}\frac{1}{\sqrt{2\pi}}B+\sqrt{\frac{1-\lambda^2}{2\pi}}B\\ &= \frac{\lambda}{4}+\frac{\lambda}{2}C+\frac{1}{\sqrt{1-\lambda^2}}\frac{1}{\sqrt{2\pi}}B = \frac{\lambda}{4}+\frac{\lambda}{2\pi}\arcsin\lambda+\frac{\sqrt{1-\lambda^2}}{2\pi}\\ &= \frac{\lambda\left(\frac{\pi}{2}+\arcsin\lambda\right)+\sqrt{1-\lambda^2}}{2\pi} = \frac{\lambda\left(\pi-\arccos\lambda\right)+\sqrt{1-\lambda^2}}{2\pi}. \quad (2.83)\end{aligned}$$

And for the second quantity,

$$\begin{aligned}\mathbb{E}_{[u,v]^T\sim\mathcal{N}(0,\Sigma)}1_{u>0}1_{v>0} &= \mathbb{E}_{[u,\tilde{v}]^T\sim\mathcal{N}(0,I)}1_{u>0}1_{\lambda u+\sqrt{1-\lambda^2}\tilde{v}>0}\\ &= \mathbb{E}_{u\sim\mathcal{N}(0,1)}\left(1_{u>0}\int_{-\frac{\lambda}{\sqrt{1-\lambda^2}}u}^{\infty}\frac{1}{\sqrt{2\pi}}e^{-\tilde{v}^2/2}\,d\tilde{v}\right)\\ &= \mathbb{E}_{u\sim\mathcal{N}(0,1)}\left(1_{u>0}\frac{1}{2}\left(1-\operatorname{erf}\left(-\frac{\lambda}{\sqrt{2-2\lambda^2}}u\right)\right)\right)\\ &= \int_0^{\infty}\frac{1}{2}\left(1-\operatorname{erf}\left(-\frac{\lambda}{\sqrt{2-2\lambda^2}}u\right)\right)\frac{1}{\sqrt{2\pi}}e^{-u^2/2}\,du\\ &= \frac{1}{4}+\int_0^{\infty}\frac{1}{2}\operatorname{erf}\left(\frac{\lambda}{\sqrt{2-2\lambda^2}}u\right)\frac{1}{\sqrt{2\pi}}e^{-u^2/2}\,du\\ &= \frac{1}{4}+\frac{1}{2}C = \frac{\frac{\pi}{2}+\arcsin\lambda}{2\pi} = \frac{\pi-\arccos\lambda}{2\pi}. \quad (2.84)\end{aligned}$$

A general positive semi-definite matrix Σ can be expressed as $\Sigma = D\Lambda D$, where $\Lambda = \begin{pmatrix}1 & \lambda\\ \lambda & 1\end{pmatrix}$, $D = \begin{pmatrix}\sqrt{q_{11}} & 0\\ 0 & \sqrt{q_{22}}\end{pmatrix}$, and $\lambda = \frac{q_{12}}{\sqrt{q_{11}q_{22}}}$. Then, using homogeneity of ReLU,

$$\begin{aligned}\mathbb{E}_{[u,v]^T\sim\mathcal{N}(0,\Sigma)}[u]_+[v]_+ &= \mathbb{E}_{[u,v]^T\sim\mathcal{N}(0,D\Lambda D)}[u]_+[v]_+ = \mathbb{E}_{[u,v]^T\sim\mathcal{N}(0,\Lambda)}[\sqrt{q_{11}}u]_+[\sqrt{q_{22}}v]_+\\ &= \sqrt{q_{11}q_{22}}\mathbb{E}_{[u,v]^T\sim\mathcal{N}(0,\Lambda)}[u]_+[v]_+ = \sqrt{q_{11}q_{22}}\frac{\lambda\left(\pi-\arccos\left(\frac{q_{12}}{\sqrt{q_{11}q_{22}}}\right)\right)+\sqrt{1-\frac{q_{12}^2}{q_{11}q_{22}}}}{2\pi}\\ &= \frac{\lambda\sqrt{q_{11}q_{22}}\left(\pi-\arccos\left(\frac{q_{12}}{\sqrt{q_{11}q_{22}}}\right)\right)+\sqrt{q_{11}q_{22}-q_{12}^2}}{2\pi}. \quad (2.85)\end{aligned}$$

$$\begin{aligned}\mathbb{E}_{[u,v]^T\sim\mathcal{N}(0,\Sigma)}1_{u>0}1_{v>0} &= \mathbb{E}_{[u,v]^T\sim\mathcal{N}(0,D\Lambda D)}1_{u>0}1_{v>0}\\ &= \mathbb{E}_{[u,v]^T\sim\mathcal{N}(0,\Lambda)}1_{u>0}1_{v>0} = \frac{\pi-\arccos\left(\frac{q_{12}}{\sqrt{q_{11}q_{22}}}\right)}{2\pi}. \quad (2.86)\end{aligned}$$

Similar explicit computations are available for convolutional networks (Arora et al., 2019b), as well as for generic tensor programs, as long as the nonlinearities used belong to a certain list (which

includes e.g. ReLU and the error function, see Novak et al. (2020) for a concrete implementation and Yang (2020a) for generic recurrent formulas in terms of expectations).

However, a typical convolutional network also uses max poolings and other nonlinear maps for which explicit formulas for expectations are not available at the moment. In this case, one can rely on a finite-width Monte-Carlo estimate for $\Theta(x, x')$, i.e. $\hat{\Theta}^{(M)}(x, x') = \frac{1}{M}\sum_{k=1}^{M}\hat{\Theta}(x, x')$, where M is a number of independent initializations and $\hat{\Theta}(x, x')$ is an empirical kernel for width n. According to convergence results, $\hat{\Theta}^{(M)}(x, x') \to \Theta(x, x')$ as $n \to \infty\ \forall M \geq 1$. Also, $\hat{\Theta}^{(M)}(x, x') \to \mathbb{E}\,\hat{\Theta}(x, x')$ as $M \to \infty\ \forall n \to \infty$. Unfortunately, one cannot guarantee that $\mathbb{E}\,\hat{\Theta}(x, x') = \Theta(x, x')$; therefore, $\hat{\Theta}^{(M)}(x, x')$ can be a biased estimate. However, according to experiments of Novak et al. (2020), discrepancy between $\hat{\Theta}^{(M)}$ and Θ decreases as M grows for any finite n. This means that the main component of this discrepancy is not bias but variance decreased by adding more Monte-Carlo samples.

We also have to note that Arora et al. (2019b) reports significant accuracy drops on a CNN of width $n = 512$ when using a single-sample Monte-Carlo estimate for the NTK instead of the exact limit NTK. However, they haven't provided any results for $M > 1$, therefore, this accuracy drop could be caused by large variance of $\hat{\Theta}$.

2.4.4 NTK for attention layers

A neural tangent kernel is typically considered for architectures for which analytical computation is available, i.e. for fully-connected and convolutional ReLU nets, see Section 2.4. One of the necessary conditions for exact computations to be possible is the fact that the output of each individual pre-activation neuron becomes a Gaussian process in the limit of large width. This allows one to apply Master theorem (Theorem 2.2.2), and express the NTK as an expectation over certain Gaussian variables.

However, there exist layers which does not enjoy Gaussian behavior even in the limit of large width. Attention layer is one of the examples:

$$f(x) = \text{Softmax}\left(G(x)\right)V(x), \qquad G(x) = \frac{1}{\sqrt{n}}Q^T(x)K(x), \tag{2.87}$$

where we define queries $Q(x) = xW_Q$, keys $K(x) = xW_K$, and values $V(x) = xW_V$. Dimensions of the corresponding matrices are: $W_Q \in \mathbb{R}^{n_0 \times n}$, $W_K \in \mathbb{R}^{n_0 \times n}$, and $W_V \in \mathbb{R}^{n_0 \times n_H}$, and $x \in \mathbb{R}^{d \times n_0}$.

If W_Q and W_K are independent with iid zero mean unit variance entries then $G_{\alpha\beta}(x) = n^{-1/2}\sum_{i=1}^{n}\sum_{j,k=1}^{n_0} x_{\alpha,j}x_{\beta,k}W_Q^{ji}W_K^{ki}$ converges by CLT to a Gaussian variable. The resulting limit matrix is therefore $d \times d$ matrix with (non-degenerate) Gaussian entries. Since d stays fixed as $n \to \infty$, we cannot apply any limit theorem to reason about the distribution of $f_i(x)$ for some $i \in [n_H]$.

Hron et al. (2020) consider a multi-head attention layer and show that it does enjoy Gaussian process behavior as width and number of heads go to infinity simultaneously:

$$f(x) = [f^1(x), \ldots, f^n(x)]W_O, \qquad f_i(x) = \text{Softmax}\left(G_i(x)\right)V_i(x), \qquad G_i(x) = \frac{1}{\sqrt{n}}Q_i^T(x)K_i(x), \tag{2.88}$$

where $W_O \in \mathbb{R}^{n_H n \times n_H}$ and all Q_i, K_i, and V_i are iid for different $i \in [n]$. To gain some intuition about the result of Hron et al. (2020), consider $n_H = 1$, i.e. outputs of all individual heads are scalars and the final output is also a scalar. In this case, $f(x)$ is a product of a vector with n iid

entries and a matrix with iid $\mathcal{N}(0, n^{-1})$ entries. This product tends to a Gaussian as $n \to \infty$ by CLT. Considering a set of inputs gives a random Gaussian vector similar to the fully-connected case, see Section 2.4.1.

Hron et al. (2020) gives exact formulas for covariances $q(x, x')$ and the kernel $\Theta(x, x')$; they are implemented as layers in NeuralTangents (Novak et al., 2020).

2.5 Computational aspects

2.5.1 Inference optimizations

Suppose one is able to compute (or approximate) the limit kernel, $\Theta(x, x')$, on any pair of points (x, x'). The result of kernel regression at convergence ($t \to \infty$) in the limit of inifinite width is then given by (see Eq. (2.9)):

$$f_\infty(x) = f_0(x) - \Theta(x, \vec{x})\Theta^{-1}(\vec{x}, \vec{x})(f_0(\vec{x}) - \vec{y}). \tag{2.89}$$

where $\Theta(\vec{x}, \vec{x}) \in \mathbb{R}^{m\times m}$ and $\Theta(x, \vec{x}) \in \mathbb{R}^{1\times m}$. For multi-class problems, $f(x) \in \mathbb{R}^k$, where k is the number of classes, and the kernel evaluated at two points becomes a $k \times k$ matrix:

$$\hat{\Theta}_{jj'}(x, x') = \nabla_\theta^T f^j(x) \nabla_\theta f^{j'}(x'). \tag{2.90}$$

Define a Gram matrix as $\hat{\Theta}_{ik+j,i'k+j'}(\vec{x}, \vec{x}) = \hat{\Theta}_{jj'}(x_i, x_{i'})$ and its limit counterpart $\Theta(\vec{x}, \vec{x}) \in \mathbb{R}^{mk\times mk}$ accordingly; similarly for $\Theta(x, \vec{x}) \in \mathbb{R}^{k\times mk}$. If one defines $f_0^{ik+j}(\vec{x}) = f_0^j(x_i)$, the corresponding solution takes the same form as Eq. (2.89).

Evaluating this quantity naively requires storing and inverting the kernel Gram matrix $\Theta(\vec{x}, \vec{x}) \in \mathbb{R}^{mk\times mk}$. Storing it requires $O(m^2k^2)$ memory, while inverting it takes $O(m^3k^3)$ time, making such a naive approach computationally infeasible for datasets with $mk \gtrsim 10^4$ (nevertheless, for small datasets, the naive approach for computing the NTK estimator (2.89) is feasible and may provide advantage over traditional SGD training, see Arora et al. (2020)).

Let us start with discussing two important optimizations implemented in Neural Tangents (Novak et al., 2020). Note that as discussed in Section 2.4, for a fully-connected net (and, in fact, for any tensor program, see Yang (2019b)) preactivations of different neurons on a given layer become iid as width goes to infinity. This implies $\Theta_{jj'}(x, x') = \Theta_{11}(x, x')1_{j=j'}$. Therefore the kernel Gram matrix has a block structure: $\Theta(\vec{x}, \vec{x}) = \Theta|_{k=1}(\vec{x}, \vec{x}) \otimes I_{k\times k}$. This reduces memory footprint to $O(m^2)$ and the time requirement to $O(m^3)$.

The second optimization deals with convolutional networks. Note that computing $\Theta(x, x')$ requires computing all intermediate covariances $q_l(x, x')$. These covariances were scalars for fully-connected nets since different neurons of a given layer became iid as width went to infinity. However, for an image with d pixels, different pixels of a given layer are dependent since their preactivations are computed using same weight matrices. That's why for convolutional nets, one has to construct intermediate covariance matrices of size $d \times d$; storing and computing them for each pair of points requires $O(m^2d^2)$ memory and time, even surpassing the time required for Gram matrix inversion when $d^2 > m$ (this happens e.g. for CIFAR10 for which $d = 32 \times 32 = 1024$, $m = 50000$, $k = 10$). However, as was noted e.g. in Xiao et al. (2018), if no pooling is used in the network, it suffices to compute and store d independent $m \times m$ blocks of this covariance matrix, boiling down to $O(m^2d)$ time requirement which is usually not greater than $O(m^3)$ time required for inversion.

So far, the main computational bottleneck was the time required for inverting the kernel Gram matrix. This problem is not specific for NTK; it appears for any regularized kernel regression problem:

$$\hat{f}_\lambda = \arg\min_{f \in \mathcal{H}} \sum_{j=1}^{m} \ell(y_j, f(x_j)) + \lambda \|f\|_{\mathcal{H}}^2. \tag{2.91}$$

Here $\mathcal{H}$ is a Hilbert space of functions of the form $f(x) = \Phi^T(x)\theta$; the corresponding scalar product is $\langle \Phi^T(x)\theta, \Phi^T(x)\theta' \rangle = \theta^T\theta'$. Hence $\|f\|_{\mathcal{H}}^2 = \langle f, f \rangle = \|\theta\|_2^2$ for $f(x) = \Phi^T(x)\theta$.

Problem (2.91) has an associated kernel, which we denote with the same letter as NTK: $\Theta(x, x') = \Phi^T(x)\Phi(x')$. Due to the representer theorem (Kimeldorf and Wahba, 1970), any solution of Problem (2.91) has the form $f(x) = \sum_{j=1}^m \alpha_j \Theta(x, x_j)$.

For now, consider quadratic loss: $\ell(y, z) = \frac{1}{2}\|y - z\|_2^2$. The problem above becomes:

$$\vec{\alpha} = \arg\min_{\vec{\alpha} \in \mathbb{R}^m} \frac{1}{2} \sum_{j=1}^{m} \left(\sum_{j'=1}^{m} \alpha_{j'} \Theta(x_j, x_{j'}) - y_j \right)^2 + \lambda \left\| \sum_{j=1}^{m} \alpha_j \Phi(x_j) \right\|_2^2. \tag{2.92}$$

This problem is convex, therefore any critical point of the corresponding functional is a solution:

$$(\Theta(\vec{x}, \vec{x}) + \lambda I)\vec{\alpha} = \vec{y}. \tag{2.93}$$

As long as $\Theta(\vec{x}, \vec{x}) + \lambda I$ is invertible, the solution is $\vec{\alpha} = (\Theta(\vec{x}, \vec{x}) + \lambda I)^{-1}\vec{y}$. Putting $\lambda = 0$, we recover expected Eq.(2.89) (since $\mathbb{E}\, f_0(x) = 0$).

While the represeneter theorem guarantees that it suffices to look for solutions only of the form $f(x) = \sum_{j=1}^m \alpha_j \Theta(x, x_j)$ instead of inspecting the whole $\mathcal{H}$, we, following Meanti et al. (2020), consider further contracting the search space by sampling m' points $(\tilde{x}_1, \ldots, \tilde{x}_{m'})$ uniformly out of m and looking for solutions of the form $f(x) = \sum_{j=1}^{m'} \tilde{\alpha}_j \Theta(x, \tilde{x}_j)$. This is known as Nyström approximation. The minimization problem then becomes:

$$\vec{\tilde{\alpha}} = \arg\min_{\vec{\tilde{\alpha}} \in \mathbb{R}^{m'}} \frac{1}{2} \sum_{j=1}^{m} \left(\sum_{j'=1}^{m'} \tilde{\alpha}_{j'} \Theta(x_j, \tilde{x}_{j'}) - y_j \right)^2 + \lambda \left\| \sum_{j=1}^{m'} \tilde{\alpha}_j \Phi(\tilde{x}_j) \right\|_2^2. \tag{2.94}$$

This problem is again convex and its critical points satisfy the following:

$$\left(\Theta\left(\vec{\tilde{x}}, \vec{x}\right) \Theta\left(\vec{x}, \vec{\tilde{x}}\right) + \lambda \Theta\left(\vec{\tilde{x}}, \vec{\tilde{x}}\right) \right) \vec{\tilde{\alpha}} = \Theta\left(\vec{\tilde{x}}, \vec{x}\right) \vec{y}. \tag{2.95}$$

Computing the kernel-kernel product takes $O(mm'^2)$ time and solving the above system directly takes $O(m'^3)$ time. The space requirement can be put to $O(m'^2)$ as the "rectangular Gram matrix" can be computed in $m' \times m'$ blocks.

Conjugate gradient methods are iterative methods designed for approximately solving linear systems of the form $A\vec{z} = \vec{b}$ without explicitly inverting the matrix A. The main operation used by these methods on each iteration is a matrix-vector product. In our case, the matrix-vector product requires $O(mm' + m'^2)$ time; note that it allows one to avoid computing the kernel-kernel product explicitly, by computing two matrix-vector product instead, costing $O(mm')$ time each.

Putting all together, solving system (2.95) with s iterations of a conjugate gradient method requires $O(s(mm' + m'^2))$ time and $O(m'^2)$ space. Based on certain theoretical results, Meanti et al.

(2020) suggest taking $m' = O(\sqrt{m})$ and $s = O(\log m)$. The resulting $O(m\sqrt{m}\log m)$ time and $O(m)$ space allows for applying their method to datasets of size up to $m \sim 10^6$ (the size of ImageNet). Meanti et al. (2020) also discuss several optimizations aiming for improving GPU-efficiency of the method. While their method is publicly available as an open-source library[2], we are not aware of any of its applications to NTK.

2.5.2 Computing the empirical kernel

All the previous discussion of the current section assumed that the kernel, Θ, can be efficiently computed. This is the case for certain models for which analytic computations are available. Indeed, for L-layer fully-connected nets, the limit Gram matrix $\Theta(\vec{x}, \vec{x})$ can be computed in $O(m^2 L)$ time while storing it requires $O(m^2)$ space, see Eqs. (2.60) and (2.61). For more complex models, e.g. for those including max-poolings, closed-form analytic expressions for the limit kernel are not currently available. However, the empirical kernel, $\hat{\Theta}$, can always be computed explicitly and is close to Θ for sufficiently large width (see convergence theorems in Section 2.2). For this reason, we are looking for ways to compute $\hat{\Theta}$ efficiently.

In order to simplify the illustration, we will discuss only time requirements in the sequel. Recall the empirical kernel is a product of two jacobians: $\hat{\Theta}_{jj'}(x, x') = \nabla_\theta^T f^j(x) \nabla_\theta f^{j'}(x')$. Therefore the time cost for computing the kernel consists of the time required to compute the jacobian and the time required for jacobian contraction.

Denote $[FP]$ the cost of a single forward pass for our network; a single backward pass has approximately the same cost. Then computing a jacobian for a given point x takes $O(k[FP])$ time. Contracting two jacobians for fixed j and j' takes $O(N)$ time, where N is the total number of parameters: $\theta \in \mathbb{R}^N$. Putting all together, computing the full $mk \times mk$ Gram matrix takes $O(mk[FP] + m^2k^2N)$ time.

Novak et al. (2021) propose a method for computing the NTK-vector product. It can be directly embedded into the method of Meanti et al. (2020) using conjugate gradients, or used for computing the kernel explicitly by applying it to columns of the $k \times k$ identify matrix.

Their method boils down to casting a matrix-vector product where the matrix is the empirical NTK to a vector-jacobian product followed by a jacobian-vector product: $\sum_{j'=1}^k \hat{\Theta}_{jj'}(x, x') v_{j'} = \nabla_\theta^T f^j(x) \sum_{j'=1}^k \nabla_\theta f^{j'}(x') v_{j'}$. Both matrix-vector products can be computed in $O([FP])$ time. Therefore this method allows to compute the full $mk \times mk$ Gram matrix in $O(m^2 k[FP])$ time, which improves over the jacobian contraction method as long as $[FP] < CkN$ for a certain constant C. Memory requirements that we do not show here are, in fact, same for both methods, see Novak et al. (2021).

Novak et al. (2021) also propose another optimization exploiting certain stucture of the function f: e.g. weights of a fully-connected net are aligned sequentially, while weights of a convolutional layer are aranged in blocks. We do not discuss it in the present survey. Both optimizations are publicly available as JAX (Bradbury et al., 2018) function transformations.[3].

[2] https://github.com/FalkonML/falkon
[3] https://github.com/iclr2022anon/fast_finite_width_ntk

2.6 Applications

2.6.1 A kernel method

Supervised learning on small datasets

The NTK is a kernel, therefore it can be used in any kernel method itself, i.e. kernel ridge regression or kernel SVM. However, computing the kernel Gram matrix on a dataset of size m requires $O(m^2)$ time, which is infeasible for large datasets. One can either rely on certain approximations, e.g. Nyström approximation, see Section 2.5, or restrict oneself to small datasets.

One possible advantage of kernel methods over neural nets is lower variance. Indeed, the only variance of a kernel method is induced by sampling the dataset, while a neural network has several more sources of variance; e.g. initialization randomness and batch sampling. It is likely that this difference in variances is especially important when the dataset is small.

The other advantage of kernel methods is having smaller number of hyperparamaters compared to neural nets. This makes kernel methods useful as robust baseline methods that may outperform large neural nets in a situation when there is no budget for careful hyperparamater tuning. As an illustration, Arora et al. (2020) demonstrated that kernel regression with 14-layer CNTK consistently outperforms ResNet-34 trained with standard hyperparameters on a random subset of CIFAR-10 with ≤ 640 samples.

Neural architecture search using NTK condition number

There are other setups where computing the Gram matrix on a small dataset is sufficient. For example, Chen et al. (2021) proposes a condition number of the NTK Gram matrix as a proxy-measure of a given architecture performance; this proxy-measure is then used to guide neural architecture search (NAS). In this case, we do not need the Gram matrix itself but only the condition number, which motivates computing the matrix on a small subset of examples. While the condition number on a random subset Gram matrix provides only a random estimate, possibly noisy and biased, of a true condition number, the way we use it does not require exact estimates. Indeed, a performance measure in NAS algorithms is mainly used to cut-off pathologic, low-performing models from a population, rather than finding the best one. Therefore any measure that correlates positively with performance suffices.

The use of condition number as a proxy-measure of performance relies on two hypotheses: (1) performance correlates with trainability, and (2) trainability correlates with NTK condition number. The first hypothesis is mainly motivated by a natural implication "bad trainability implies low performance". To motivate the second hypothesis, let us consider kernel ridge regression trained with usual discrete-time gradient descent:

$$f_{t+1}(\vec{x}) = f_t(\vec{x}) + \eta\Theta(\vec{x}, \vec{x})(\vec{y} - f_t(\vec{x})), \tag{2.96}$$

where now t is a discrete time-step and η is a learning rate.

Consider eigenvalue decomposition of the kernel: $\Theta(\vec{x}, \vec{x}) = \sum_{k=1}^{m} \lambda_k \vec{v}_k \vec{v}_k^T$, where $\lambda_1 \geq \ldots \geq \lambda_m \geq 0$, and $(\vec{v}_k)_{k=1}^m$ forms an orthonormal basis. Let us decompose our model's predictions as $f_t(\vec{x}) = \sum_{k=1}^{m} u_{t,k}\vec{v}_k$. Then the dynamics above decomposes as

$$u_{t+1,k} = u_{t,k} + \eta\lambda_k(\vec{y}^T\vec{v}_k - u_{t,k}). \tag{2.97}$$

This gives

$$u_{t+1,k} - \vec{y}^T \vec{v}_k = (1 - \eta\lambda_k)(u_{t,k} - \vec{y}^T \vec{v}_k), \tag{2.98}$$

and the solution is therefore

$$u_{t,k} = \vec{y}^T \vec{v}_k + (1 - \eta\lambda_k)^t (u_{0,k} - \vec{y}^T \vec{v}_k). \tag{2.99}$$

The dynamics above converges as $t \to \infty$ for any $u_{0,k}$ if and only if $\eta < 2/\lambda_k$. Since this should hold for all $k \in [m]$ and the maximal λ is λ_1, we need to have $\eta < 2/\lambda_1$. Therefore the m-th principal component converges at rate $\eta\lambda_m < 2\lambda_m/\lambda_1$. $\kappa = \lambda_m/\lambda_1$ is our condition number. We see that small condition number implies low trainability and thus, by the first hypothesis, low performance.

Using a combination of two proxy-measures, the condition number and the number of linear regions (we do not discuss it here), Chen et al. (2021) constructed a NAS method that provided state-of-the-art performance on NAS-Bench-201 (Dong and Yang, 2020), while using much smaller time compared to most of the other methods. Chen et al. (2021) tested their method on CIFAR10 and ImageNet as well. In both cases, their method demonstrated competetive performance while using orders of magnitude less time.

Matrix completion and image impainting

In some cases, posing the problem as kernel regression allows for certain optimizations. In particular, Radhakrishnan et al. (2022) proposed approaching the problem of matrix completion by minimizing the following loss:

$$\mathcal{L}(\theta) = \sum_{(i,j)\in S} (Y_{ij} - \operatorname{tr}(f(Z;\theta)M^{(ij)}))^2, \tag{2.100}$$

where $S \subset [k] \times [d]$ is a set of coordinates of known entries of the target matrix $Y \in \mathbb{R}^{k\times d}$, $M^{(ij)} \in \mathbb{R}^{k\times d}$ has 1 at position (i,j) and 0 elsewhere, $f(\cdot;\theta)$ is a neural network with parameters θ, n_0 inputs and k outputs, and $Z \in \mathbb{R}^{n_0\times d}$ is an a-priori given matrix. The model f is applied to each column of Z seperately, therefore $f(Z;\theta)$ is $k \times d$ matrix.

The above setup can be treated as a usual l_2 regression problem on a dataset $(Y_{ij}, M^{(ij)})_{(i,j)\in S}$. The corresponding empirical NTK is defined as $\hat{K}(M^{(ij)}, M^{(i'j')}) = \nabla_\theta^T \operatorname{tr}(f(Z;\theta)M^{(ij)}) \nabla_\theta \operatorname{tr}(f(Z;\theta)M^{(i'j')})$. Naturally, it does not depend on target matrix entries Y, and since there is only a finite set of possible inputs $M^{(ij)}$ (namely, kd), the resulting $kd \times kd$ Gram matrix will be the same for all possible matrix completion problems of a given target matrix dimensions. In other words, one can precompute the Gram matrix once and use it to all possible matrix completion problems of given dimensions. In contrast, original neural network formulation would require training a new network for each dataset $(Y_{ij}, M^{(ij)})_{(i,j)\in S}$.

When $f(\cdot;\theta)$ is given by a fully-connected network with L layers, Radhakrishnan et al. (2022) provide a closed-form formula for its limit NTK: $K(M^{(ij)}, M^{(i'j')}) = \kappa_L\left(z_{\cdot,j}^T z_{\cdot,j'}\right) 1_{i=i'}$, where κ_L is given by a certain recurrent relation. As we see, according to this kernel, elements of different rows of Y are orthogonal (does not effect each other), while similarity of elements of the same row is given by a scalar product of the corresponding columns of Z. Therefore columns of Z encodes a-priori similarities between columns of Y.

The matrix Z is called a feature-prior matrix. The ideal feature-prior matrix would be the target matrix Y itself. Since one does not have access to it, Radhakrishnan et al. (2022) suggest using the output $\hat{Y}$ of a separate matrix completion method instead. The resulting joint method performs better than the backbone one on popular collaborative filtering and virtual drug screening datasets.

Image impainting can be viewed as a special case of matrix completion. Apart from using the same Gram matrix for all problems of a given size, image impainting with convolutional networks allows for one more optimization.

When f is a convolutional network, we pose the problem a bit differently to above. Suppose f has n_0 input channels, 1 output channel, and it maps an image to an image of the same size. Suppose $Z \in \mathbb{R}^{n_0 \times 2^p \times 2^q}$ and it is treated as a $2^p \times 2^q$ image with n_0 channels. This in contrast to the previous considerations, where Z was a matrix with columns treated as different inputs to a vector-valued model. Similar to the above, $Y \in \mathbb{R}^{2^p \times 2^q}$ is a target image, and $M^{(ij)}$ of the same size has 1 at (i, j) and zero elsewhere.

Note that f applied to the "image" Z has $2^p \times 2^q$ output and therefore its NTK Θ is a $2^p \times 2^q \times 2^p \times 2^q$ tensor. Suppose f has no downsampling or upsampling layers. Radhakrishnan et al. (2022) provides exact formula for the corresponding limit NTK in terms of the limit NTK of the model f in this case: $K(M^{(ij)}, M^{(i'j')}) = \Theta(Z, Z)_{i,j,i',j'}$.

Now suppose f has s downsampling and s upsampling layers. Computing the Gram matrix for its NTK requires $O(2^{2p+2q})$ memory and $O(L2^{2p+2q})$ time, where L is the number of convolutions in f. It is already prohibitive for moderate-size images, i.e. when $p, q \approx 10$. Radhakrishnan et al. (2022) propose a way to reconstruct the $2^p \times 2^q \times 2^p \times 2^q$ Gram matrix from a smaller Gram matrix of size 2^{2s+p+q}. Moreover, this smaller Gram matrix requires computing the "usual" Gram matrices only for images of size $2^{s+1} \times 2^{s+1}$ which requires only $O(L2^{4s})$ time.

Approximate integration with application to federated learning

Even in the case when the NTK Gram matrix can be computed and stored, the exact solution (2.89) requires inverting the kernel Gram matrix, which costs $O(m^3)$ when performed naively. Fortunately, mixing continuous-time and discrete-time formulations allows one to avoid computing the inverse explicitly.

Denote $H_{t,ij} = \hat{\Theta}_t(x_i, x_j)$, $Z_{t,ik} = \partial_{\theta_i} f(x_k; \theta)$, and $u_{t,k} = f_t(x_k)$. Note that $H_t = Z_t^T Z_t$. Discrete-time weight evolution with learning rate η is given by

$$\theta_{t+1} = \theta_t + \eta Z_t(\vec{y} - \vec{u}_t). \tag{2.101}$$

Recall that assuming stationary kernel $H_t = H_0$ is equivalent to assuming stationary jacobian $Z_t = Z_0$. With this assumption, the dynamics above is solved as

$$\theta_t = \theta_0 + \eta Z_0 \sum_{s=0}^{t-1} (\vec{y} - \vec{u}_s). \tag{2.102}$$

Recall that integrating continuous-time gradient descent dynamics under assumption $H_t = H_0$ gives

$$\vec{u}_s = \vec{y} + e^{-\eta s H_0}(\vec{u}_0 - \vec{y}). \tag{2.103}$$

Combining the two latter equations, we get the weights at any time-step t:

$$\theta_t = \theta_0 + \eta Z_0 \sum_{s=0}^{t-1} e^{-\eta s H_0}(\vec{y} - \vec{u}_0). \tag{2.104}$$

The continuous analogue of the above evolution is obtained by replacing the sum with an integral:

$$\theta_t = \theta_0 + \eta Z_0 \int_0^t e^{-\eta s H_0}(\vec{y} - \vec{u}_0)\, ds = \theta_0 + Z_0 H_0^{-1} \left(I - e^{-\eta s H_0}\right)(\vec{y} - \vec{u}_0). \tag{2.105}$$

(a) Ground truth (b) No mapping (c) Basic (d) Positional enc. (e) Gaussian

Figure 2.1: Images are borrowed from Tancik et al. (2020).

Here we get the inverse, as expected.

Note that in this approach we do not assume that the network to be infinitely wide, we just assume it to be linear in its weights. This allows us to reason in terms of the network weight vector θ_t instead of reasoning in terms of some abstract feature space associated to the kernel. This aspect gives us one additional advantage: we can integrate the dynamics up to some time t_1 and, since we know the weights θ_{t_1}, compute Z_{t_1} and H_{t_1}. We can then proceed integration with these updated matrices. This method lies in between the usual gradient descent training and kernel gradient descent with constant kernel. The latter never updates the kernel, while the former updates the kernel at each timestep. In contrast, the method we discuss updates the kernel only at given timesteps.

The approach under discussion requires computing and storing Z of size $N \times m$, which is an obvious disadvantage. As a remedy, Yue et al. (2022) propose splitting the job of computing Z between several workers. A server joins the parts together, integrates the dynamics up to some timestep t, and sends θ_t to all of the workers, starting a new iteration. Tuning the timesteps of kernel updates may help balancing load between the server and the workers. The data used to compute Z is never stored on the server, making this approach promising for federated learning. However, since the server may attempt reconstructing the data from Z, one has to ensure each worker's privacy cannot be compromised; see Yue et al. (2022) for further details.

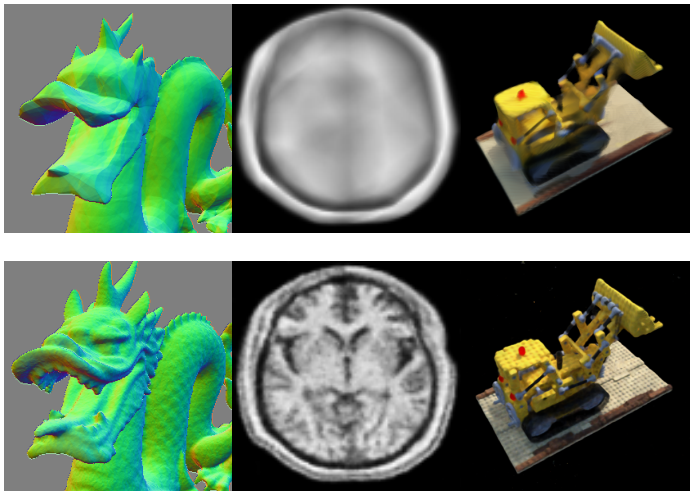
2.6.2 Pathology analysis

While the empirical NTK of a neural network is not the same as its limit NTK, they may have certain properties in common. In particular, certain issues of a finite-width network may reflect in certain issues of its limit NTK, and fixing these issues in the limit NTK may result in fixing them in a finite-width net.

As an example where this approach is proven to work, consider image regression. In this task, input samples are image coordinates, $x \in [0,1]^d$ for $d = 2$, and targets are pixel colors; we assume grey-scale images with $y \in [0,1]$. The task is therefore to regress the full image given a set of pixels.

Let us consider applying a fully-connected network for this task. As we have already observed in Section 2.4.1, the limit NTK $\Theta(x, x')$ of a fully-connected network depends only on $x^T x$, $x'^T x'$, and $x^T x'$. All of these terms are rotation-invariant, hence the kernel itself is rotation-invariant. However, none of this terms is translation-invariant, hence the kernel cannot be translation-invariant (otherwise, it has to be constant). Therefore it is quite unlikely that the empirical kernel will be invariant to translations.

On the other hand, both translation and rotation invariance are desirable for a kernel used for



(d) 3D shape regression(e) MRI reconstruction (f) Inverse rendering

Figure 2.2: Images are borrowed from Tancik et al. (2020).

image regression. Indeed, this means that applying these transformations to the train set of pixels results in the same image as without them, up to translation and rotation. In order to achieve this property, one may start working on translationaly invariant embeddings of image coordinates. The simplest non-trivial embedding of this kind is $z(x) = [\cos(2\pi x), \sin(2\pi x)]^T$, where cos and sin are applied elementwise. Following Tancik et al. (2020), we shall refer it as "basic". Comparing (b) and (c) of Fig. 2.1, this indeed results in better perceived quality.

However the regressed image is still blurry: see Fig. 2.1 (c). As we shall see shortly, NTK kernel regression learns low-frequency components of the image before its high-frequency ones. If we assume that the same property holds for the corresponding finite-width net then achieving sharp images may be impossible for a given number of gradient steps.

Recall the training dynamics of a kernel regression with kernel Θ trained to minimize square loss on a training dataset $(\vec{x}, \vec{y})$:

$$\dot{f}_t(\vec{x}) = \Theta(\vec{x}, \vec{x})(\vec{y} - f_t(\vec{x})). \tag{2.106}$$

Θ is a kernel, therefore its Gram matrix is positive-semidefinite. Consider its eigenvalue decomposition: $\Theta(\vec{x}, \vec{x}) = \sum_{k=1}^{m} \lambda_k \vec{v}_k \vec{v}_k^T$, where $\lambda_1 \geq \ldots \geq \lambda_m \geq 0$, and $(\vec{v}_k)_{k=1}^m$ forms an orthonormal basis.

Let us decompose our model's predictions as $f_t(\vec{x}) = \sum_{k=1}^{m} u_{t,k} \vec{v}_k$. Then the dynamics above decomposes as

$$u_{t,k} = \lambda_k (\vec{v}_k^T \vec{y} - u_{t,k}), \tag{2.107}$$

which solves as

$$u_{t,k} = \vec{v}_k^T \vec{y} - e^{-\lambda_k t} (\vec{v}_k^T \vec{y} - u_{0,k}). \tag{2.108}$$

As one clearly sees, time required to learn the k-th principal component of the target is inversely proportional to its strength λ_k. In other words, strong components are learned before weak ones.

The question is: what are the eigenvectors of the NTK Gram matrix? It is hard to answer this question in general since a Gram matrix depends on the dataset. However, for a kernel, there is an analogue of eigenvalue decomposition called Mercer's representation.

Let X be a compact metric space and let μ be a sigma-additive measure on X with $\operatorname{supp}\mu = X$. Suppose $K: X\times X \to \mathbb{R}$ is continuous, symmetric, and satisfies $\int_X \int_X K(x,x')f(x)f(x')\,d\mu(x)\,d\mu(x') < \infty\ \forall f \in L^2_\mu(X)$. Define Gram-Schmidt operator $T_K : L^2_\mu(X) \to L^2_\mu(X)$ as $T_K[f](x) = \int_X K(x,x')\,d\mu(x')$. Then the above operator admits an eigenvalue decomposition with eigenfunctions $(\psi_k)_{k=1}^\infty$ and corresponding eigenvalues $(\lambda_k)_{k=1}^\infty$, and the set of eigenfunctions forms an orthonormal basis in $L^2_\mu(X)$. The Mercer's representation is the corresponding decomposition of the kernel:

$$K(x,x') = \sum_{k=1}^\infty \lambda_k \psi_k(x)\psi_k(x'). \tag{2.109}$$

The series converges uniformly in $X \times X$.

From the above, we have $\int_X \int_X K(x,x')\psi_k(x)\psi_k(x')\,d\mu(x)\,d\mu(x') = \lambda_k\ \forall k \geq 1$. Hence if $\vec{x} = (x_k)_{k=1}^m$ and $\vec{x}' = (x'_k)_{k=1}^m$ are sampled iid from μ then

$$\begin{aligned}&\frac{1}{m^2}\psi_k^T(\vec{x})K(\vec{x},\vec{x}')\psi_k(\vec{x}')\\ &\quad= \frac{1}{m^2}\sum_{i,j=1}^m K(x_i,x'_j)\psi(x_i)\psi(x'_k) \to \int_X\int_X K(x,x')\psi_k(x)\psi_k(x')\,d\mu(x)\,d\mu(x') = \lambda_k\end{aligned} \tag{2.110}$$

a.s. as $m \to \infty$ by the Law of Large Numbers (LLN). Note that considering $\psi^T(\vec{x})K(\vec{x},\vec{x})\psi(\vec{x})$ instead of $\psi^T(\vec{x})K(\vec{x},\vec{x}')\psi(\vec{x}')$ may result in a different limit because the diagonal of K is now calculated on two dependent arguments. Nevertheless, there are only m elements on the diagonal, which results in $O(m^{-1})$ error vanishing in the limit. Hence

$$\frac{1}{m^2}\psi_k^T(\vec{x})K(\vec{x},\vec{x})\psi_k(\vec{x}) \to \lambda_k \tag{2.111}$$

a.s. as $m \to \infty$. In other words, given $\vec{x}$ sampled iid from μ, $(\psi_k(\vec{x}))_{k=1}^m$ are approximately the eigenvectors of $K(\vec{x},\vec{x})$ with eigenvalues $(m^2\lambda_k)_{k=1}^m$.

Recall that, as was noted above, the limit NTK of a fully-connected net $\Theta(z,z')$ depends only on z^Tz', $\|z\|_2$, and $\|z'\|_2$. Recall also that we have decided to embed inputs with $z(x) = [\cos(2\pi x), \sin(2\pi x)]^T$. This embedding maps $[0,1]^d$ on a d-dimensional torus that lies inside a $2d-1$-dimensional sphere. In this case, our $\Theta(x,x') = \Theta(z(x),z(x'))$ depends only on $z^T(x)z(x')$.

Kernels with this property are called zonal. Any zonal kernel $K : S^{p-1} \times S^{p-1} \to \mathbb{R}$ admits the following Mercer's decomposition with respect to the uniform measure on S^{p-1}:

$$K(z^Tz') = \sum_{k=0}^\infty \lambda_k \sum_{j=1}^{N(p,k)} Y_{k,j}(z)Y_{k,j}(z'), \tag{2.112}$$

where $N(p,k)$ are so-called Gegenbauer polynomials and $Y_{k,j}$ are spherical harmonics. For $p=2$, this decomposition gets a simpler form:

$$K(z^Tz') = \frac{1}{4\pi^2} + \frac{1}{\pi^2}\sum_{k=1}^\infty \lambda_k \cos(k\arccos(z^Tz')). \tag{2.113}$$

As we see, large k's correspond to high-frequency harmonics, while small k's correspond to low-frequency ones. A recent result of Chen and Xu (2021) states that the NTK of a fully-connected

net with inputs lying on S^{p-1} has eigenvalues decaying as a power-law: $\lambda_k \sim k^{-p}$ as $k \to \infty$; see also Geifman et al. (2020) for an earlier result for shallow nets and Bietti and Mairal (2019) for an even earlier result for bias-free shallow nets. This means that learning the k-th harmonic of the input image requires $O(k^p)$ time. Hence for a finite amount of training steps, high-frequency components remain not learned, which results in blurry images similar to Fig. 2.1 (c).

The possible remedy would be increasing λ_k for large k. But how to achieve it? We illustrate the solution proposed in Tancik et al. (2020) in the following.

Consider the case $d = 1$ for simplicity. In this case, the embedding map $z(x) = [\cos(2\pi x), \sin(2\pi x)]^T$ traverses a circle. Consider a modified embedding $\tilde{z}(x) = [\cos(2\pi bx), \sin(2\pi bx)]^T$ instead, where $b \in \mathbb{N}$ is a tunable parameter. The corresponding kernel is then given as

$$\begin{aligned} K(\tilde{z}^T \tilde{z}') &= \frac{1}{4\pi^2} + \frac{1}{\pi^2} \sum_{k=1}^{\infty} \lambda_k \cos(k \arccos(\tilde{z}^T \tilde{z}')) \\ &= \frac{1}{4\pi^2} + \frac{1}{\pi^2} \sum_{k=1}^{\infty} \lambda_k \cos(4\pi kb(x - x')) = \frac{1}{4\pi^2} + \frac{1}{\pi^2} \sum_{k=1}^{\infty} \lambda_k \cos(kb \arccos(z^T z')), \end{aligned} \quad (2.114)$$

which means that λ_k becomes the kb-th eigenvalue in the original embedding space. If λ_k decreased monotonically this would mean that each kb-th eigenvalue increased from λ_{kb} to λ_k, implying faster convergence to kb-th principal component.

The obvious downside of the method above is that in a new parameterization some of the eigenvalues become zero — therefore they are never learned. A simple solution is to enlarge the embedding: $\tilde{z}(x) = [\cos(2\pi\sigma^{j/M} x), \sin(2\pi\sigma^{j/M} x)]^T$, where $M \in \mathbb{N}$ and $\sigma \in \mathbb{R}_+$ are tunable parameters; this referred as "positional encoding" in Tancik et al. (2020). Another solution proposed by Tancik et al. (2020) is random Gaussian projections: $\tilde{z}(x) = [\cos(2\pi Bx), \sin(2\pi Bx)]^T$, where $B \in \mathbb{R}^{M \times d}$, each element of B is sampled independently from $\mathcal{N}(0, \sigma^2)$, and M and σ are tunable parameters. Both solution perform on par with each other and much better than the original embedding: compare (c), (d), and (e) in Fig. 2.1.

The same method suites other low-dimensional regression problems as well; Tancik et al. (2020) provide examples of 3D shape regression, MRI reconstruction, and inverse rendering. See Fig. 2.2 for comparison of outputs of a neural net with no enconding of inputs (top row) and the proposed Gaussian encoding (bottom row).

One more notable example is Solid Isotropic Material Penalisation, an instance of topology optimization. The task here is to optimize over material density at N points $y \in [0, 1]^N$ to obtain a shape that can withstand forces applied at certain points.

Given a density y and a force vector F, the SIMP method constructs a stiffness matrix $K(y)$, and derives a displacement vector $U(y)$ by solving a linear system $K(y)U(y) = F$. The resulting construction is stable if the forces do not do any work, i.e. $U^T(y)F = 0$. The density is therefore optimized to minimize the work $C(y) = U^T(y)FU(y) \to \min_y$ under a volume constraint $\sum_{i=1}^N y_i = V$; C is usually called compliance.

We can cast the constrained optimization problem as an unconstrained one by introducing pre-density $x \in \mathbb{R}^N$ and constructing density as $y_i = \sigma(x_i + b(x))$, where b is a function that ensures the volume constraint. Denoting this operation as $y = \Sigma(x)$, we get a new unconstrained optimization problem in the space of pre-densities: $C(\Sigma(x)) \to \min_x$.

While the above problem is not a regression problem, we can still model x as outputs of a neural net at the corresponding grid points. However, lack of translation invariance results in unplausible

patterns. Dupuis and Jacot (2021) used a similar embedding scheme as Tancik et al. (2020) to control this issue. On the other hand, in contrast to Tancik et al. (2020), Dupuis and Jacot (2021) used $\sin(\omega x)$ as activation instead of ReLU, and used ω together with bias initialization variance to control sharpness of output shapes, instead of modifying the embedding. Both methods aim to "widen" the spectrum of the limit NTK.

2.6.3 A theoretical tool

Apart from providing a meaningful kernel for kernel methods, NTK can be used as a concept useful for reasoning about neural nets of large width. Indeed, as stated in Section 2.2, NTK, while being random and evolving, converges to a constant deterministic limit as width goes to infinity. One can hope that for large enough width, the NTK stays close to its limit with high probability. Therefore, any result valid for kernel regression with NTK taken as a kernel, may become also valid with high probability for a wide enough net.

Global GD convergence

Let us start with the following result valid for kernel regression with a constant kernel: when the kernel is positive-definite, kernel regression learns the dataset. Indeed, recall the training dynamics of a kernel regression with kernel Θ trained to minimize square loss on a training dataset $(\vec{x}, \vec{y})$:

$$\dot{f}_t(\vec{x}) = \Theta(\vec{x}, \vec{x})(\vec{y} - f_t(\vec{x})). \tag{2.115}$$

Assuming $\Theta(\vec{x}, \vec{x}) \geq \lambda$,

$$\frac{d}{dt}\left(\frac{1}{2}\|\vec{y} - f_t(\vec{x})\|_2^2\right) = -(\vec{y} - f_t(\vec{x}))^T \Theta(\vec{x}, \vec{x})(\vec{y} - f_t(\vec{x})) \leq -\lambda \|\vec{y} - f_t(\vec{x})\|_2^2, \tag{2.116}$$

which gives

$$\|\vec{y} - f_t(\vec{x})\|_2^2 \leq e^{-2\lambda t}\|\vec{y} - f_0(\vec{x})\|_2^2. \tag{2.117}$$

Hence $\lambda > 0$ suffices to guarantee that $f_t(\vec{x})$ converges to $\vec{y}$ as $t \to \infty$.

Suppose now our kernel regression uses a random time-dependent kernel $\hat{\Theta}_t$ instead of Θ:

$$\dot{f}_t(\vec{x}) = \hat{\Theta}_t(\vec{x}, \vec{x})(\vec{y} - f_t(\vec{x})). \tag{2.118}$$

If we manage to guarantee that with probability $\geq 1 - \delta$ $\forall t \geq 0$ $\hat{\Theta}_t(\vec{x}, \vec{x}) \geq \lambda$ then $\lambda > 0$ suffices to guarantee that $f_t(\vec{x})$ converges to $\vec{y}$ as $t \to \infty$ with probability $\geq 1 - \delta$. Indeed,

$$\frac{d}{dt}\left(\frac{1}{2}\|\vec{y} - f_t(\vec{x})\|_2^2\right) = -(\vec{y} - f_t(\vec{x}))^T \hat{\Theta}_t(\vec{x}, \vec{x})(\vec{y} - f_t(\vec{x})) \leq -\lambda \|\vec{y} - f_t(\vec{x})\|_2^2 \quad \text{w.p. } \geq 1 - \delta, \tag{2.119}$$

which gives

$$\|\vec{y} - f_t(\vec{x})\|_2^2 \leq e^{-2\lambda t}\|\vec{y} - f_0(\vec{x})\|_2^2 \quad \text{w.p. } \geq 1 - \delta. \tag{2.120}$$

One of the first results of this kind concerns ReLU nets with one hidden layer under NTK parameterization:

$$f(x; a_{1:n}, w_{1:n}) = \frac{1}{\sqrt{n}} \sum_{i=1}^{n} a_i [w_i^T x]_+. \tag{2.121}$$

We aim to minimize square loss on a dataset $(\vec{x}, \vec{y})$ of size m with gradient descent on the input weights:

$$\dot{w}_i(t) = \frac{1}{\sqrt{n}} \sum_{k=1}^{m} (y_k - f(x_k; a_{1:n}, w_{1:n}(t))) a_i [w_i^T(t) x_k > 0] x_k \quad \forall i \in [n]. \tag{2.122}$$

We sample $w_i \sim \mathcal{N}(0, I_{n_0})$ and $a_i \in U(\{-1, 1\})$ $\forall i \in [n]$ independently. The goal of sampling a_i from this particular distribution is mere simplification: in this case $a_i^2 = 1$, which simplifies the NTK Gram matrix a little bit:

$$\hat{\Theta}_t(x_k, x_l) = \frac{1}{n} \sum_{i=1}^{n} [w_i^T(t) x_k > 0][w_i^T(t) x_l > 0] x_k^T x_l. \tag{2.123}$$

However, it is possible to apply the same technique to any distribution of the output layer not depending on n. Note that the Gram matrix depends merely on activation patterns of the hidden layer computed on the dataset.

The limit NTK is therefore given as:

$$\Theta(x_k, x_l) = \mathbb{E}_{w \sim \mathcal{N}(0, I_{n_0})} [w^T x_k > 0][w^T x_l > 0] x_k^T x_l. \tag{2.124}$$

Note that in our two-layered case, $\Theta(x, x') = \lim_{n \to \infty} \hat{\Theta}_t(x, x') = \mathbb{E}\, \hat{\Theta}_0(x, x')$. In the sequel, we denote the Gram matrices $\hat{\Theta}_t(\vec{x}, \vec{x})$ as $H(t)$ and $\Theta(\vec{x}, \vec{x})$ as H^∞. Let λ_0 to be the least eigenvalue of H^∞.

Theorem 2.6.1 (Du et al. (2019b))**.** *Consider the setting discussed above and further assume* $\|x_k\|_2 \geq 1$ *and* $|y_k| \leq 1$ $\forall k \in [m]$*. Then* $\exists C, C_0 > 0$ *such that* $\forall \delta \in (0, 1)$ *taking*

$$n > \max\left(C \frac{m^6}{\lambda_0^4 \delta^3},\ C_0 \frac{m^2}{\lambda_0^2} \log\left(\frac{2m}{\delta} \right) \right) \tag{2.125}$$

guarantees $H(t) \geq \lambda_0 / 2$ $\forall t \geq 0$ *w.p.* $\geq 1 - \delta$*.*

This result implies $\|\vec{y} - f_t(\vec{x})\|_2^2 \leq e^{-\lambda_0 t} \|\vec{y} - f_0(\vec{x})\|_2^2$ w.p. $\geq 1 - \delta$, as discussed above.

For the full proof, see the original paper Du et al. (2019b) or lecture notes Golikov (2020b). We are going to discuss, very briefly, only crucial parts of the proof in the sequel.

The proof is based on four lemmas. The first lemma states that as long as $n = \Omega(m^2 \lambda_0^{-2} \log(m/\delta))$, where Ω hides a certain constant, $\|H(0) - H^\infty\|_2 \leq \lambda_0 / 4$, where $\|\cdot\|_2$ denotes a singular norm, w.p. $\geq 1 - \delta$; this implies $H(0) \geq 3\lambda_0 / 4$ with the same probability. As already noted above, $\mathbb{E}\, H(0) = H^\infty$. This allows one to apply a concentration inequality to each element of $H(0)$. Union bound then gives a bound that holds uniformly for all elements of $H(0)$. This implies a bound on $\|H(0) - H^\infty\|_F$, hence on a singular norm as well.

The second lemma states that as long as $\forall i \in [n]$ $\|w_i - w_i(0)\|_2 \leq R$ for certain $R = R(\delta, \lambda_0, m)$, $\|H - H(0)\|_2 \leq \lambda_0 / 4$ w.p. $\geq 1 - \delta$. In other words, as long as weights are close to initialization, the corresponding Gram matrix is close to the initial one too. The idea is that as long as the weights are not far from their initialization, with certain probability, not many of the hidden neurons can alter their activation patterns on the train dataset. Since as already noted above, our Gram matrices depend only on activation patterns on the train dataset, this implies a tail bound on $|H_{kl}(0) - H_{kl}^\infty|$ $\forall k, l \in [m]$, which gives a tail bound on $\|H(0) - H^\infty\|_2$ with the same technique as used in the first lemma.

The third lemma states that as long as $H(s) \geq \lambda_0/2 \; \forall s \in [0,t]$ (we haven't proven it yet), weights indeed stay close to their initialization: $\forall i \in [n] \; \|w_i(t) - w_i(0)\|_2 \leq R'$ for certain $R' = R'(\lambda_0, m, n)$. This can be proven by a very simple estimate:

$$\left\| \frac{dw_i(s)}{ds} \right\|_2 = \left\| \frac{1}{\sqrt{n}} \sum_{k=1}^{m} (y_k - f_s(x_k)) a_i [w_i^T(s) x_k > 0] x_k \right\|_2 \leq$$
$$\leq \frac{1}{\sqrt{n}} \sum_{k=1}^{m} |y_k - f_s(x_k)| \leq \sqrt{\frac{m}{n}} \|\vec{y} - f_s(\vec{x})\|_2 \leq \sqrt{\frac{m}{n}} e^{-\lambda_0 s/2} \|\vec{y} - f_0(\vec{x})\|_2. \quad (2.126)$$

This gives $\forall i \in [n]$:

$$\|w_i(t) - w_i(0)\|_2 = \left\| \int_0^t \frac{dw_i(s)}{ds} \, ds \right\|_2 \leq \int_0^t \left\| \frac{dw_i(s)}{ds} \right\|_2 ds \leq$$
$$\leq \frac{2\sqrt{m}}{\lambda_0 \sqrt{n}} \left(1 - e^{-\lambda_0 t/2}\right) \|\vec{y} - f_0(\vec{x})\|_2 \leq \frac{2\sqrt{m}}{\lambda_0 \sqrt{n}} \|\vec{y} - f_0(\vec{x})\|_2. \quad (2.127)$$

Finally, the fourth lemma states that as long as $R' < R$, $\|H(t) - H(0)\|_2 \leq \lambda_0/4 \; \forall t \geq 0$ w.p. $\geq 1 - \Omega(\delta)$ where Ω hides a certain constant. Combined with the first lemma, this implies $H(t) \geq \lambda_0/2 \; \forall t \geq 0$ w.p. $\geq 1 - \Omega(\delta)$. The condition $R'(\lambda_0, m, n) < R(\delta, \lambda_0, m)$ gives the second lower bound on n (the first one is given be the first lemma). By changing δ, we get the desired result.

The fourth lemma is proven as follows. Let t_0 be the first moment of time when the second lemma becomes no longer applicable, i.e. $t_0 = \inf \left\{ t \geq 0 : \; \max_{i \in [n]} \|w_i(t) - w_i(0)\|_2 > R \right\}$. Assume it is finite. Since weights are continuous functions of time, $\max_{i \in [n]} \|w_i(t_0) - w_i(0)\|_2 = R$. Hence the second lemma holds for $w_{1:n} = w_{1:n}(t) \; \forall t \in [0, t_0]$ and $\|H(t) - H(0)\|_2 \leq \lambda_0/4$ w.p. $\geq 1 - \delta$ $\forall t \in [0, t_0]$, therefore $H(t) \geq \lambda_0/2$ w.p. $\geq 1 - \Omega(\delta) \; \forall t \in [0, t_0]$. But then the third lemma holds as well: $\forall i \in [n] \; \|w_i(t_0) - w_i(0)\|_2 \leq R' < R$; contradiction. Hence $\forall t \geq 0 \; \max_{i \in [n]} \|w_i(t) - w_i(0)\|_2 \leq R$ and the second lemma gives the desired statement.

Theorem 2.6.1 requires the number of hidden units n to grow as m^6 with the size of a train dataset and as δ^{-3} with the failure probability. This bound is way too loose for practical purposes: indeed, even for very small datasets $m \geq 100$ which results in a bound of the order at least 10^8. If we want the bound to be valid with at least 90% probability, we pay three orders of magnitude more. Note that modern architectures designed to be trained on large datasets like ImageNet ($m = 10^6$) have width barely exceeding 10^4.

We state one of the existing improvements of Theorem 2.6.1 below:

Theorem 2.6.2 (Song and Yang (2019))**.** *Under the same setting as Theorem 2.6.1,* $\exists C, C_0 > 0$ *such that* $\forall \delta \in (0,1)$ *taking*

$$n > \max \left(C \frac{m^4}{\lambda_0^4} \log^3 \left(\frac{m}{\delta} \right), \; C_0 \frac{m^2}{\lambda_0^2} \log \left(\frac{2m}{\delta} \right) \right) \quad (2.128)$$

guarantees $H(t) \geq \lambda_0/2 \; \forall t \geq 0$ *w.p.* $\geq 1 - \delta$.

This result decreases the exponent of m from 6 to 4 and makes the δ-dependence logarithmic. The proof follows the same path as above. Note however that the previous result aimed for elementwise

tail bounds on $H(0)-H^\infty$ or $H-H(0)$ which lead to tail bounds on $\|H(0)-H^\infty\|_2$ and $\|H-H(0)\|_2$ by union bound, which gives an m^2 factor. One of the improvements proposed by Song and Yang (2019) is to replace these elementwise bounds with matrix-Chernoff bounds — they do not give this m^2 factor, thus leading to better bounds. The other improvement is to replace Markov inequalities that result in $1/\delta$ factors with Bernstein inequality that results only in $\log(1/\delta)$ ones.

The m^4 width bound is still far from being realistically tight. Under the same setting as Theorem 2.6.1, Oymak and Soltanolkotabi (2020) proves that the minimal eigenvalue of the NTK remains lower-bounded with a positive constant as long as $nd \gtrsim m^2$; that is, when the number of parameters of the trained input layer grows quadratically with the dataset size. This improves the bound to be quadratic in m, and also improves the leading constant by d.

Bombari et al. (2022) further improves the NTK stability guarantee to $n > \sqrt{m}$ for network with at least two hidden layers, all of width n. This guarantee is asymptotically tight, since for smaller width the number of parameters becomes less than the number of training points, therefore the network is no longer guaranteed to be able to fit generic data. However, this work assumes specific parameterization different from both the NTK and the standard ones.

Global gradient descent convergence can be also proven by first proving gurantees on convergence to local minima and then proving that all minima are global for wide enough nets. See Lee et al. (2016); Panageas and Piliouras (2017); Mertikopoulos et al. (2020) for the first line of works and Yu and Chen (1995); Nguyen and Hein (2017); Nguyen (2019b, 2021) for the second. None of the works of both lines use the idea of NTK stability and they neither rely on NTK parameterization. Nguyen (2019b) proves that $n = m$ is enough of leaky ReLU nets to have only global "local valleys" (generalization of global minima to certain losses such as cross-entropy) and Nguyen (2021) demonstrates that this bound cannot be improved for two-layered nets and general data.

Du et al. (2019a) extends Theorem 2.6.1 to deep nets. Their proof idea is the same: first show that $H(0)$ is close to H^∞, then show that $H(t)$ stays close to $H(0)$. However for the multilayer case, $H(0)$ cannot be proven to be close to H^∞ just by concentration of measure. When layers are many, perturbations caused by finite width result in deviations exponential with respect to the number of layers L. For this reason, their bound grows exponentially with L. Using a different technique, Nguyen and Mondelli (2020) improves their result to a quadratic scaling with width, but still exponential in depth. See also Allen-Zhu et al. (2019); Yun et al. (2019) for a similar result with a bound depending polynomially on both width and depth, proven using a different technique.

Generalization guarantees

Stability of NTK has another interesting consequence. Suppose the empirical NTK is constant, i.e. $\hat{\Theta}_t = \hat{\Theta}_0$. It is equivalent to say that the corresponding model is linearized:

$$f(x;\theta) = f(x;\theta_0) + \nabla_\theta^T f(x;\theta_0)(\theta - \theta_0). \tag{2.129}$$

For brevity, denote $\vec{u}_t = f_t(\vec{x})$ and $Z_t^{ik} = \partial_{\theta_i} f(x_k;\theta_t)$. Hence $Z_t \in \mathbb{R}^{N\times m}$ where N is the total number of parameters and $\vec{u}_t = \vec{u}_0 + Z_0^T(\theta_t - \theta_0)$.

Note that $H_t = Z_t^T Z_t$. Recall the train set predictions for constant kernel:

$$\vec{u}_t = \vec{y} + e^{-H_0 t}(\vec{u}_0 - \vec{y}). \tag{2.130}$$

In our linearized dynamics, the weights evolve as follows:

$$\dot{\theta}_t = Z_0(\vec{y} - \vec{u}_t) = Z_0 e^{-H_0 t}(\vec{y} - \vec{u}_0). \tag{2.131}$$

Straightforward integration gives:

$$\theta_t = \theta_0 + Z_0 H_0^{-1} \left(I - e^{-H_0 t}\right)(\vec{y} - \vec{u}_0). \tag{2.132}$$

Recalling $H_0 = Z_0^T Z_0$, at the end of training ($t \to \infty$) we get

$$\|\theta_\infty - \theta_0\|_2^2 = (\theta_\infty - \theta_0)^T(\theta_\infty - \theta_0) = (\vec{y} - \vec{u}_0)^T H_0^{-1}(\vec{y} - \vec{u}_0). \tag{2.133}$$

Define $\mathcal{F}_B^{w_{1:n}(0),a_{1:n}}$ as a set of models of the form (2.121) with output weights $a_{1:n}$ and input weights $w_{1:n}$ such that $\|W - W(0)\|_F \le B$ for given $w_{1:n}(0)$. The above considerations state that a trained model always lies in $\mathcal{F}_B^{w_{1:n}(0),a_{1:n}}$ with $B = (\vec{y} - \vec{u}_0)^T H_0^{-1}(\vec{y} - \vec{u}_0)$.

Hence our training procedure outputs models in a certain set rather than any model in of the form (2.121). Upper-bounding Rademacher complexity of this model set will give us a generalization bound as we shall see below. Let us upper-bound the Rademacher complexity conditioned on a dataset $(\vec{x}, \vec{y})$ of size m:

$$\begin{aligned}
\mathrm{Rad}(\mathcal{F}_B^{w_{1:n}(0),a_{1:n}} \mid \vec{x}, \vec{y}) &= \mathbb{E}_{\sigma_{1:m}\sim\{-1,1\}^m} \sup_{f \in \mathcal{F}_B^{w_{1:n}(0),a_{1:n}}} \left(\frac{1}{m}\sum_{k=1}^m \sigma_k u_k\right) = \\
&= \frac{1}{m}\mathbb{E}_{\sigma_{1:m}\sim\{-1,1\}^m} \sup_{\|W - W(0)\|_F \le B} \left(\sum_{k=1}^m \sigma_k \frac{1}{\sqrt{n}} \sum_{i=1}^n a_i [w_i^T(0) x_k \ge 0] w_i^T x_k\right) = \\
&= \frac{1}{m}\mathbb{E}_{\sigma_{1:m}\sim\{-1,1\}^m} \sup_{\|W - W(0)\|_F \le B} \left(\vec{\sigma}^T Z^T(0)\theta\right) = \\
&= \frac{1}{m}\mathbb{E}_{\sigma_{1:m}\sim\{-1,1\}^m} \sup_{\|W - W(0)\|_F \le B} \left(\vec{\sigma}^T \tilde{Z}^T(0)(\theta - \theta_0)\right) = \\
&= \frac{B}{m}\mathbb{E}_{\sigma_{1:m}\sim\{-1,1\}^m} \|Z(0)\vec{\sigma}\|_2 \le \frac{B}{m}\sqrt{\mathbb{E}_{\sigma_{1:m}\sim\{-1,1\}^m} \|Z(0)\vec{\sigma}\|_2^2} = \frac{B}{m}\|Z(0)\|_F.
\end{aligned} \tag{2.134}$$

Note that

$$\|Z(0)\|_F^2 = \frac{1}{n}\sum_{i=1}^n \sum_{k=1}^m [w_i^T(0) x_k \ge 0]. \tag{2.135}$$

It is an average of i.i.d random variables, which allows for Hoeffding's inequality:

$$\mathcal{P}(\|Z(0)\|_F^2 - \frac{m}{2} \ge \epsilon) \le e^{-2n\epsilon^2/m^2}. \tag{2.136}$$

This gives w.p. $\ge 1 - \delta$ over initialization,

$$\|Z(0)\|_F^2 \le \frac{m}{2} + \sqrt{\frac{m^2}{2n}\log\left(\frac{1}{\delta}\right)}. \tag{2.137}$$

Finally, we got that w.p. $\ge 1 - \delta$ over initialization,

$$\mathrm{Rad}(\mathcal{F}_B^{w_{1:n}(0),a_{1:n}} \mid (\vec{x}, \vec{y})) \le \frac{B}{\sqrt{m}}\sqrt{\frac{1}{2} + \sqrt{\frac{1}{2n}\log\left(\frac{1}{\delta}\right)}}. \tag{2.138}$$

Consider zero-one risk: $r(y,z) = [yz < 0]$; we have $R(f) = \mathbb{E}_{x,y\sim\mathcal{D}} r(y, f(x))$ and $\hat{R}(f) = \mathbb{E}_{x,y\in S_m} r(y, f(x))$, correspondingly. From the generalization theory, we know that for any B and for any initialization $w_{1:n}(0), a_{1:n}$, w.p. $\geq 1-\tilde{\delta}$ over the training dataset, $\forall f \in \mathcal{F}_B^{w_{1:n}(0),a_{1:n}}$,

$$R(f) \leq \hat{R}_m(f) + \mathbb{E}_{(\vec{x},\vec{y})} \mathrm{Rad}(\mathcal{F}_B^{w_{1:n}(0),a_{1:n}} \mid (\vec{x},\vec{y})) + \sqrt{\frac{1}{2m}\log\frac{1}{\tilde{\delta}}} \quad \text{w.p. } \geq 1-\tilde{\delta} \text{ over } (\vec{x},\vec{y}). \tag{2.139}$$

We want to take $B = (\vec{y}-\vec{u}_0)^T H_0^{-1}(\vec{y}-\vec{u}_0)$ but it depends on the dataset $(\vec{x},\vec{y})$. Take a sequence $\{B_j\}_{j=1}^{\infty}$ monotonically increasing to infinity and a sequence $\{\tilde{\delta}_j\}_{j=1}^{\infty}$ of deltas $\in (0,1)$ that sum to $\tilde{\delta}$. This allows us to apply a union bound: w.p. $\geq 1-\tilde{\delta}$ over the training dataset, for any initialization $w_{1:n}(0), a_{1:n}$, $\forall j \in \mathbb{N}$, $\forall f \in \mathcal{F}_{B_j}^{w_{1:n}(0),a_{1:n}}$,

$$R(f) \leq \hat{R}_m(f) + \mathbb{E}_{(\vec{x},\vec{y})} \mathrm{Rad}(\mathcal{F}_{B_j}^{w_{1:n}(0),a_{1:n}} \mid (\vec{x},\vec{y})) + \sqrt{\frac{1}{2m}\log\frac{1}{\tilde{\delta}_j}}. \tag{2.140}$$

We are free to choose minimal j such that $B_j \geq (\vec{y}-\vec{u}_0)^T H_0^{-1}(\vec{y}-\vec{u}_0)$; denote it by $\hat{j}$. Let for definiteness $B_j = j$. Then $B_{\hat{j}} \leq 1 + (\vec{y}-\vec{u}_0)^T \hat{\Theta}_0^{-1}(\vec{y}-\vec{u}_0)$.

Putting all together, we have w.p. $\geq 1-\tilde{\delta}$ over the training dataset, w.p. $\geq 1-\delta$ over initialization,

$$R(f(\theta_\infty)) \leq \hat{R}_m(f(\theta_\infty)) + \\ + \frac{1 + (\vec{y}-\vec{u}_0)^T H_0^{-1}(\vec{y}-\vec{u}_0)}{\sqrt{m}} \sqrt{\frac{1}{2} + \sqrt{\frac{1}{2n}\log\left(\frac{1}{\delta}\right)}} + \sqrt{\frac{1}{2m}\log\frac{1}{\tilde{\delta}_{\hat{j}}}}. \tag{2.141}$$

Recall that the bound above was obtained under the assumption of constant NTK. In order to relax this assumption, one has to show that, possibly for large enough width, H_t^{-1} stays close to H_0^{-1}. Note that when proving global GD convergence we had to prove that H_t stays close to H_0, which is different. The required closeness result is proven in Arora et al. (2019a), it leads to the following theorem:

Theorem 2.6.3 (Arora et al. (2019a))**.** *Under the same setting as Theorem 2.6.1, $\exists p, C, C_0 > 0$ such that $\forall \delta \in (0,1)$ taking*

$$n > \max\left(C\frac{m^7}{\lambda_0^4 \delta^p},\ C_0\frac{m^2}{\lambda_0^2}\log\left(\frac{2m}{\delta}\right)\right) \tag{2.142}$$

guarantees w.p. $\geq 1-\delta$ over the training dataset of size m and w.p. $\geq 1-\delta$ over initialization,

$$R(f(\theta_\infty)) \leq \hat{R}_m(f(\theta_\infty)) + \\ + \frac{1 + (\vec{y}-\vec{u}_0)^T (H^\infty)^{-1} (\vec{y}-\vec{u}_0)}{\sqrt{m}} \sqrt{\frac{1}{2} + \sqrt{\frac{1}{2n}\log\left(\frac{1}{\delta}\right)}} + \sqrt{\frac{1}{2m}\log\frac{1}{\delta}}. \tag{2.143}$$

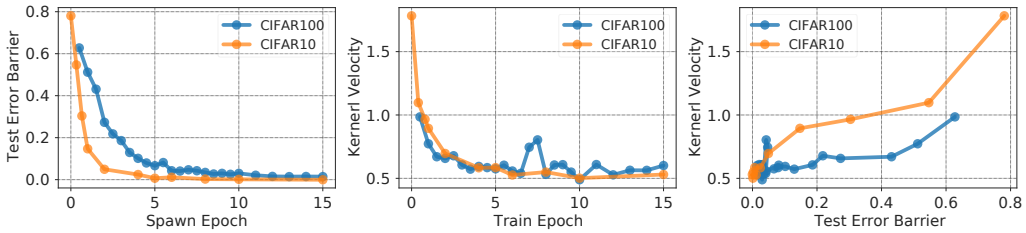


Figure 2.3: The figure is borrowed from Fort et al. (2020).

2.7 Standard parameterization and kernel evolution

As was noted in Section 2.2, NTK diverges under standard parameterization. Recall the example of a two-layered net:

$$f(x; a_{1:n}, w_{1:n}) = \sum_{i=1}^{n} a_i \phi(w_i x), \quad a_{1:n} \sim \mathcal{N}(0, n^{-1} I), \quad w_{1:n} \sim \mathcal{N}(0, I); \tag{2.144}$$

$$\hat{\Theta}_t(x, x') = \sum_{i=1}^{n} \left(\phi(w_i(t)x)\phi(w_i(t)x') + a_i^2(t)\phi'(w_i(t)x)\phi'(w_i(t)x')xx' \right). \tag{2.145}$$

At $t = 0$, since w_i are independent and of the order of $O(1)$, the sum diverges proportionaly to n. Since under square loss, $\dot{f}_t(x) = \hat{\Theta}_t(x, \vec{x})(\vec{y} - f_t(\vec{x}))$, the model prediction at any point x receive a $O(n)$ increment at the very beginning of training. In other words, model predictions diverge with width, making the model useless for regression.

However, if the goal is classification, magnitude of predictions does not matter; what matters is their signs for binary classification, or indices of the largest logits when classes are multiple. Therefore in this case, an infinite-width limit under standard parameterization still may make sense besides of divergent NTK, see Golikov (2020a).

In order to deal with divergence, consider a normalized empirical NTK $\tilde{\Theta}_t(x, x') = \hat{\Theta}_t(x, x')/n$; its infinite-width limit at initialization is $\mathbb{E}_{w \sim \mathcal{N}(0,1)} \phi(wx)\phi(wx')$; we shall refer it as normalized NTK and denote as $\tilde{\Theta}(x, x')$. In contrast to NTK under NTK parameterization, normalized NTK under standard parameterization evolves with time (Golikov, 2020a):

$$\begin{aligned} \frac{d\tilde{\Theta}_t(x, x')}{dt} = &\frac{1}{n} \sum_{i=1}^{n} \left(\phi(w_i(t)x)\phi'(w_i(t)x')x' + \phi'(w_i(t)x)\phi(w_i(t)x')x \right) \frac{dw_i(t)}{dt} \\ &+ \frac{1}{n} \sum_{i=1}^{n} a_i^2(t)xx' \left(\phi'(w_i(t)x)\phi''(w_i(t)x')x' + \phi''(w_i(t)x)\phi(w_i(t)x')x \right) \frac{dw_i(t)}{dt} \\ &+ \frac{1}{n} \sum_{i=1}^{n} 2a_i(t)\phi'(w_i(t)x)\phi'(w_i(t)x')xx' \frac{da_i(t)}{dt}. \end{aligned} \tag{2.146}$$

Recall the gradient flow dynamics under standard parameterization:

$$\frac{a_k(t)}{dt} = \sum_{j=1}^{m} \phi(w_k(t)x_j), \quad \frac{w_k(t)}{dt} = \sum_{j=1}^{m} a_k(t)\phi'(w_k(t)x_j)x_j. \tag{2.147}$$

At $t = 0$, we have $\dot{a}_k = O(1)$, while $\dot{w}_k = O(n^{-1/2})$. Since $a_k(0) = O(n^{-1/2})$ and $w_k(0) = O(1)$, it means that for any $t > 0$ independent on n, $a_k(t) = O(1)$, $\dot{a}_k(t) = O(1)$, $w_k(t) = O(1)$, and $\dot{w}_k(t) = O(1)$. A naive estimate of the sums then gives $\frac{d\bar{\Theta}_t(x,x')}{dt} = O(1) + O(1) + O(1) = O(1)$ for any $t > 0$ independent on n. Therefore the normalized kernel keeps evolving with time even in the limit of infinite width.

This can be the reason for superior performance of neural networks to conventional kernel methods and NTK. A kernel measures similarity between points in a feature space. While for NTK this feature space is fixed, a neural net varies its corresponding kernel feature space, hopefully making it better suitable for the task at hand; moreover, under standard parameterization, this feature does not vanish for large width.

The way an empirial NTK varies with time can be measured with kernel velocity, defined as kernel distance between the kernels corresponding to two consequent optimization steps. Kernel distance is in its turn defined as one minus cosine similarity between Gram matrices H and H' of the corresponding kernels:

$$\rho(H, H') = 1 - \frac{\operatorname{tr}(HH'^{,T})}{\sqrt{\operatorname{tr}(HH^T)\operatorname{tr}(H'H'^{,T})}}. \tag{2.148}$$

After measuring kernel velocity for a realistic net under standard parameterization, Fort et al. (2020) distinguished two phases of training: a phase of rapid kernel evolution, and a phase of almost constant NTK, see Section 2.7. The first phase is called *chaotic*, while the second one is coined *ordered*. Curiously enough, these two phases can be distinguished not only by kernel velocity. Suppose the network is trained up to time T, called *spawn epoch*. Two independent copies of the same network is then trained further. In other words, we train two networks which remain the same up to time T and may diverge afterwards due to randomness of training procedure. We then measure *test error barrier* between these two networks, i.e. height of the error "hill" on a straight segment between their corresponding weights. A small error barrier would mean that training of the two networks ended up in the same valley of test error, which likely means that they are similar. As one can see in Section 2.7, the test error barrier drops dramatically with growth of spawn epoch. Also, the two quantities under discussion, kernel velocity and error barrier appear to be strongly correlated, see again Section 2.7. There are also other quantities that experience sharp transition on the border of the two phases: kernel distance between child networks as a function of spawn epoch, ReLU activation Hamming distance, and Hamming distance between responses on the test set; see Fort et al. (2020) for details.

2.8 Beyond NTK

While NTK kernel regression has a natural interpretation of training an infinitely wide neural network under certain parameterization with gradient flow (see Section 2.2), NTK is not the only possible kernel that can be constructed using a neural net.

2.8.1 NNGP kernel

One of the other notable "neural kernels" is the NNGP-kernel (Lee et al., 2018), defined as $K(x, x') = \mathbb{E}_\theta f(x;\theta) f(x';\theta)$, where $f(\cdot;\theta)$ is a parametric model with weights θ and scalar output. Suppose f is a neural network with the output layer of the form $f(x) = v^T h(x)$, where $h(x) \in \mathbb{R}^n$ is

its last layer representation and $v \sim \mathcal{N}(0, I_n/n)$ independent on h. Then $K(x,x') = \frac{1}{n}\mathbb{E}\, h^T(x)h(x')$. As we have seen in Section 2.4 on the example of fully-connected and convolutional nets, the last layer representations tend to iid Gaussians as width go to infinity. In other words, $\forall i \in [n]$ h^i tend to identical and independent Gaussian processes with covariance $\mathbb{E}\, h^i(x)h^i(x') = \frac{1}{n}\mathbb{E}\, h^T(x)h(x')$, which is exactly $K(x,x')$. This motivates the term "NNGP" — *Neural Network Gaussian Process.*

Note that we have already seen the object $\mathbb{E}\, h^i(x)h^i(x')$ in Section 2.4: when $h = h_l$ — the l-th layer hidden representation of a fully-connected network, the above object is hidden layer covariance $q_l(x,x')$. Therefore the NNGP of this fully-connected network is nothing else but $q_L(x,x')$. This can be generalized to the whole class of architectures expressible by tensor programs: see the Master theorem of Yang (2019b) mentioned in Section 2.2. That is, any neuron of any hidden representation of a neural network expressible by a tensor program tends to a Gaussian process.

Learning a Gaussian process with zero mean and covariance $K(\cdot,\cdot)$ on a training dataset $(\vec{x}, \vec{y})$ means computing its Bayesian prosterior, which is again a Gaussian with mean $\mu(\cdot \,|\, (\vec{x}, \vec{y}))$ and covariance $K(\cdot,\cdot \,|\, (\vec{x}, \vec{y}))$ given below:

$$\mu(x \,|\, (\vec{x}, \vec{y})) = K(x, \vec{x})K^{-1}(\vec{x}, \vec{x})\vec{y}; \tag{2.149}$$

$$K(x, x' \,|\, (\vec{x}, \vec{y})) = K(x,x') - K(x, \vec{x})K^{-1}(\vec{x}, \vec{x})K(\vec{x}, x'). \tag{2.150}$$

Interestingly, training the last layer of an infinitely wide network with NNGP $K(\cdot,\cdot)$ results in exactly the same Gaussian process. When only the last layer is trained, the NNGP coincides with the NTK. Indeed, an NTK-parameterized NN of width n with readout weights v can be expressed as $f(x) = \frac{1}{\sqrt{n}} v^T h(x)$ with $v \sim \mathcal{N}(0, I_n)$. The empirical NTK is therefore given by $\hat{\Theta}_0(x,x') = \frac{1}{n}\nabla_v^T(v^T h(x))\nabla_v(v^T h(x')) = \frac{1}{n} h^T(x)h(x')$, which converges to $\mathbb{E}\, h^i(x)h^i(x') = K(x,x')$ as $n \to \infty$; note that $h(\cdot)$ also depends on n.

Recall the model prediction dynamics under constant NTK which is K in our case:

$$f_t(x) = f_0(x) - K(x, \vec{x})K^{-1}(\vec{x}, \vec{x}) \left(I - e^{-K(\vec{x},\vec{x})t}\right)(f_0(\vec{x}) - \vec{y}). \tag{2.151}$$

Since $f_0(\cdot)$ is a Gaussian process as discussed before and $K(\vec{x}, \vec{x})$ is deterministic, $f_t(\cdot)$ is a Gaussian process for any $t \geq 0$. Its mean $\mu_t(\cdot)$ and covariance $K_t(\cdot,\cdot)$ are:

$$\mu_t^{NNGP}(x) = K(x, \vec{x})K^{-1}(\vec{x}, \vec{x}) \left(I - e^{-K(\vec{x},\vec{x})t}\right)\vec{y}; \tag{2.152}$$

$$\begin{aligned} K_t^{NNGP}(x,x') &= K(x,x') \\ &+ K(x,\vec{x})K^{-1}(\vec{x},\vec{x}) \left(I - e^{-K(\vec{x},\vec{x})t}\right) K(\vec{x},\vec{x}) \left(I - e^{-K(\vec{x},\vec{x})t}\right) K^{-1}(\vec{x},\vec{x})K(\vec{x},x') \\ &- \left[K(x,\vec{x})K^{-1}(\vec{x},\vec{x}) \left(I - e^{-K(\vec{x},\vec{x})t}\right) K(\vec{x},x') + K(x',\vec{x})K^{-1}(\vec{x},\vec{x}) \left(I - e^{-K(\vec{x},\vec{x})t}\right) K(\vec{x},x)\right]. \end{aligned} \tag{2.153}$$

It is easy to see that $\mu_t^{NNGP}(x) \to \mu(x \,|\, (\vec{x}, \vec{y}))$ and $K_t^{NNGP}(x,x') \to K(x, x' \,|\, (\vec{x}, \vec{y}))$ as $t \to \infty$ $\forall x, x'$.

If not only the last layer is trained, NNGP does not generally correspond to NTK. The corresponding training dynamics is given by

$$f_t(x) = f_0(x) - \Theta(x, \vec{x})\Theta^{-1}(\vec{x}, \vec{x}) \left(I - e^{-\Theta(\vec{x},\vec{x})t}\right)(f_0(\vec{x}) - \vec{y}). \tag{2.154}$$

While $f_t(\cdot)$ is again a Gaussian process for any $t \geq 0$, its mean and covariance are different. In particular, as $t \to \infty$, they tend to

$$\mu_\infty^{NTK}(x) = \Theta(x, \vec{x})\Theta^{-1}(\vec{x}, \vec{x})\vec{y}; \tag{2.155}$$

$$K_\infty^{NTK}(x, x') = K(x, x') + \Theta(x, \vec{x})\Theta^{-1}(\vec{x}, \vec{x})K(\vec{x}, \vec{x})\Theta^{-1}(\vec{x}, \vec{x})\Theta(\vec{x}, x') \\ - \left[\Theta(x, \vec{x})\Theta^{-1}(\vec{x}, \vec{x})K(\vec{x}, x') + \Theta(x', \vec{x})\Theta^{-1}(\vec{x}, \vec{x})K(\vec{x}, x)\right]. \tag{2.156}$$

As was shown in Lee et al. (2019), there does not exist an initial covariance matrix (a "prior") such that these mean and covariance correspond to Bayesian posterior given the training data.

The "empirical" counterpart of NNGPs is $\hat{K}(x, x') = \frac{1}{n}h^T(x)h(x')$. Compared to empirical NTKs, empirical NNGPs are easier to compute as they do not require a backward pass. The corresponding memory footprint is also lower for empirical NNGPs as they do not require computing Jacobian matrices that scale as $O(N)$ where N is the number of weights. This makes NNGPs more suitable for large models. As an example, Park et al. (2020) used performance of empirical NNGPs as a proxy measure for neural architecture search. They argue that first, empirical NTKs are too costly to compute, and second, they provide worse learning signal for their task.

NNGP of a generic neural network can be computed in a recursive manner, as was demonstrated in Section 2.4 on the example of fully-connected and convolutional nets: $q_{l+1}(x, x') = \mathbb{E}_{[z,z']^T \sim \mathcal{N}(0, \Sigma_l(x,x'))}\phi(z)\phi(z')$, where $\Sigma_l(x, x') = \begin{pmatrix} q_l(x, x) & q_l(x, x') \\ q_l(x', x) & q_l(x', x') \end{pmatrix}$; the Master theorem of Yang (2019b) gives similar fomulas for a generic neural net. In the above example, there is an operation that maps a kernel $q_l(x, x')$ to a subsequent kernel $q_{l+1}(x, x')$. Shankar et al. (2020) presents an algebra of operations on kernels. While this algebra consists of operations of only three types, it is enough to express NNGP of a fully-connected or a convolutional network with any elementwise nonlinearities.

2.8.2 Label-aware NTK

One of the major problems of kernel methods is *label agnosticism.* Recall that a kernel evaluated at a pair of points is a scalar product of their mappings to some feaure space: $K(x, x') = \langle \Phi(x), \Phi(x') \rangle$. Therefore a kernel measures how similar the two points are, and a kernel method uses this information to derive responses on unseen data: $f(x) = K(x, \vec{x})\vec{\alpha}$. Intuitively, a kernel K should result in a good-generalizing model if $K(x, x')$ is positive when $y = y'$ and negative otherwise. Therefore the "perfect" kernel would be $K^*(x, x') = yy'$; the obvious problem is that it cannot be computed on unseen data.

A kernel that can be computed on unseen data cannot depend on labels. Therefore, if data has several possible labelings, for a pair of data points (x, x'), there could be a labeling with $y = y'$ and a labeling with $y \neq y'$. At the same moment, $K(x, x')$ stays the same on both cases; therefore, the corresponding kernel method cannot generalize well on both of the labelings.

As an example of several possible labelings on a single dataset, consider a dataset of pictures with two objects in each frame, and let the two objects belong to two disjoint sets of classes. Then one of the labelings may consider only the objects of the first classes set, while the other may consider the objects of the second set.

Chen et al. (2020) propose two ways of making a kernel *label-aware.* The first is mixing the kernel at hand with the perfect kernel $K^*(x, x') = yy'$: $K^{HR}(x, x') = (1 - \lambda)K(x, x') + \lambda K^*(x, x')$

for $\lambda \in [0, 1]$. If the perfect kernel was available, the best choice would be to take $\lambda = 1$. Since it is not available, we have to approximate it somehow, therefore making the optimal λ to become less than one.

In order to approximate $K^*(x, x')$, we need a model that maps (x, x') to yy'. Since the training dataset for this model consists $O(m^2)$ samples, and since the model itself has to be evaluated on $O(m)$ samples for each test point x, the model has to be relatively simple. Chen et al. (2020) consider models of the form $Z(x, x') = \vec{y}^T M(x, x', \vec{x})\vec{y}$, where $M \in \mathbb{R}^{m\times m}$. One of the possible choices of M is $M(x, x', \vec{x})_{ij} = \psi(K(x, x'), K(x_i, x_j))$, where $\psi(z_1, z_2)$ measures similarity. As one can see, this choice of Z takes a linear combination of $y_i y_j$ with weights being similarities of $K(x, x')$ and $K(x_i, x_j)$. Intuitively, this reads as "yy' and $y_i y_j$ are similar if $K(x, x')$ and $K(x_i, x_j)$ are close".

While the above proposal can be applied to any kernel K, the second label-aware kernel of Chen et al. (2020) is a specific modification of NTK. Let us recall the construction of Θ^{NTH} resulted from integrating the learning dynamics up to the order n^{-1}, taking the limit of $t \to \infty$, and taking expectation (see Section 2.3 and specifically Eq. (2.52)):

$$\begin{aligned}
\Theta^{NTH}(x_1, x_2) &= O^{(0)}_{2,0}(x_1, x_2) + n^{-1}\mathbb{E}\, O^{(1)}_{2,\infty}(x_1, x_2) \\
&= \Theta(x_1, x_2) + n^{-1}\mathbb{E}\left[O^{(1)}_{2,0}(x_1, x_2)\right] - n^{-1}\mathbb{E}\left[O^{(1)}_{3,0}(x_1, x_2, \vec{x})\Theta^{-1}(\vec{x}, \vec{x})f^{(0)}_0(\vec{x})\right] \\
&\quad + n^{-1}\vec{y}^T\Theta^{-1}(\vec{x}, \vec{x})\mathbb{E}\left[O^{(1)}_{4,0}(x_1, x_2, \vec{x}, \vec{x})\right]\Theta^{-1}(\vec{x}, \vec{x})\vec{y} \\
&\quad + n^{-1}\mathbb{E}\left[f^{(0),T}_0(\vec{x})\Theta^{-1}(\vec{x}, \vec{x})O^{(1)}_{4,0}(x_1, x_2, \vec{x}, \vec{x})\Theta^{-1}(\vec{x}, \vec{x})f^{(0)}_0(\vec{x})\right] \\
&\quad - n^{-1}\sum_{k,l=1}^{m}\frac{1}{\lambda_k(\lambda_k + \lambda_l)}\vec{y}^T\vec{v}_k\vec{v}_k^T\mathbb{E}\left[O^{(1)}_{4,0}(x_1, x_2, \vec{x}, \vec{x})\right]\vec{v}_l\vec{v}_l^T\vec{y} \\
&\quad - n^{-1}\sum_{k,l=1}^{m}\frac{1}{\lambda_k(\lambda_k + \lambda_l)}\mathbb{E}\left[f^{(0),T}_0(\vec{x})\vec{v}_k\vec{v}_k^T O^{(1)}_{4,0}(x_1, x_2, \vec{x}, \vec{x})\vec{v}_l\vec{v}_l^T f^{(0)}_0(\vec{x})\right]. \quad (2.157)
\end{aligned}$$

Since $\hat{\Theta}_0(x_1, x_2) = O^{(0)}_{2,0}(x_1, x_2) + n^{-1}O^{(1)}_{2,0}(x_1, x_2) + O(n^{-2})$, we have $\Theta(x_1, x_2) + n^{-1}\mathbb{E}\left[O^{(1)}_{2,0}(x_1, x_2)\right] = \mathbb{E}\,\hat{\Theta}_0(x_1, x_2) + O(n^{-2})$ and $\Theta(x_1, x_2) = \mathbb{E}\,\hat{\Theta}_0(x_1, x_2) + O(n^{-1})$. For the same reason, $\mathbb{E}\left[O_{4,0}(x_1, x_2, x_3, x_4)\right] = n^{-1}\mathbb{E}\left[O^{(1)}_{4,0}(x_1, x_2, x_3, x_4)\right] + O(n^{-2})$. Suppose $f^{(0)}_0(\vec{x}) = 0$. Given this approximation, up to order $O(n^{-2})$,

$$\begin{aligned}
\Theta^{NTH}(x_1, x_2) \approx \mathbb{E}\,\hat{\Theta}_0(x_1, x_2) &+ \vec{y}^T\left(\mathbb{E}\,\hat{\Theta}_0(\vec{x}, \vec{x})\right)^{-1}\mathbb{E}\left[O_{4,0}(x_1, x_2, \vec{x}, \vec{x})\right]\left(\mathbb{E}\,\hat{\Theta}_0(\vec{x}, \vec{x})\right)^{-1}\vec{y} \\
&- \sum_{k,l=1}^{m}\frac{1}{\lambda_k(\lambda_k + \lambda_l)}\vec{y}^T\vec{v}_k\vec{v}_k^T\mathbb{E}\left[O_{4,0}(x_1, x_2, \vec{x}, \vec{x})\right]\vec{v}_l\vec{v}_l^T\vec{y}. \quad (2.158)
\end{aligned}$$

As one can see, $\Theta^{NTH}(x_1, x_2)$ depends on train labels $\vec{y}$. Roughly speaking, this kernel corresponds to the NTK of a network trained until convergence ($t \to \infty$); obviously, this kernel should depend on training data.

As an interesting observation $\Theta^{NTH}(x_1, x_2) = \mathbb{E}\,\hat{\Theta}_0(x_1, x_2) + \vec{y}^T M(x_1, x_2, \vec{x})\vec{y}$ for a certain matrix M — recall that $K^{(HR)}(x_1, x_2)$ considered previously has a similar form.

Note that computing the Gram matrix $\Theta^{NTH}(\vec{x}, \vec{x})$ requires computing the Gram "matrix" of

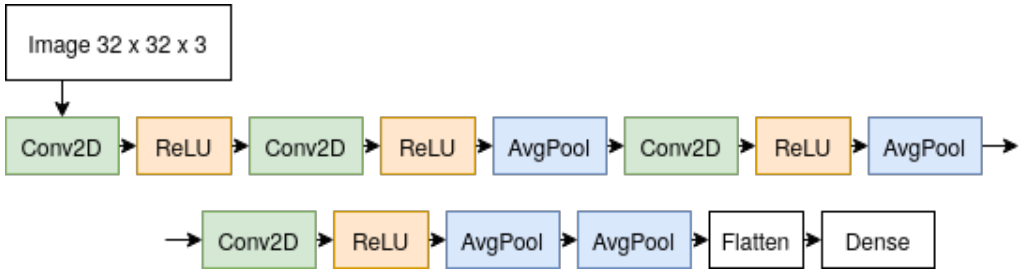


Figure 2.4: Myrtle architecture.

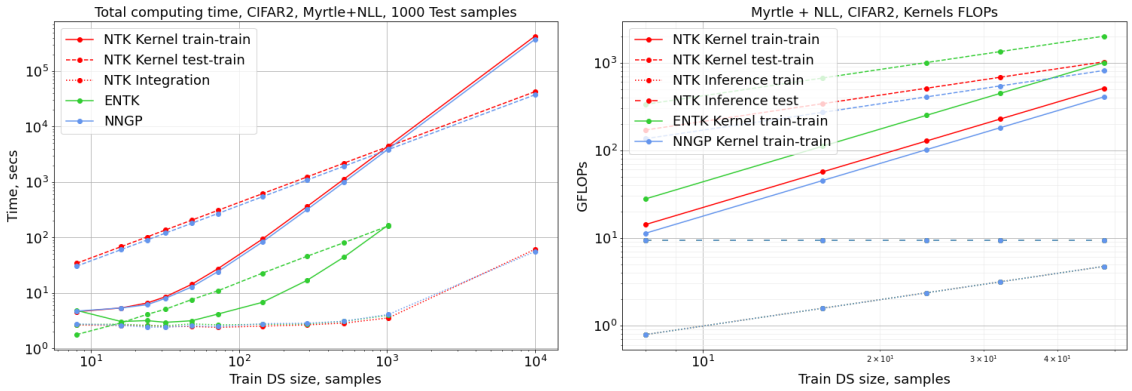


Figure 2.5: Myrtle network trained on subsets of CIFAR2 of different sizes. Different lines refer to different regimes of training (e.g. NTK, NNGP etc.) and different stages of training (e.g. cosntructing the kernel, integrating the dynamics etc.). We use BCE loss, and integrate the dynamics numerically for $T = 10^4$ steps. We measure training time and number of FLOPS.

the expected 4-th order empirical kernel $\mathbb{E}\left[O_{4,0}(\vec{x},\vec{x},\vec{x},\vec{x})\right]$. Instantiating this tensor requires $O(m^4)$ time and $O(m^4)$ memory which is only possible for very small datasets.

2.9 Limits of applicability

In this section, we present a small experimental study on scope of applicability for NTK regression to real-time scenarios. In particular, we would like to investigate first, what is the maximal size m of training dataset of images of given size we can afford with limited computational resources. Second, what is the maximal image resolution d we can afford given fixed dataset size. We restrict ourselves to these two questions since for practical purposes, dependence of NTK regression complexity on these two parameters is the most worrying: it is $O(m^2d^4)$ for constructing the Gram matrix, $O(m^3)$ for integrating the dynamics analytically, and $O(m^2T)$ for integrating the dynamics numerically for T steps; see Section 2.5.

We use NeuralTangents (Novak et al., 2020) and perform all our experiments on a single GTX

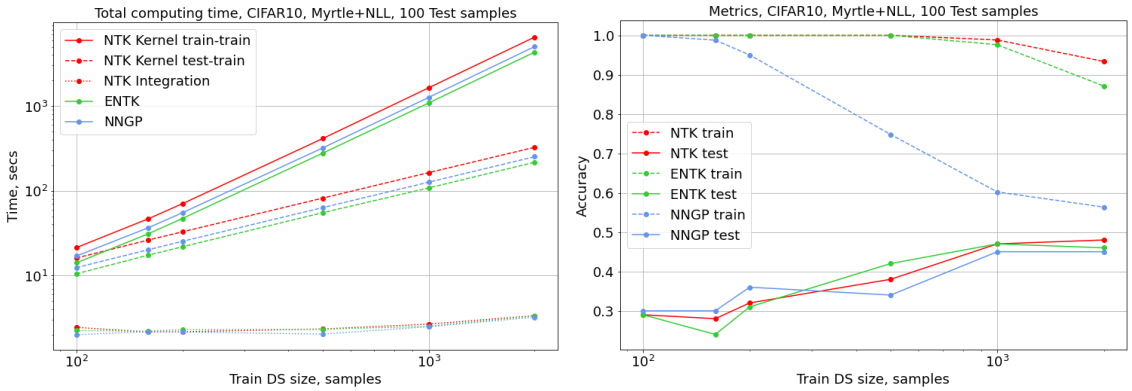


Figure 2.6: Myrtle network trained on subsets of CIFAR10 of different sizes. Different lines refer to different regimes of training (e.g. NTK, NNGP etc.) and different stages of training (e.g. cosntructing the kernel, integrating the dynamics etc.). We use cross-entropy loss, and integrate the dynamics numerically for $T = 10^4$ steps. We measure training time and accuracy.

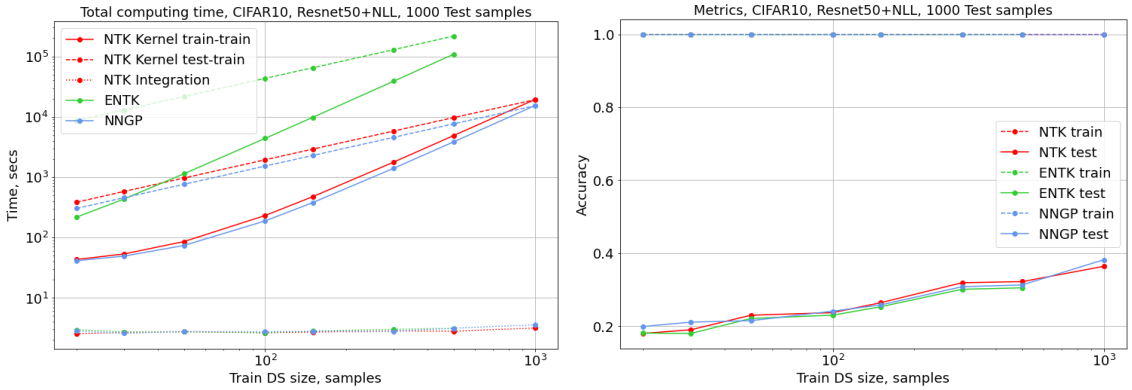


Figure 2.7: Resnet50 trained on subsets of CIFAR10 of different sizes. Different lines refer to different regimes of training (e.g. NTK, NNGP etc.) and different stages of training (e.g. cosntructing the kernel, integrating the dynamics etc.). We use cross-entropy loss, and integrate the dynamics numerically for $T = 10^4$ steps. We measure training time and accuracy.

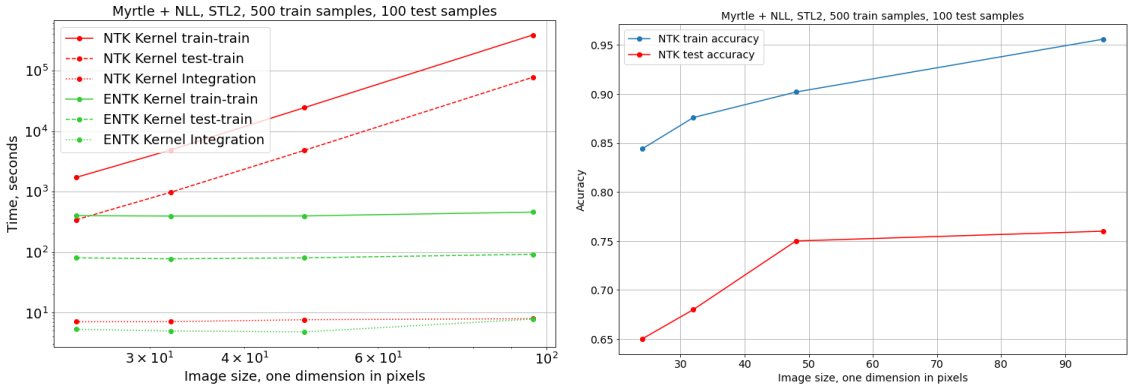


Figure 2.8: Myrtle network trained on a subset of STL2 of size 500 with images of different resolutions. Different lines refer to different regimes of training (e.g. NTK, NNGP etc.) and different stages of training (e.g. cosntructing the kernel, integrating the dynamics etc.). We use BCE loss, and integrate the dynamics numerically for $T = 10^4$ steps. We measure training time and accuracy.

1080Ti GPU with 12 GiB of memory. We consider a Myrtle network[4] with 64 channels in all convolutional layers, see Fig. 2.4. We pick this architecture because it is lightweight and uses only those layers for which NTK can be computed analytically.

For the first experiment, we consider two classes of CIFAR10 and refer this dataset as CIFAR2. We pick a subset of 1000 samples of the original test set of CIFAR2 and vary the size of the training subset. We optimize binary cross-entropy (BCE) and integrate the dynamics numerically for $T = 10^4$. We compute the Gram matrix of a kernel using batch size 4. On Fig. 2.5, we plot training time and the number of floating-point operations (FLOPS) for different stages (i.e. Gram matrix computation, integrating the dynamics, inference on a test set) and for different regimes of training (analytical NTK, analytical NNGP, and empirical NTK) versus size of training dataset. As one can see, already for relatively small datasets ($m = 10^4$), the most time-demanding stage is construction of the Gram matrix $\Theta(\vec{x}, \vec{x})$ (solid line), but not integration (which is also takes time quadratic to size of the dataset) (dotted line). Also, the time to compute the NNGP kernel is almost the same as the one for NTK, since both are computed analytically; see Section 2.4. We could not obtain the point $m = 10^4$ for empirical NTK (ENTK) due to numerical reasons. If we extrapolate the solid line to $m = 10^6$, the size of ImageNet, noting the quadratic growth, we will get 5×10^9 seconds, which is around 160 years of computations. While our time measurements are device-dependent, we also measure the number of FLOPS, which while being device-independent, grows the same way as time and is also quite large. This experiment demonstrates that indeed, the naive approach for integrating the NTK dynamics falls short on datasets of realistic sizes, thus striving for major optimizations. As mentioned in Section 2.5, a promising approach could be the one of Meanti et al. (2020).

On Fig. 2.6, we present the same experiment but with all 10 classes of CIFAR10. We observe the same quadratic time growth issue for all three regimes of training (analytical NTK, analytical NNGP, and empirical NTK). We also report accuracy for comparison with previous works on small data training with kernel methods (i.e. Arora et al. (2020)).

In addition to experiments with a small network, we experimented with a variant of Resnet50

[4] https://myrtle.ai/how-to-train-your-resnet-4-architecture/

(He et al., 2016a). We modify this architecture by removing batch normalizations and substituting max poolings with average poolings, so to make analytical computations possible. Results are shown on Fig. 2.7. Doing the same extrapolation to ImageNet size, we get 6.25×10^{11} seconds, which is around 20000 years.

Lastly, we consider two classes of STL10 and similarly to CIFAR2, refer this dataset as STL2. We pick a subset of 100 samples of the original test set of STL2 and 500 samples of its original train set. While STL10 has fewer labeled examples compared to CIFAR10, it has larger images: 96×96 for STL10 versus 32×32 for CIFAR10. We vary size of the input image and measure training time and accuracy, similarly to the first experiment. As before, we optimize binary cross-entropy (BCE) and integrate the dynamics numerically for $T = 10^4$. However, we use batch size 1 for computing the Gram matrix, since larger batch sizes do not fit in GPU memory for large image sizes. Results are shown on Fig. 2.8. As before, the most time-demanding part is kernel Gram matrix computation (blue line): it grows as $O(d^4)$, where d is image resolution; see Section 2.4. If we extrapolate this line to $d = 224$, the resolution on which traditional ImageNet classification models operate, we will get around 150 days of computations. This experiment therefore demonstrates that not only dataset size, but also image resolution complexity can also be a serious bottleneck in applying NTK approach in practice. Also, while for dataset size, certain optimizations are available (e.g. Meanti et al. (2020)), we are not aware of any optimizations aiming for decreasing image resolution complexity.

2.10 Conclusions

The use of NTK theory is twofold: first, it relates neural networks to kernel methods, a far more well-developped class of models. Second, it gives a machine learning practitioner a kernel that shares some properties with neural nets.

Recall what we have concerning the first application. We have a theorem (Theorem 2.2.2) that implies that a neural tangent kernel of a wide class of architectures is deterministic and does not evolve with time in the limit of infinite width, and provides a recurrent formula for the limit. Therefore a network that is wide enough should share some properties, i.e. convergence and generalization, see Section 2.6.3, with the corresponding kernel method. However, the resulting width bounds are far from realistic. Second, the limit kernel does not evolve with time only under certain non-standard parameterization rarely used in practice. In contrast, standard parameterization results in evolving (normalized) kernel, see Section 2.7. The fact that the kernel evolves may be the key to understanding superior performance of neural nets to kernel methods. Unfortunately, we have little understanding of this aspects at the moment. Lastly, Theorem 2.2.2 requires Gaussian weight initialization rarely used in practice. Generalizing it to non-Gaussian weight distribution remains to be done in the future.

Let us discuss the second application. At the moment of writing, computing the exact limit kernel was available only for convolutional and fully-connected networks with average poolings and nonlinearities in a certain class, see Section 2.5. For other architectures, one has to rely on empirical NTK which is a biased estimate of the limit one. Computing the empirical NTK requires instantiating output-by-weight jacobians at every pair of training points, which is especially memory risky for realistically large architectures. Storing the Gram matrix of the kernel also requires $O(m^2)$ memory where m is dataset size. Even if the kernel is sucessfully computed on every pair of training points, integrating the training dynamics naively requires inverting the Gram matrix, which costs $O(m^3)$ time, while for datasets of size 10^6 one can barely afford more than $O(m)$ time and memory.

We study applicability limits of this naive approach in Section 2.9. Still, certain optimization are available, see Section 2.5.

Also concerning the second application, NTK is not the only kernel that can be constructed using a neural network; certain other kernels may have computational or performance gains compared to NTK, see Section 2.8.

3 Non-Gaussian Tensor Programs

Authors: Eugene Golikov, Greg Yang.

Link and reference: Golikov and Yang (2022), published at *Neural Information Processing Systems (NeurIPS)*[1].

Contribution: most of the proofs, a sufficient part of the final text, as well as experiments, were contributed by the author of the present thesis.

3.1 Introduction

Universality in Neural Network Initialization Common initialization schemes of neural networks (e.g., Glorot and Bengio (2010); He et al. (2015)) define specific ways of scaling initial layer weights with layer sizes. However, in general, they do not express any preference on the initial weight distribution beyond iid sampling. While some works (Hinton et al., 2012; He et al., 2016a) use a Gaussian distribution, other practitioners advocate for a uniform one since it is possible to sample very large weights from a Gaussian, causing numerical instabilities.

However, a theorist, especially with a background on physics or probability theory, would suspect that the precise choice of distribution does not matter as the neural network's width tends to infinity. This is a belief in the general concept of *universality* — large systems display a consistent behavior at a macroscopic scale regardless of the microscopic details.[2]

This consideration has led deep learning practitioners to treat the distribution of random initialization as a matter of personal choice[3] However, as we shall see, the reality appears to be more subtle: roughly speaking, distribution universality holds for hidden weights but does not hold for other parameters like input and output weights. We formulate the precise *universality principle* (Principle 2) for neural network random intialization in Section 3.2. We do so in plain English, but deducing this principle requires some fundamental advancements in the theory of *Tensor Programs.* Below, we briefly review this theory before describing our contributions to it.

[1] https://proceedings.neurips.cc/paper_files/paper/2022/hash/8707924df5e207fa496f729f49069446-Abstract-Conference.html

[2] The simplest example of universality: taking the average of many iid copies of a random variable always yield its mean, regardless of whether the variable is Gaussian, uniform, Laplace, etc.

[3] See e.g. a discussion in https://github.com/fchollet/keras/issues/52.

Tensor Programs Just like autograd (Rumelhart et al., 1986) empirically automates the calculation of chain rule of arbitrary computation graphs, Tensor Programs (TP) (Yang, 2019a) has automated the theoretical calculation of infinite-width limits of the same (where width of a computation graph corresponds to the size of matrices). What previously were difficult limits to calculate, now becomes routine via TP. For example, Neal (1995) in 1994 showed that randomly initialized wide shallow neural networks are Gaussian Processes (which is called the Neural Network-Gaussian Process Correspondence, or NNGP Correspondence), but only recently this has been extended to deep perceptrons (Lee et al., 2018; Matthews et al., 2018) and more advanced architectures such as convolutional neural networks (Garriga-Alonso et al., 2019; Novak et al., 2019), and each such extension requires painstaking calculations and careful applications of Law of Large Numbers and Central Limit Theorem. But with Tensor Programs, one can show that NNGP Correspondence holds for any architecture all at once (Yang, 2019b). Similarly, in a certain parametrization, a wide multi-layer perceptron (MLP) evolves like a linear model during training (Jacot et al., 2018), but showing this for advanced architectures was very difficult. TP (Yang, 2020a; Yang and Littwin, 2021) again was able to prove this behavior for any architecture. Finally, TP gave rise to the Dynamical Dichotomy Theorem (Yang and Hu, 2021), a classification of all natural infinite-width limits of neural networks, and led to the discovery of Maximal Update Parametrization, or μP. These results underlie the hyperparameter transfer technology that for the first time enabled the hyperparameter tuning of enormous neural networks too expensive to train more than once (Yang et al., 2021).

Distributional Universality However, these results were only proven for computational graphs whose matrices and vectors are random Gaussians (which, in the special case of graphs pertaining to neural network training, means random Gaussian initialization of NN). What happens when they take more general distributions?

In this work, we show that any program with non-Gaussian matrices and vectors also have infinite-width limits under mild conditions, and thus recovering all of the aforementioned results for non-Gaussian objects automatically (see Section 3.4). In fact, they often coincide with the limits of Gaussian samplings: we shall formulate a general *distributional universality principle* for Tensor Programs (Principle 3), from which the universality principle (Principle 2) for NN random intialization (mentioned in the beginning of this section) follows as a special case.

For those familiar with the language of Tensor Programs, the principle can be summarized simply: all programs have the same limit if their Gaussian matrices are swapped out for non-Gaussian matrices with the same variances; however, this is not true in general for their initial vectors.

Applications to Random Matrix Theory The universality principle also makes the proof of the Semicircle and Marchenko-Pastur Laws in Yang (2020b) automatically valid for non-Gaussian random matrix ensembles. Likewise, it shows that the asymptotic singular value distribution of the input-output Jacobian of a random neural network does not depend on the distribution of random initialization beyond its variance.

Contributions In summary:

- We clarify where initialization distribution matters and where it does not in wide neural networks by formulating a precise *universality principle* (Principle 2).
- More generally, we develop the theory of non-Gaussian Tensor Programs, as well as stating the corresponding *universality principle* for Tensor Programs (Principle 3).

- We apply this general theory to obtain previous results of the Tensor Programs papers, such as NNGP and NTK correspondence, for non-Gaussian weight initializations (Section 3.4).

3.2 Distributional Universality in Words

In this section, we state, at a level understandable to practitioners, where the sampling distribution in random initialization matters (beyond the first two moments). We begin with the simple example of MLPs.

Motivating Example: MLP Let us first consider the case of a simple biasless multilayer perceptron (MLP) $f(\xi)$ with L hidden layers and a nonlinearity ϕ:

$$f(\xi) = W^{L+1}\phi(W^L\phi(\cdots\phi(W^1\xi)\cdots)). \tag{3.1}$$

Here $W^2, \ldots, W^L \in \mathbb{R}^{n\times n}$, $W^1 \in \mathbb{R}^{n\times d_{\text{in}}}$ and $W^{L+1} \in \mathbb{R}^{d_{\text{out}}\times n}$, where d_{in} (resp. d_{out}) is the input (resp. output) dimension, and we call n the *width* of f. Then we can formulate the following *universality* principle:

Principle 1 (Universality in MLP Initialization)**.** *As width tends to infinity, two different iid random initializations of a biasless multilayer perceptron (MLP) induce* identical training behavior *as long as 1) they sample input and output weights the same way, and 2) they sample hidden weights with the same mean and variance.*[4]

Here "identical training behavior" means that for *any* sequence of batches of data, performing SGD from either random initialization yields the same function after any number of training steps.[5]

The key point in Principle 1 is that hidden weights are not sensitive to the exact distribution but the input and output weights are. We can demonstrate the sensitivity of the input and output weights on a simple example. Consider the 1-hidden-layer (i.e. $L = 1$) version of Eq. (3.1) with $d_{\text{in}} = d_{\text{out}} = 1$ and with input and output weights tied, which we name $U = W^1 = W^{2\top} \in \mathbb{R}^n$ (where again n is width):

$$f(\xi) = \frac{1}{n}U^\top\phi(U\xi). \tag{3.2}$$

Here, the additional $\frac{1}{n}$ factor compared to Eq. (3.1)[6] is just a normalization so that the network output will not blow up to infinity as width n becomes large; it's convenient but not essential for what we will discuss. Consider two alternatives (G) and (R) for sampling U:

$$\text{(R)}\ \ U_\alpha = \pm 1 \text{ with prob. } 1/2 \quad \text{or} \quad \text{(G)}\ \ U_\alpha \sim \mathcal{N}(0,1).$$

Both methods have variance 1. Now suppose the nonlinearity $\phi(x)$ equals to $x\mathbb{I}[x \in -1/2, 1/2]$ Then

$$f(1) = 0 \text{ with init (R)} \quad \text{but} \quad f(1) \to \mathbb{E}_z z\phi(z) > 0 \text{ with init (G)}$$

[4] "same" means between the two initialization methods; on the other hand, different weight entries can have different distributions.

[5] This can be made even more general in the context of Tensor Programs; see (Yang and Littwin, 2021; Yang and Hu, 2021).

[6] This results in the so-called mean field parametrization (Rotskoff and Vanden-Eijnden, 2018; Chizat and Bach, 2018; Sirignano and Spiliopoulos, 2020), which is a special case of the maximal update parametrization (Yang and Hu, 2021).

as $n \to \infty$, where $z \sim \mathcal{N}(0,1)$. Thus (G) and (R) definitely do not induce identical training behaviors — they are not even identical at initialization!

This example can be generalized to deep biasless MLPs to show that input and output weights are sensitive to the sampling distribution. Conversely, from our main theorem (Theorem 3.3.7) below, it will also be clear that hidden weights are insensitive to the sampling distribution.

What Principle 1 Gets Wrong in General Architectures Unfortunately, Principle 1 is not correct if we go beyond biasless MLP. Indeed, if we just add bias to such an MLP, then one can easily generalize the example of Principle 1 to show that biases are also sensitive to their exact sampling distribution. As the architecture becomes complex, it becomes difficult to figure out whether a particular parameter tensor is sensitive or not in an ad hoc fashion (e.g., layernorm weights and biases? Self-attention weights? etc).

Principle for General Case Fortunately, there is a simple rule to tell which parameter tensors are sensitive based on the following.

Definition 3.2.1 (Yang et al. (2021))**.** Let P be a parameter tensor in a neural network of any architecture. As width becomes large, if P's size remains constant, then we say P is *scalar-like*; if exactly one dimension of P becomes large, we say P is *vector-like*; if exactly two dimensions of P become large, we say P is *matrix-like.*[7]

Example 3.2.2. In Eq. (3.1), $W^2, \dots, W^L$ are all matrix-like while W^1 and W^{L+1} are vector-like because d_{in} and d_{out} are fixed even as n varies.[8] If we add biases,

$$f(\xi) = b^{L+1} + W^{L+1}\phi(b^L + W^L\phi(\cdots\phi(b^1 + W^1\xi)\cdots)), \tag{3.3}$$

then $b^{L+1} \in \mathbb{R}^{d_{\text{out}}}$ is scalar-like while $b^1, \dots, b^L \in \mathbb{R}^n$ are vector-like.

Example 3.2.3. Some more advanced examples for practitioners, that may be skipped on first reading: If f in Eq. (3.3) is convolutional (instead of dense) then the same categorization (of W^l, b^l into scalar-, vector-, and matrix-like) holds because the kernel size of convolutions is constant as width increases: 1) $W^2, \dots, W^L$ are matrix-like, 2) $W^1, W^{L+1}, b^1, \dots, b^L$ are vector-like, and 3) b^{L+1} is scalar-like. Layernorm weights and biases are vector-like if the input to that layernorm has exactly one hidden dimension (which is almost always the case in practice). Self-attention weights W^k, W^q, W^v are matrix-like if d_{head} is fixed as d_{model} and n_{head} increases or if n_{head} is fixed as d_{model} and d_{head} increases (one of which happens almost always in practice).

With this concept of scalar-, vector-, and matrix-like tensors in mind, we can formulate the general universality principle for neural network random initialization.

Principle 2 (Universality in General Neural Network Initialization)**.** *As width becomes large, two different iid random initializations of a neural network of any architecture* induce identical training behavior[9] *as long as 1) they sample scalar- and vector-like parameters the same way, and 2) they sample matrix-like parameters with the same mean and variance.*

[7] We can further define *tensor-like* and so on, but practically speaking, all relevant large neural networks (e.g., BERT, GPT-3, etc (Devlin et al., 2018; Brown et al., 2020)) will have at most matrix-like parameters due to storage and computation costs of "tensor-like" parameters.

[8] Think of d_{in} and d_{out} as being fixed by the dataset, while you, as a model builder, can freely vary n.

[9] See the discussion below Principle 1 for the meaning of *identical training behavior*.

Remark 3.2.4. Note both Principle 1 and Principle 2 only give *sufficient* conditions for identical training behavior for *all possible datasets and batches*. But practically, when we only focus on specific datasets and specific training procedures at hand, it is possible that there could be weaker conditions that would ensure identical training behavior in such specific settings. For example, if the input dimension of a dataset is large and each example has entries that look iid then the input layer is not sensitive to the exact sampling distribution (aside from the first two moments) either. Nevertheless, Principle 1 and Principle 2 yield guidelines that are generally applicable, upon which we may refine our reasoning, if we wish, based on individual specifics.

3.3 Tensor Programs: Main Result

Colloquially, a Tensor Program is just a computation interleaving matrix multiplication and coordinatewise nonlinearities. In prior works, there have been several different formalizations of this concept. Here, we simplify and take the following definition.

Definition 3.3.1. Given matrices $A^1, \dots, A^L \in \mathbb{R}^{n\times n}$, initial vectors $g^1, \dots, g^{M_0} \in \mathbb{R}^n$, and initial scalars $c^1, \dots, c^{M_0} \in \mathbb{R}$, consider the following iteration for $i = M_0 + 1, \dots, M$ that generates new vectors $g^{M_0+1}, \dots, g^M \in \mathbb{R}^n$ and scalars $c^{M_0+1}, \dots, c^M \in \mathbb{R}$:

$$g^i_\alpha \leftarrow \sum_{\beta=1}^n W^i_{\alpha\beta} x^i_\beta, \quad c^i \leftarrow \frac{1}{n}\sum_{\beta=1}^n x^i_\beta, \quad \text{where } x^i_\alpha = \phi^i(g^1_\alpha, \dots, g^{i-1}_\alpha; c^1, \dots, c^{i-1}). \tag{3.4}$$

Here each ϕ^i is a chosen scalar function with $(i-1)+(i-1)$ arguments and W^i is an $n \times n$ matrix. Each matrix W^i equals to either some matrix A^j of the program or its transpose $A^{j\top}$. The matrices W^i for different i can possibly be the same. In this work, we shall call any computation of this form a *Tensor Program (TP)*, or just a *program* for short. Thus each program is entirely determined by the data $\{A^j\}_{j=1}^L \cup \{g^i\}_{i=1}^{M_0} \cup \{c^i\}_{i=1}^{M_0} \cup \{\phi^i\}_{i=M_0+1}^M$ along with the correspondence between W^i and A^j or $A^{j\top}$.

This formulation of a Tensor Program is equivalent to NETSOR$\top^+$ in Yang (2020b), as shown in Appendix A.4. As such, Eq. (3.4) can express any computation expressible in a DL framework such as PyTorch (Paszke et al., 2019), including gradient descent iterations of neural networks of any architecture, e.g. (Yang and Hu, 2021; Yang and Littwin, 2021). This expressivity allows one to treat a wide range of problems uniformly using just Theorem 3.3.4 below.

Example program For example, consider the first forward pass of a simple MLP with scalar input ξ and output $f(\xi)$:

$$f(\xi) = V^\top \sigma(W\sigma(\xi U)), \tag{3.5}$$

where $\xi \in \mathbb{R}; U, V \in \mathbb{R}^n; W \in \mathbb{R}^{n\times n}$, and σ is an activation function. We can express this in a TP as follows: $g^1 \leftarrow U, g^2 \leftarrow nV$ are the initial vectors and $c^1 \leftarrow \xi, c^2 \leftarrow 1$ are initial scalars (where c^2 will just be ignored). $A^1 \leftarrow W$ is the sole matrix of the program. Then the program computes

$$g^3 \leftarrow W^3 x^3, \quad \text{where } W^3 \leftarrow A^1 = W \text{ and } x^3_\alpha = \phi^3(g^1_\alpha, g^2_\alpha; c^1, c^2) \leftarrow \sigma(\xi U_\alpha)$$
$$f(\xi) = c^4 \leftarrow \frac{1}{n}\sum_{\beta=1}^n x^4_\beta, \quad \text{where } x^4_\alpha = \phi^4(g^1_\alpha, g^2_\alpha, g^3_\alpha; c^1, c^2, c^3) \leftarrow (nV)_\alpha \sigma(g^3_\alpha)$$

where c^3 and g^4 are implicitly computed but ignored.[10] Extending this first step, the entire training process can further be written in a TP, where the learned function outputs are expressed as scalars. See Yang (2019b, 2020a); Yang and Littwin (2021); Yang and Hu (2021) for more examples.

Gaussian Tensor Programs The results achieved by the TP framework Yang (2019a, 2020b); Yang and Littwin (2021) so far most commonly spring from the following version of the so-called *Master Theorem*:

Theorem 3.3.2 (Gaussian Master Theorem, original formulation of Yang (2020b))**.** *Consider Setup 3.3.3 below. Then, as $n \to \infty$, for any pseudo-Lipschitz ψ,*

$$\frac{1}{n}\sum_{\alpha=1}^{n} \psi\left(g_\alpha^1, \ldots, g_\alpha^M, c^1, \ldots, c^M\right) \xrightarrow{a.s.} \mathring{\Psi}, \tag{3.6}$$

where $\mathring{\Psi}$ is a deterministic scalar given by a certain recurrent formula.

Setup 3.3.3 (Gaussian Tensor Programs)**.** *Consider a Tensor Program with M vectors $g^1, \ldots, g^M \in \mathbb{R}^n$ and scalars $c^1, \ldots, c^M$. Suppose 1) all initial vectors $g^1, \ldots, g^{M_0}$ have standard Gaussian entries[11];2) all initial scalars $c^1, \ldots, c^{M_0}$ have almost sure limits as $n \to \infty$; 3) all matrices A^i have iid entries from $\mathcal{N}(0, n^{-1})$; 4) all the nonlinearities ϕ^i are pseudo-Lipschitz.*[12]

For example, Theorem 3.3.4 implies that Eq. (3.5)'s function values $f(\xi)$ *after training* converge to deterministic values in various infinite-width limits, in particular, the feature learning limit (Yang and Hu, 2021; Yang et al., 2021).

We can reformulate the above theorem in a shorter form:

Theorem 3.3.4 (Gaussian Master Theorem, equivalent formulation)**.** *Consider Setup 3.3.3. Then, as $n \to \infty$, every scalar c^i converges almost surely to a deterministic limit $\mathring{c}^i$ which can be computed via a recurrent formula.*

It is easy to see that the above statement is equivalent to the original Theorem 3.3.2. Indeed, any scalar in the program has the form $\frac{1}{n}\sum_\alpha \psi(g_\alpha^1, \ldots, g_\alpha^M; c_\alpha^1, \ldots, c_\alpha^M)$ for some function ψ, while any expression of the above form can be thought as a scalar in a new program. We are going to build upon the formulation of Theorem 3.3.4 since it introduces fewer entities.

Non-Gaussian Tensor Programs We are going to generalize Theorem 3.3.4 to non-Gaussian distributions. But before we do so rigorously, we first formulate an easily statable principle that summarizes our results in an intuitive way.

Principle 3 (Universality in Tensor Program Sampling)**.** *For simplicity, consider TP without initial scalars. As $n \to \infty$, two different iid random samplings of a TP's matrices and initial vectors result in identical limits of scalars as long as 1) they sample all initial vectors the same way, and 2) they sample all matrix entries with the same variance (and zero mean).*

[10] We intentionally simplified the formulation of TP to the form Eq. (3.4) in particular to simplify the proofs, but at the cost that expressing common computation encounters some redundancy as shown in this example.

[11] In the original formulation of Yang (2020b), initial vectors were assumed to be sampled as $(g_\alpha^1, \ldots, g_\alpha^{M_0}) \sim \mathcal{N}(\mu^{in}, \Sigma^{in})$ iid over $\alpha = 1, \ldots, n$ for some $\mu^{in} \in \mathbb{R}^{M_0}, \Sigma^{in} \in \mathbb{R}^{M_0 \times M_0}$. However, one can always construct such vectors from ones with iid standard Gaussian entries using a linear elementwise map. This map can be further absorbed into subsequent nonlinear maps in the program.

[12] A function $f : \mathbb{R}^k \to \mathbb{R}$ is called *pseudo-Lipschitz* if $|f(x) - f(y)| \le C\|x - y\|(1 + \sum_{i=1}^k |x_i|^d + |y_i|^d)$ for some $C, d > 0$.

More generally, for programs with initial scalars, we just require the two samplings to have the same almost sure deterministic limits for them.[13]

Principle 1 and Principle 2 are special cases of Principle 3, since any neural network can be expressed as a TP. Principle 3 itself will follow straightforwardly from Theorem 3.3.7.

Now, let us setup our discussion of our main theorem, Theorem 3.3.7.

Definition 3.3.5. We say $f : \mathbb{R}^k \to \mathbb{R}$ is *polynomially smooth* if it is C^∞ and its partial derivatives of any order are polynomially bounded, i.e. for any sequence $(P_1, \dots, P_r) \in [k]^r$, we have $\left|\frac{\partial^r}{\partial x^{P_1} \dots \partial x^{P_r}} f\right| \le C(1 + |x_1|^p + \dots + |x_k|^p)$ for some $C, p > 0$ that may depend on $(P_1, \dots, P_r)$.

Setup 3.3.6. *Consider Setup 3.3.3, but replace 3) and 4) with the following: 3*) there exists a sequence $\nu_3, \nu_4, \dots > 0$ such that all matrices A^i have independent entries drawn from distributions with zero mean, variance n^{-1}, and all higher kth moment bounded by $\nu_k n^{-k/2}$; 4*) all the nonlinearities ϕ^i are polynomially smooth. We further require 5) all moments of initial scalars $c^1, \dots, c^{M_0}$ to exist.*

Many widely-used nonlinearities, such as ReLU or MaxPool, are not polynomially smooth, thus contradicting 4*). Nevertheless, if the original nonlinearity is continuous and polynomially bounded (in particular, if it is pseudo-Lipschitz), one may apply Gaussian smoothing and get a polynomially smooth nonlinearity; see Appendix A.1. The narrower the smoothing kernel, the better the approximation.

The most relevant specific scenario satisfying 3*) is if $A^i_{\alpha\beta} \sim \frac{1}{\sqrt{n}}\mathcal{D}$, where $\mathcal{D}$ is a fixed distribution, like uniform or truncated Gaussian with mean 0 and variance 1. In this case, ν_k can just be taken to be the kth moment of $\mathcal{D}$. But in general 3*) allows entries of A^i to come from different distributions.

Our main result is the following:

Theorem 3.3.7 (Non-Gaussian Master Theorem)**.** *Consider Setup 3.3.6. Then, as $n \to \infty$, every scalar c^i converges to the same $\mathring{c}^i$ as in Theorem 3.3.4 almost surely and in L^p for every $p \in [1, \infty)$:*

$$c^i \xrightarrow{\text{a.s. \& } L^p} \mathring{c}^i, \quad \forall p \in [1, \infty).$$

In short, Theorem 3.3.7 relaxes matrix sampling to be non-Gaussian and non-identically-distributed in general, at the cost of requiring a) more smoothness in nonlinearities and b) all moments of initial scalars to exist. In past applications of TP, b) has always been satisfied but a) has not for relu networks. On the other hand, we also gain L^p convergence compared to Theorem 3.3.4.[14] Note that Setup 3.3.6 still requires initial vectors to be Gaussian, but this is not essential, as we discuss in Section 3.4.

On Tensor Programs with variable dimensions. While Setup 3.3.3 and 3.3.6 assume all hidden dimensions to be equal, Theorem 3.3.4 holds also for Tensor Programs with variable dimensions, see e.g. Yang (2020b). Since our proof technique is based on interpolation between Gaussian and non-Gaussian weights, it can be straightforwardly extended to variable dimensions, as long as Theorem 3.3.4 holds in this setting.

[13] We can, in fact, allow the initial scalars to have non-deterministic limits, in which case the non-initial scalars will have non-deterministic limits as well.In this case, we want the two samplings to have identical limit distributions for the joint distributions of all initial scalars.

[14] We believe L^p convergence can be established without requiring so much smoothness in nonlinearities, but leave this to future work.

3.4 Applications

As mentioned in the introduction, the Tensor Programs series of papers so far has proven a wide range of results, which typically have the characteristic of *architectural universality*, i.e. covering most existing neural architectures. But all of such results have assumed Gaussian weight initialization. Now, armed with Theorem 3.3.7, we show that the same results hold with *non-Gaussian* weight initialization as well under mild assumptions, thus extending them to other prevalent initializations such as uniform and truncated Gaussian. Theorem 3.3.7 here acts like a drop-in replacement for Theorem 3.3.4 in their proofs, except we need to add more smoothness assumptions on nonlinearities.

In general, coarse, qualitative statements, such as "wide neural networks at initialization are Gaussian processes," still hold as stated. But note the exact quantitative statement, in this example regarding the kernel of the Gaussian process, can change as one changes the sampling distribution of vector-like and scalar-like parameters, as indicated by Principle 2 and Principle 3

Definition 3.4.1. We shall call a scalar random variable a *Gaussian image* if it is an image of a standard Gaussian variable under a polynomially smooth function.

Below, we express a rather abstract notion of a neural network that may seem bewildering to a reader who has not read previous papers in the Tensor Programs series. It helps to keep in mind specific examples, for example, the MLP with biases from Section 3.2, when reading the below statements. This notion generalizes all typical neural architectures (as shown in Yang (2019b)) whose matrix-like parameters are randomly initialized with mean 0 and variance $\Theta(1/n)$ (under mild moment conditions of Setup 3.3.6) and whose scalar- and vector-like parameters are randomly initialized from a Gaussian image.

Setup 3.4.2. *Consider a neural network $f(\xi) = \frac{1}{\sqrt{n}} V\Phi(\xi)$ with output layer weights V and embedding $\Phi(\xi)$ on input ξ, and where n is the dimension of the Tensor Program we describe next. Suppose 1) $\Phi(\xi)$ can be expressed as a concatenation of a constant (wrt n) number of vectors $x^i \in \mathbb{R}^n$ (c.f. Eq. (3.4)) from a Tensor Program T in Setup 3.3.6, and 2) V's entries are initialized iid from a Gaussian image with mean 0 and variance 1 and are independent from the random objects of the program T.*

NNGP Correspondence. The result below follows from Theorem 3.3.7:

Corollary 3.4.3. *Any neural network function described in Setup 3.4.2* [15] *converges in finite-dimensional distribution to a Gaussian process in the limit of $n \to \infty$.*[16]

In Appendix A.3, we prove a similar result without assuming the activation functions to be polynomially smooth (we require Lipschitzness instead), see Corollary A.3.3. This result will follow from our non-Gaussian Lipschitz Master Theorem, Theorem A.3.2. The proof of the above corollary follows the same lines as the proof of Corollary A.3.3 given in Appendix A.3[17].

[15] As we can notice from analyzing the proof of Corollary A.3.3, we do not need the output weights V to be Gaussian images. Instead, we could assume that they are iid with zero mean, unit variance, and all higher moments existing. However, we are not able to prove Corollary 3.4.4 below without assuming the entries V to be Gaussian images. We assumed a weaker setup since we wanted to have the same setup for both corollaries.

[16] Notice we allow a weight matrix and its transpose be both involved in the forward pass of the network, in contrast to Yang (2019b), but in line with a more general theorem in Yang (2020b).

[17] There is, however, a small subtlety when applying the proof technique of Corollary A.3.3 to Corollary 3.4.3. There

As discussed above, the kernel of this process in general will be affected by the distribution of vector-like parameters. For example, take $f(\xi) = 1/\sqrt{n} V\phi(U\xi)$ in Setup 3.4.2 where $\xi \in \mathbb{R}$ and ϕ is the indicator function of the interval $[-1/2, 1/2]$. Then sampling U_α as ± 1 with equal probability implies that f is identically 0 while sampling U_α from a standard Gaussian implies f converges to a nontrivial Gaussian process.

NTK Correspondence. We can directly plug our Theorem 3.3.7 into the proof of Yang and Littwin (2021) and obtain

Corollary 3.4.4. *Consider Setup 3.4.2 and assume the loss function is continuously differentiable in the network output. Then under NTK parametrization and SGD weight updates*[18]*, in the limit of $n \to \infty$, 1) the NTK of the network at any optimization step converges pointwise almost surely to a finite deterministic limit $\mathring{\Theta}$ that does not depend on the timestep, and 2) moreover, the network function evolves according to kernel gradient descent with kernel $\mathring{\Theta}$.*

As in the NNGP case, the NTK's infinite-width limit can also be affected by the distribution of input and output weights. The same example there illustrates this: the network's limit NTK as ϕ is equal to the kernel of the limit Gaussian process because ϕ has 0 derivative almost everywhere.

We empirically validate Corollaries 3.4.3 and 3.4.4 in Appendix A.13.

While our non-Gaussian Lipschitz Master Theorem, Theorem A.3.2, mentioned above allows us to generalize Corollary 3.4.3 to ReLU nets, it does not allow us to generalize Corollary 3.4.4 in the same setup (because a Tensor Program expressing the backward pass of a ReLU net has ReLU derivatives, which are not even continuous, as nonlinearities).

Random Matrix Theory. Our Theorem 3.3.7 implies the semi-circle law for non-Gaussian Wigner matrices, the Marchenko-Pastur law for $AA^\top$, where A is non-Gaussian, and Free Independence Principle (FIP) for Tensor Programs with non-Gaussian initial weights, thus generalizing TP3 Yang (2020b); we discuss these results in Appendix A.2. Moreover, since our Theorem 3.3.7 guarantees convergence in mean, we were able to state FIP without assuming linearly bounded nonlinearities as in Yang (2020b).

Classification of Infinite-Width Limits. Consider now an L-hidden-layer biasless perceptron with width n trained using stochastic gradient descent (SGD). As in (Yang and Hu, 2021, Sec 3.2), we shall assume in this section that this perceptron's nonlinearities are either tanh or the so-called σ-gelu for sufficiently small σ (c.f. (Yang and Hu, 2021, Assm 3.1)). [19] Note both tanh and σ-gelu are polynomially smooth.

We generalize the notion of *abc-parametrization* from Yang and Hu (2021) to non-Gaussian initializations:

---

we use a test function ψ and take it to be Lipschitz and bounded. Theorem A.3.2 used to prove Corollary A.3.3 works for Lipschitz nonlinearities, therefore we could embed ψ into the underlying tensor program. However, Theorem 3.3.7 does not work for Lipschitz nonlinearities and it is not obvious if we can use polynomially smooth functions as test functions for weak convergence. Nevertheless, we can take a bounded Lipschitz test function ψ and smoothen it with a kernel of width $\delta > 0$ the same way as we do in the proof of Theorem A.3.2. Then it is easy to see that $|c - c^\delta| \le \delta$ surely for any $\delta > 0$, where c^δ is the same as c in the proof of Corollary A.3.3 but with a δ-smoothed version of ψ. Since $|\mathbb{E}\, c - \mathring{c}| \le |\mathbb{E}\, c - \mathbb{E}\, c^\delta| + |\mathbb{E}\, c^\delta - \mathring{c}^\delta| + |\mathring{c}^\delta - \mathring{c}|$, we get $\limsup_{n\to\infty} |\mathbb{E}\, c - \mathring{c}| \le \delta + |\mathring{c}^\delta - \mathring{c}|$. Taking infimum over $\delta > 0$ gives $\mathbb{E}\, c \to \mathring{c}$.

[18] For the exact meaning of this setup, see Yang and Littwin (2021).

[19] This means σ-gelu smoothly approximates relu sufficiently well.

Definition 3.4.5. Let $\mathcal{D}$ be a distribution with mean 0, variance 1, and all moments finite. An *abc-parametrization for* $\mathcal{D}$, specified by a set of numbers $\{a_l, b_l\}_{l=1}^{L+1} \cup \{c\}$, parametrizes the MLP as follows: 1) each weight factors as $W^l = n^{-a_l} w^l$ for the actual trainable parameter w^l, 2) the weights are sampled iid $w^l_{\alpha\beta} \sim n^{-b_l}\mathcal{D}$ at initialization, and 3) the SGD learning rate is taken as ηn^{-c} for some constant η.

In their Dynamical Dichotomy Theorem, (Yang and Hu, 2021) classified all abc-parametrizations where $\mathcal{D} = \mathcal{N}(0,1)$ into the following categories: stable, nontrivial, kernel regime, and feature learning. Here, the *stable* and *nontrivial* categories overlap, and the *kernel* and *feature learning* regimes are mutually exclusive categories within their intersection. Each category is characterized by some set of linear inequalities in $\{a_l, b_l\}_{l=1}^{L+1} \cup \{c\}$ (c.f. (Yang and Hu, 2021, Sec 3)). It turns out that all but one abc-parametrization, the so-called μ-parametrization (abbreviated μP), exhibit defects in the infinite-width limit (such as losing the ability to learn features). This is formalized in (Yang and Hu, 2021, Thm 5.6). (Yang et al., 2021) then showed that μP gives rise to a new technology called μ*Transfer* that allows one to, for the first time, tune extremely large neural networks too expensive to train more than once, such as GPT-3 Brown et al. (2020).

Using Theorem 3.3.7, we see that all of the above notions, originally defined for Gaussian initialization, in fact are distributionally universal:

Theorem 3.4.6. *Let $\mathcal{D}, \mathcal{D}'$ be two distributions satisfying the moment conditions of Definition 3.4.5. Then an abc-parametrization for $\mathcal{D}$ is in feature learning (resp. kernel/stable/nontrivial) regime iff it is so for $\mathcal{D}'$. It is the μ-parametrization for $\mathcal{D}$ iff it is so for $\mathcal{D}'$.*

This suggests that the μTransfer technique mentioned above works regardless of the initialization distribution of weights.

3.5 Proof of Theorem 3.3.7

Limitations of the Proof Technique of Theorem 3.3.4. The proof of Theorem 3.3.4 as given in Yang (2020b) uses the Gaussianity of the matrices A^j in a very essential way. It leverages the property of multivariate Gaussians to remain Gaussian after conditioning on *linear* constraints.

In particular, it lets one understand the distribution of g^i conditioned on $g^{i-1}, \dots, g^1$. Indeed, under such conditioning, W^i is in general no longer iid but nevertheless constrained only by the following equalities (where now g^j and x^j should be considered deterministic under such conditioning):

$$g^j = W^j x^j, \quad \text{for all } j < i \text{ and where } W^j = W^i \text{ or } W^{i\top}.$$

This constraint is *linear* in W^i (even though it's generally nonlinear in $g^1, \dots, g^{i-1}$ since x^j is a nonlinear function of them). Therefore, when conditioned on $g^{i-1}, \dots, g^1$, the matrix W^i is *still Gaussian*, albeit with some conditional mean and covariance. Quickly glossing over the rest of the proof, this allows one to reason about the conditional distribution of g^i and eventually to reduce the almost sure convergence of the $(i+1)$th scalar c^{i+1} to the almost sure convergence of the ith scalar in a different program. Then an inductive argument over M for all programs of size M can prove Theorem 3.3.4. A more detailed proof sketch can be found in (Yang, 2020b, Section 6).

Now, when the matrices W^i are no longer Gaussian, this argument completely breaks down as we no longer have good control of the conditional distribution of W^i for general entry distributions. We therefore apply a different argument. The idea is to interpolate the weights from the Gaussian ones, for which convergence of c^i is given by Theorem 3.3.4, to the non-Gaussian ones, and show that c^i does not change along this interpolation in the limit of large n.

Our Proof of Theorem 3.3.7

Definition 3.5.1 (Interpolated Program)**.** Given a program T in Setup 3.3.6, let $\tilde{A}^i$ denote an iid Gaussian matrix with the same mean and variance as the non-Gaussian matrix A^i of the program. Then we define the *interpolated program* $T(t)$ for $t \in [0,1]$ as follows: $T(t)$ is identical to T except that its matrices take the following values:

$$A^i(t) \overset{\text{def}}{=} \tilde{A}^i \cos\left(\frac{\pi}{2}t\right) + A^i \sin\left(\frac{\pi}{2}t\right), \quad \text{for all } t \in [0,1] \text{ and } i = M_0 + 1, \dots, M, \tag{3.7}$$

Naturally, $W^i(t)$ inherits the same definition. The vectors and scalars in the program will change continuously as t varies, and consequently we write them as $g^i(t)$ and $c^i(t)$ for $i = 1, \dots, M$.[20]

In Section 3.5.1, we will discuss why this specific form of interpolation[21] is important in our case.

Here $t = 0$ corresponds to the Gaussian program, while $t = 1$ corresponds to the "target" non-Gaussian one. Eventually, we aim to show that all scalars $c^i(t)$, almost surely in the limit of $n \to \infty$, will remain constant as t varies from 0 to 1 (as stated by Theorem 3.5.2 shortly after setting some notations), thus proving that the Gaussian and non-Gaussian programs have the same limits.

Notations For any object (matrix, vector, or scalar) $\omega(t)$ of this interpolated program, we shorthand

$$\dot{\omega}(t) \overset{\text{def}}{=} \frac{d}{dt}\omega(t).$$

Big-O notation, e.g., $O(n^{-1/2})$, always suppress multiplicative constants independent of n, but which may depend on everything else. The supremum $\sup_t$ is always taken over $t \in [0,1]$.

Theorem 3.5.2. *Consider a program in Setup 3.3.6 and let c be a scalar in it. For any $p \geq 1$, we have*[22]

$$\sup_t \mathbb{E}\ |\dot{c}(t)|^p = O(n^{-p/2}), \quad \text{as } n \to \infty.$$

In other words, this result says $\dot{c}(t)$ is small in L^p norm uniformly over all t. With Theorem 3.5.2, some routine calculations (detailed below) then show that $c(1)$ converges almost surely and in L^p to the same limit as $c(0)$ yielding a proof of Theorem 3.3.7 as desired. Theorem 3.5.2 is proven in Appendix A.11, but we shall demonstrate it on a simple example in Section 3.5.2.

Routine Calculations Finishing the Proof of Theorem 3.3.7 from Theorem 3.5.2 Once Theorem 3.5.2 is proven, we have, for any $p \geq 1$,

$$\mathbb{E}\,|c(1) - c(0)|^p = \mathbb{E}\left|\int_0^1 \dot{c}(t)dt\right|^p \leq \int_0^1 \mathbb{E}\ |\dot{c}(t)|^p\, dt = O(n^{-p/2})$$

where the inequality follows from power mean or Hölder's inequality and the last equality follows from Theorem 3.5.2. Since the RHS goes to 0 with n, this implies that $c(1) - c(0)$ converges to 0

[20] but note that they are invariant to t for any n when $i \leq M_0$ because by construction, the initial vectors and scalars are the same between the Gaussian and non-Gaussian programs.

[21] This interpolation is similar to that used in Chen and Lam (2021) in the context of approximate message passing, which can be considered as an application of the TP framework to a specific kind of programs; see Appendix A.5 for comparison.

[22] The hidden multiplicative constant here is only independent of n but can depend on p and other details of the program.

in L^p as $n \to \infty$. With a standard application of Borel-Cantelli lemma, this also implies almost sure convergence of $c(1) - c(0)$. Since $c(0)$ converges to $\mathring{c}$ almost surely by the Gaussian theorem, Theorem 3.3.4, the same holds for $c(1)$.

In the process of proving Theorem 3.5.2, we will have proved pth moment bound on $c(1)$ for every finite $p \geq 1$ (Lemma A.9.6). Then a standard truncation technique uses this to convert the almost sure convergence of $c(1)$ to L^p convergence for all $p \in [1, \infty)$; see Theorem A.12.3.[23]

It remains to prove Theorem 3.5.2, which we demonstrate on a simple example in Section 3.5.2 after discussing the key properties of our interpolation (Eq. (3.7)).

3.5.1 Key Properties of Our Matrix Interpolation

There are many ways of interpolating between (or, more generally, to *couple*) $\tilde{A}$ and A other than Eq. (3.7), for example by diffusion instead. Why did we pick Eq. (3.7)? The following lemma summarizes the key properties of it, where especial attention should be paid to the fact that $\mathbb{E}\, a\dot{a} = 0$:

Lemma 3.5.3 (Interpolation Properties)**.** *For any matrix entry $a(t) = A^i_{\alpha\beta}(t)$ of a program in Setup 3.3.6:*

1. $\mathbb{E}\, \dot{a}(t) = \mathbb{E}\, a(t)\dot{a}(t) = 0$ *for all t*
2. *For any integers $j, k \geq 0$ with sum $\ell = j + k$,*

$$\sup_t \mathbb{E}\, |a(t)^j \dot{a}(t)^k| \leq \pi^\ell \nu_\ell n^{-\ell/2},$$

where ν_ℓ is the scaled moment bound in Setup 3.3.6.

The fact $\mathbb{E}\, a\dot{a} = 0$ is crucial and the main reason we picked the specific form Eq. (3.7) of the interpolation. It will allow us to kill leading terms in the Taylor expansions important to proving Theorem 3.5.2. We demonstrate this in an example in Eq. (3.11). On the other hand, Item 2 is just a very strong version of the obvious statement that "a and $\dot{a}$ both have typical size $1/\sqrt{n}$," but importantly, this is uniform over $t \in [0, 1]$.

These properties follow from easy calculations with Eq. (3.7), with details spelled out in Appendix A.6.

3.5.2 Proof of Theorem 3.5.2 for $p = 2$ in a Simple Scenario

Setup To make our arguments clear, we illustrate the proof of Theorem 3.5.2 on the simplest non-trivial case, with only a single matrix $A = A^1$, a single initial vector $g^1 \sim \mathcal{N}(0, I)$, and with c^1, c^2, and g^3 ignored. The objects of the interpolated program (Definition 3.5.1) are as follows:

$$g^2(t) = A(t)\phi(g^1(t)), \quad c(t) = \frac{1}{n}\sum_{\alpha=1}^n x_\alpha(t), \quad \text{where } x_\alpha(t) = \psi(g^1_\alpha(t), g^2_\alpha(t))$$

where, for brevity, we have shorthanded $c = c^3$, $x = x^3$, $\phi = \phi^1$, and $\psi = \phi^3$.[24]

[23] Note since $c(0)$ (the scalar in the Gaussian program) is not guaranteed by Theorem 3.3.4 to converge in L^p, the above L^p convergence of $c(1) - c(0)$ does not immediately imply the L^p convergence of $c(1)$.

[24] So we would have $M_0 = 1$, $M = 3$, but these values are not important for our purposes.

For simplicity, we will assume ϕ and ψ have derivatives of all orders bounded by 1 in absolute value. Our goal in this section is to demonstrate Theorem 3.5.2 with $p = 2$:

$$\sup_t \mathbb{E}\,\dot{c}(t)^2 = O(n^{-1}). \tag{3.8}$$

In what follows, we will only talk about the interpolated program, so we will suppress the argument (t) to lighten notation.

Bounding Derivatives of c against A

We will denote $x'_\alpha \stackrel{\text{def}}{=} \partial_{g_\alpha^2}\psi(g_\alpha^1, g_\alpha^2)$, $x''_\alpha \stackrel{\text{def}}{=} \partial^2_{g_\alpha^2}\psi(g_\alpha^1, g_\alpha^2)$, and so on, which shall cause no confusion because we will never take other derivatives of x or ψ. Then we can calculate

$$\frac{\partial c}{\partial A_{\alpha\beta}} = \frac{1}{n}x'_\alpha\phi(g_\beta^1), \quad \frac{\partial^2 c}{(\partial A_{\alpha\beta})^2} = \frac{1}{n}x''_\alpha\phi(g_\beta^1)^2, \quad \text{etc.} \tag{3.9}$$

By the assumption that ψ and ϕ have derivatives of any order bounded by 1, this shows c's derivative against $A_{\alpha\beta}$ of any order ≥ 1 is bounded in absolute value by $1/n$. Generalizing the above calculation to mixed derivatives, one can easily see more generally,

$$|\partial^r c| \leq 1/n \quad \text{for any order } r \geq 1 \text{ mixed derivative } \partial^r \text{ in entries of } A. \tag{3.10}$$

If we allow ψ and ϕ to be polynomially smooth in general, then this is still true in expectation, up to multiplicative constants; see Lemma A.10.6.

Expanding $\dot{c}^2$

At this point, it helps to think of the entries of A as a big vector a of size $N = n^2$. We will index entries of a using letters like κ, which stands for a pair $(\alpha, \beta) \in [n]^2$. We also let $D \in \mathbb{R}^N$ be the vector of derivatives $D_\kappa \stackrel{\text{def}}{=} \partial_{a_\kappa} c$ (with exact values given by Eq. (3.9)), so that, by chain rule,

$$\dot{c} = \sum_\kappa D_\kappa \dot{a}_\kappa.$$

Squaring this sum, we get

$$\dot{c}^2 = \sum_{\kappa,\lambda} D_\kappa D_\lambda \dot{a}_\kappa \dot{a}_\lambda = \sum_\kappa D_\kappa^2 \dot{a}_\kappa^2 + \sum_{\kappa\neq\lambda} D_\kappa D_\lambda \dot{a}_\kappa \dot{a}_\lambda$$

To show Eq. (3.8), it suffices to show that both sums $\sum_\kappa$ and $\sum_{\kappa\neq\lambda}$ in expectation can be bounded by $O(n^{-1})$ with a constant independent of $t \in [0, 1]$.

Bounding $\sum_\kappa D_\kappa^2 \dot{a}_\kappa^2$.

Because κ ranges over a set of size $N = n^2$, we have

$$\mathbb{E}\sum_\kappa D_\kappa^2 \dot{a}_\kappa^2 \leq n^2 \max_\kappa \mathbb{E}\, D_\kappa^2 \dot{a}_\kappa^2.$$

So it suffices to show $\mathbb{E}\, D_\kappa^2 \dot{a}_\kappa^2$ is bounded by Cn^{-3} where C is a constant independent of κ and $t \in [0,1]$. But by Cauchy-Schwarz,

$$\mathbb{E}\, D_\kappa^2 \dot{a}_\kappa^2 \le \sqrt{\mathbb{E}\, D_\kappa^4} \cdot \sqrt{\mathbb{E}\, \dot{a}_\kappa^4} = n^{-2} \cdot O(n^{-1}) = O(n^{-3})$$

where we used Eq. (3.10) and Lemma 3.5.3. The hidden constant is independent of κ and t, as desired.

Bounding $\sum_{\kappa \neq \lambda} D_\kappa D_\lambda \dot{a}_\kappa \dot{a}_\lambda$

Because (κ, λ) ranges over a set of size $\le N^2 = n^4$, we have

$$\mathbb{E} \sum_{\kappa \neq \lambda} D_\kappa D_\lambda \dot{a}_\kappa \dot{a}_\lambda \le n^4 \max_{\kappa \neq \lambda} \mathbb{E}\, D_\kappa D_\lambda \dot{a}_\kappa \dot{a}_\lambda.$$

So it suffices to show $\mathbb{E}\, D_\kappa D_\lambda \dot{a}_\kappa \dot{a}_\lambda$ is bounded by Cn^{-5} where C is a constant independent of κ, λ (as long as $\kappa \neq \lambda$) and $t \in [0,1]$.

Taylor Expansion Now $D_\kappa D_\lambda$ is a function of a. In particular, we will think of it as a function of a_κ and a_λ, keeping other entries of a fixed. To this end, we will write $D_\kappa D_\lambda = f(a_\kappa, a_\lambda)$. To reduce the amount of subscripts, denote $y = a_\kappa, z = a_\lambda$. Then Taylor expanding f to some order K to be specified below, we have

$$f(y,z) = \sum_{i+j \le K} y^i z^j \Delta_{ij} + \sum_{i+j = K+1} y^i z^j R_{ij}(y,z)$$

where $\Delta_{ij} = \frac{1}{(i+j)!}\binom{i+j}{i} \partial_y^i \partial_z^j f(0,0)$ and $R_{ij}(y,z) = \frac{1}{K!}\binom{K+1}{i} \int_0^1 (1-\xi)^K \partial_y^i \partial_z^j f(\xi y, \xi z) d\xi$.

This may look like a big scary expression, but as we will see soon enough, the exact form of Δ_{ij} does not matter beyond the fact that it is independent from y and z, and we can easily bound R_{ij} using Eq. (3.10).

Cancellation Using Independence and Zero-Mean Since Δ_{ij} is independent from y and z (as random variables), we have

$$\mathbb{E}\, \dot{y}\dot{z} y^i z^j \Delta_{ij} = (\mathbb{E}\, \dot{y} y^i)(\mathbb{E}\, \dot{z} z^j) \mathbb{E}\, \Delta_{ij}. \tag{3.11}$$

We see immediately that if $i < 2$ or $j < 2$ then this expectation will vanish due to Lemma 3.5.3. This motivates us to take K (the order of Taylor expansion) to be 3, so that

$$\mathbb{E}\, \dot{y}\dot{z} f(y,z) = \mathbb{E}\, \dot{y} y^2 \dot{z} z^2 R_{ij}(y,z)$$

Now $R_{ij}(y,z)$ is not independent from y and z unlike Δ_{ij}, but by Cauchy-Schwarz, we have

$$\mathbb{E}\, \dot{y} y^2 \dot{z} z^2 R_{ij}(y,z) \le \sqrt{\mathbb{E}\, (\dot{y} y^2 \dot{z} z^2)^2} \sqrt{\mathbb{E}\, R_{ij}(y,z)^2}$$

By Eq. (3.10) and the product rule, we see all order-$(K+1)$ (mixed) derivatives of f are bounded by $(K+2)/n^2 = 5/n^2$, so that $R_{ij}(y,z) \le C/n^2$ for some absolute constant C. This implies $\sqrt{\mathbb{E}\, R_{ij}(y,z)^2} \le C/n^2$ as well.

Finally, by Lemma 3.5.3, $\mathbb{E}\,(\dot{y}y^2\dot{z}z^2)^2 = \mathbb{E}\,(\dot{y}y^2)^2\mathbb{E}\,(\dot{z}z^2)^2 = O(n^{-6})$ independent of κ and λ. So altogether,

$$\mathbb{E}\,\dot{y}y^2\dot{z}z^2 R_{ij}(y,z) \le O(n^{-3})O(n^{-2}) = O(n^{-5})$$

with a constant independent of t, κ, and λ. This finishes the proof of Theorem 3.5.2 with $p = 2$ in our simple scenario.

From this example, one can see the importance of $\mathbb{E}\,a\dot{a} = 0$ in Lemma 3.5.3: had it not been the case, then Eq. (3.11) could be nonvanishing for $i = 2, j = 0$, so we could only Taylor expand f to order 1 instead of 3, losing a factor of n in the final bound as a result.

Going Beyond This Simple Example

To get the full result of Theorem 3.5.2, we need to extend this argument to 1) all powers p instead of just 2 (i.e., bounding $\mathbb{E}\,|\dot{c}|^p$), and to 2) all polynomially smooth nonlinearities.

For 1), the extension follows more or less the line of reasoning demonstrated here, which can be extracted as a general *moment argument* which we explain in Appendix A.8.

For 2), significant effort is needed to establish the property corresponding to Eq. (3.10). In fact, this is the bulk of the technical content of our work, done in Appendix A.9 and Appendix A.10.1, culminating in Lemma A.10.6 that generalizes Eq. (3.10).

3.6 Related Works

The Tensor Programs series discusses different applications of the (Gaussian) Master Theorem proven by Yang (2019a). They include: Gaussian process behavior at initialization (TP1, Yang (2019b)), convergence to a kernel method (TP2, Yang (2020a)), Free Independence Principle (TP3, Yang (2020b)), Dynamical Dichotomy and μ-parametrization (TP4, Yang and Hu (2021)), and finally application of μP to hyperparameter tuning (TP5, Yang et al. (2021)).

Neural networks converge to Gaussian processes as their width goes to infinity, as was proven by Lee et al. (2018); Matthews et al. (2018) for fully-connected nets, and by Novak et al. (2019); Garriga-Alonso et al. (2019) for convolutional nets; see also Hanin (2023). Using the Master Theorem, Yang (2019b) showed that this behavior holds for a very wide class of architectures, including not only convolutional, but also graph convolutional and recurrent neural nets, ResNets, networks with batch normalization, and networks with attention.

The seminal work of Jacot et al. (2018) demonstrated that under certain parametrization, the learning dynamics a neural net converges to that of a kernel method. The corresponding kernel was called Neural Tangent Kernel, or NTK, and drawn a lot of attention in recent years. While the result of Jacot et al. (2018) was proven only for fully-connected nets with smooth activations, the Master Theorem allows to generalize this result for a wider class of architectures (the same as mentioned above), see Yang (2020a).

Poole et al. (2016); Pennington et al. (2017); Xiao et al. (2018) and others studied trainability of very deep and wide neural networks using random matrix theory. Their analysis crucially relied on the assumption that hidden representations of a neural network at initialization were *freely independent* from the weights. TP3 Yang (2020b) was among the first works to validate this assumption rigorously; see also Pastur (2020); Pastur and Slavin (2020). Specifically, Pennington et al. (2017) used free independence in order to compute the spectrum of the *input-output Jacobian* of a deep and wide neural network. The work Pastur (2020) calculated this spectrum rigorously

without relying on free independence (thus proving that assuming free independence gives correct results), but assuming that all weights are iid Gaussians, while the follow-up work Pastur and Slavin (2020) relaxed this assumption.

Infinite-width behavior of a neural net depends on scaling of its hyperparameters (like initial weights variance and learning rate) with width. Dynamical Dichotomy proposed in TP4 Yang and Hu (2021) is a classification of scalings that are meaningful in a certain sense. Another classification of scalings with a different notion of meaningfulness was proposed earlier by Golikov (2020c,a), but only for two-layered networks.

A distribution universality property similar to our Theorem 3.3.7 was shown by Chen and Lam (2021) for approximate messaging passing. However, their model does not cover most of possible neural network computations; see Appendix A.5 for discussion.

3.7 Limitations

First, our Theorem 3.3.7 is applicable only to Tensor Programs with smooth nonlinearities, which rules out several popular activation functions like ReLU or MaxPool. Our Theorem A.3.2 (see Appendix A.3) does not really solve the issue since a Tensor Program expressing the backward pass involves derivatives of the activation functions, which are not even continuous for ReLU. As a workaround, we could consider their smoothed versions (e.g. Softplus instead of ReLU) with a controllable smoothness parameter, and put this parameter very close to zero thus getting "almost ReLU". Nevertheless, it would be nice to have a result similar to our Theorem 3.3.7 that would work with the same class of nonlinearities as Theorem 3.3.4 does; we leave it for future work.

3.8 Conclusions

We presented a generalization of the Master Theorem of Yang (2019a) to non-Gaussian weight initializations. Our generalization allows for the same applications as the original Master Theorem, thus broadening the scope of applicability of the Tensor Programs machinery.

4 Tail bounds for Tensor Programs

Authors: Eugene Golikov, Greg Yang.

Link and reference: this is an unpublished work.

Contribution: the proofs and the present text are by the author of the present thesis.

4.1 Introduction

As discussed in Section 1.5, the Master theorem (Theorems 1.4.1 and 1.4.2), while giving limits to scalar functions generated by a Tensor Program (e.g. loss and accuracy values at each training step), does not provide a tail bound, i.e. an expression of the form

$$\mathcal{P}\left(c^l - \mathring{c}^l > \tau\right) \leq B(n, l, \tau). \tag{4.1}$$

where c^l is a scalar variable generated at l-th iteration of the Tensor Program at hand, n is width of the program (i.e. the dimension of all hidden vectors), and $\mathring{c}^l$ is the (almost sure) limit of c^l as $n \to \infty$.

The present chapter presents a tail bound for a specific class of Tensor Programs. Our bound is the first such bound applicable to this class of Tensor Programs. Also, our bound is exact, meaning that all the constant are defined by a specific recurrence relation. However, our present bound is arguably loose as it claims $B(n, l, \tau)$ to have a suboptimal dependence on l and the Lipschitz constant of the nonlinearities of the program. We refer the reader Section 1.5 for the discussion of how this bound could be improved in the future.

4.2 Main tail bound

Setup. In this work, we consider a Tensor Program performing L steps, using a fresh-new matrix for the MatMul operation at each step:

$$g^{l+1} = A^l \phi(g^1, \dots, g^l), \qquad c^l = \frac{1}{n} \sum_\alpha \psi(g^1_\alpha, \dots, g^l_\alpha) \qquad \forall l \in [L]. \tag{4.2}$$

As already discussed in Section 1.5, such a program could describe the very first forward pass in a neural network, but not a backward pass. The Master theorem could be applied to such programs

to give a Neural Network Gaussian Process correspondence: see Section 1.4 and Yang (2019b).

Assumptions. We assume there exists $\lambda > 0$ such that ϕ is λ-Lipschitz and ψ is λ^2-order-2-pseudo-Lipschitz with a constant ν^l, i.e.

$$|\psi(z^1, \dots, z^l) - \psi(\tilde{z}^1, \dots, \tilde{z}^l)| \le \lambda^2 \left(\nu^l + \sum_{j=1}^{l} \left(|z^j| + |\tilde{z}^j|\right)\right) \sum_{k=1}^{l} \left|z^k - \tilde{z}^k\right|. \tag{4.3}$$

We assume also $\phi(0) = 0$, so that ϕ^2 is λ^2-order-2-pseudo-Lipschitz with a constant 0.

4.2.1 Result

Define

$$v^l = \frac{1}{n}\sum_{\beta} \phi^2(g_\beta^{1:l}), \qquad d^l(\nu) = \frac{1}{n}\sum_{\beta}\left(\nu + 2\sum_{j=1}^{l}|g_\beta^j|\right)^2, \tag{4.4}$$

and let $\mathring{v}^l$ and $\mathring{d}^l(\nu)$ be their respective a.s. limits. We are going to prove the following result:

Theorem 4.2.1. *Let $\mathring{c}^l$ be an a.s. limit of c^l. Then $\forall l \in [L],\ n \in \mathbb{N},\ \tau^l > 0$*

$$\mathcal{P}(c^l - \mathring{c}^l > \tau) \le B^l \left(e^{-n\left(\frac{\tau}{C_2^l(\tau, \nu^l)}\right)^2} + ne^{-C_0 n}\right), \tag{4.5}$$

where $B^l = \frac{3^l - 1}{2}$, $C_0 = 8e^{-\frac{2}{\pi}} \approx 4.23$, while C_2^l is defined recursively:

$$\hat{C}_2(v, d) = \lambda^2 \sqrt{2\left(\left(3 + 256e^{\frac{2}{\pi}}\right)v^2 + \left(2 + 32\pi e^{\frac{2}{\pi}}\right)\sqrt{\frac{(2v)^3 d}{\pi}} + \left(1 + 8\pi e^{\frac{2}{\pi}}\right)vd\right)}; \tag{4.6}$$

$$C_2^1(\tau, \nu) = \hat{C}_2(1, \nu^2), \qquad \bar{C}_2^l(\tau, \nu) = \hat{C}_2\left(\bar{v}^l(\tau), \bar{d}^l(\tau, \nu)\right); \tag{4.7}$$

$$C_2^{l+1}(\tau, \nu) = \bar{C}_2^l\left(\hat{\tau}^l(\tau, \nu), \nu\right) + \lambda^2 D^l(\nu) C_2^l\left(\hat{\tau}^l(\tau, \nu), \nu\right) + C_2^l\left(\tau, \bar{\nu}^l(\tau, \nu)\right), \tag{4.8}$$

where $D^l(\nu) = 1 + \sqrt{\frac{\mathring{d}^l(\nu)}{2\pi \mathring{v}^l}}$, $\hat{\tau}^l(\tau, \nu) = \frac{\tau}{\lambda^2 D^l(\nu)}$, $\bar{v}^l(\tau) = \mathring{v}^l + \tau$, $\bar{d}^l(\tau, \nu) = \mathring{d}^l(\nu) + \frac{4\tau}{\lambda^2}$, $\bar{\nu}^l(\tau, \nu) = \nu + \sqrt{\frac{8}{\pi}\bar{v}^l(\hat{\tau}^l(\tau, \nu))}$, and we have the same bound for $\mathcal{P}(c^l - \mathring{c}^l < -\tau)$.

4.2.2 Asymptotics

Before moving on to the proof, let us discuss the asymptotics for large λ. We have $g^l = \Theta(\lambda^{l-1})$. Suppose $\nu^l = 0\ \forall l \ge 1$. Then $c^l = \Theta(\lambda^{2l})$ and $v^l = \Theta(\lambda^{2l})$, while $d^l(\nu) = \Theta\left(\left(\nu + 2\lambda^{l-1}\right)^2\right)$. We take $\tau^l \sim tc^l \sim t\lambda^{2l}$.

We have $D^l(0) = 1$, hence $\hat{\tau}^l(\tau, 0) = \tau/\lambda^2$, and in particular, $\hat{\tau}^l(\tau^{l+1}, 0) = \tau^l$.

Suppose $v^l \sim (\rho\lambda)^{2l}$ and $d^l(0) \sim 4(\sigma\lambda)^{2l-2}$ for some $\rho, \sigma > 0$. Then $\bar{v}^l(\tau^l) \sim (\rho^{2l} + t)\lambda^{2l}$, $\bar{d}^l(\tau^l, 0) \sim 4(\sigma^{2l-2} + t)\lambda^{2l-2}$, and

$$\bar{C}_2^l(\tau^l, 0) = \hat{C}_2(\bar{v}^l(\tau^l), \bar{d}^l(\tau^l, 0)) \sim \lambda^{2l+2}(\rho^{2l} + t)\sqrt{2\left(3 + 256e^{\frac{2}{\pi}}\right)}. \tag{4.9}$$

We will look for $C_2^l(\tau^l, 0) \sim q^l\lambda^{2l}$.

It seems natural to suppose $C_2^l(\tau^{l+1}, \bar{\nu}(\tau^{l+1}, \nu)) = o(\lambda^{2l+2})$. If we do, we get

$$q^{l+1} = (\rho^{2l} + t)\sqrt{2\left(3 + 256e^{\frac{2}{\pi}}\right)} + q^l. \tag{4.10}$$

Since $q^1 = \sqrt{2\left(3 + 256e^{\frac{2}{\pi}}\right)}$, the solution is

$$q^l = \sqrt{2\left(3 + 256e^{\frac{2}{\pi}}\right)}\left((l-1)t + \frac{1-\rho^{2l}}{1-\rho^2}\right). \tag{4.11}$$

Therefore if we would like to bound the tail as $\mathcal{P}(c^l - \mathring{c}^l > \tau) \leq \delta$ for some fixed $\delta \in (0, 1)$, we need to have

$$n \geq 2\left(3 + 256e^{\frac{2}{\pi}}\right)\left(\frac{\frac{1-\rho^{2l}}{1-\rho^2} + (l-1)t}{t}\right)^2 \ln\left(\frac{3^l - 1}{2\delta}\right) \sim 2\left(3 + 256e^{\frac{2}{\pi}}\right)(\ln 3)l^3 \tag{4.12}$$

as $l \to \infty$. As we discuss in Section 1.5, this is likely to be too weak since the bound of Hanin and Nica (2020b) applicable for ReLU nets, stays finite when $n \sim l$.

However, even the assumption we made above for $C_2^l(\tau^{l+1}, \bar{\nu}(\tau^{l+1}, \nu))$ seems to be wrong: while it is quite difficult to resolve the recurrence for this quantity even in the limit of large λ, our numerical experiments suggest that it grows as $\lambda^{\Theta(n^2)}$.

This indicates that the present bound is suboptimal. In Section 4.3 below, we state an alternative tail bound that does not have this drawback, but works only for an even more restricted class of Tensor Programs. This even more restricted setup is also closer (but still stronger as a wider class of activation functions are covered) to that of Hanin and Nica (2020b) we compare to.

4.2.3 The proof

Now we can move on to the proof of Theorem 4.2.1. All the missing proofs for auxiliary results we use below are in Appendix B.1.

Proof. The proof is by induction on l.

Induction base. For $l = 1$, we use the following tail bound for sums of iid random variables:

Corollary 4.2.2 (Bakhshizadeh et al. (2023))**.** *Suppose $(X_i)_{i=1}^m$ are iid RVs and $\mathcal{P}(X_i > t) \leq e^{-kt}$. Fix $\beta \in (0, 1)$ and define $S_m = \sum_{i=1}^m X_i$ and*

$$v_\beta = \mathbb{E}\left[(X - \mathbb{E}\,X)^2 \operatorname{Ind}[X \leq \mathbb{E}\,X]\right] + \frac{1}{(1-\beta)^3}\frac{2}{k^2 e^{k\mathbb{E}\,X}}. \tag{4.13}$$

Then $\forall m > \frac{1}{\beta}\frac{\mathbb{E}\,X}{v_\beta k}$

$$\mathcal{P}(S_m - \mathbb{E}\,S_m > mt) \le e^{-\frac{mt^2}{2v_\beta}} + me^{-\beta m v_\beta k^2}. \tag{4.14}$$

The original corollary also contains a tighter bound for $t \ge \beta v_\beta k$ but we do not need it for the present construction.

For the sake of simplicity, we take $\beta = 1/2$. We upper-bound the first term of v_β with $\mathbb{E}\,[X^2]$; the above result remains valid if we substitute v_β with an upper-bound. Finally, we change the notation in order to match ours. This gives us

Lemma 4.2.3. *Suppose* $(x_\beta)_{\beta=1}^n$ *are iid RVs and* $\mathcal{P}(x > t) \le e^{-\kappa t}$ *for some* $\kappa > 0$. *Define* $c = \frac{1}{n}\sum_\beta x_\beta$ *and*

$$q = \mathbb{E}\,[x^2] + \frac{16}{\kappa^2 e^{\kappa \mathbb{E}\,[x]}}. \tag{4.15}$$

Then $\forall n > \frac{2\mathbb{E}\,x}{q\kappa}$

$$\mathcal{P}(c - \mathbb{E}\,[c] > t) \le e^{-\frac{nt^2}{2q}} + ne^{-\frac{nq\kappa^2}{2}}. \tag{4.16}$$

Applying this result to $x_\beta = \psi(g_\beta^1) - \psi(0)$ gives the theorem statement for $l = 1$:

Corollary 4.2.4.

$$\forall n \in \mathbb{N},\ \tau > 0 \qquad \mathcal{P}(c^1 - \mathring{c}^1 > \tau) \le e^{-C_2^1 n\tau^2} + ne^{-C_0 n}, \tag{4.17}$$

and we have the same bound for $\mathcal{P}(c^1 - \mathring{c}^1 < -\tau)$.

Induction step. Suppose the theorem claim holds for l. In order to perform the induction step, we would like to apply Lemma 4.2.3 to $x_\beta = \psi(g_\beta^{1:l}, \sqrt{v^l} z_\beta) - \psi(g_\beta^{1:l}, 0)$ conditioned on $g^{1:l}$, where $z \sim \mathcal{N}(0, I_n)$ and $v^l = \frac{1}{n}\sum_\beta \phi^2(g_\beta^{1:l})$. However, this result cannot be applied directly because x_β are not indentically distributed, even when conditioned on $g^{1:l}$. For this reason, we generalize the lemma:

Lemma 4.2.5. *Suppose* $(x_\beta)_{\beta=1}^n$ *are independent RVs, and* $\forall \beta \in [n]$ $\mathcal{P}(x_\beta > t) \le e^{-\kappa_\beta t}$ *for some* $\kappa_\beta > 0$. *Define* $c = \frac{1}{n}\sum_\beta x_\beta$ *and*

$$q_\beta = \mathbb{E}\,[x_\beta^2] + \frac{16}{\kappa_\beta^2 e^{\kappa_\beta \mathbb{E}\,[x_\beta]}}. \tag{4.18}$$

Then $\forall n > 2\max_\beta\left(\frac{\mathbb{E}\,[x_\beta]}{q_\beta \kappa_\beta}\right)$

$$\mathcal{P}(c - \mathbb{E}\,[c] > t) \le e^{-\frac{nt^2}{\frac{2}{n}\sum_\beta q_\beta}} + ne^{-\frac{n \min_\beta (q_\beta \kappa_\beta^2)}{2}}. \tag{4.19}$$

We now apply Lemma 4.2.5 to $x_\beta = \psi(g_\beta^{1:l}, \sqrt{v^l} z) - \psi(g_\beta^{1:l}, 0)$ conditioned on $g^{1:l}$, where $z \sim \mathcal{N}(0, 1)$:

Corollary 4.2.6. *Let* $\mathring{c}^{l+1}(g^{1:l}) = \frac{1}{n}\sum_\beta \mathbb{E}_{z\sim\mathcal{N}(0,1)}\psi(g_\beta^{1:l}, \sqrt{v^l} z)$, $d^l = \frac{1}{n}\sum_\beta \left(\nu^{l+1} + 2\sum_{j=1}^l |g_\beta^j|\right)^2$. *Then*

$$\forall n \in \mathbb{N},\ \tau > 0 \qquad \mathcal{P}\left(c^{l+1} - \mathring{c}^{l+1}(g^{1:l}) > \tau \,\middle|\, g^{1:l}\right) \le e^{-\hat{C}_2^l n\tau^2} + ne^{-C_0 n}, \tag{4.20}$$

where

$$\hat{C}_2^l = \frac{1}{2\lambda^4\left((3+256e^{\frac{2}{\pi}})(v^l)^2 + \left(2+32\pi e^{\frac{2}{\pi}}\right)\sqrt{\frac{(2v^l)^3 d^l}{\pi}} + (1+8\pi e^{\frac{2}{\pi}})v^l d^l\right)}, \tag{4.21}$$

and we have the same bound for $\mathcal{P}(c^{l+1} - \mathring{c}^{l+1}(g^{1:l}) < -\tau)$.

Let us perform the actual induction step:

$$\begin{aligned}\mathcal{P}(c^{l+1} - \mathring{c}^{l+1} > \tau) \le \inf_{\rho\in[0,\tau]}\inf_{\sigma\ge 0}\Bigg[&\sup_{g^{1:l}:\begin{cases}\mathring{c}^{l+1}(g^{1:l}) - \mathring{c}^{l+1} \le \rho,\\ v^l - \mathring{v}^l \le \sigma,\\ \frac{\lambda^2}{4}\left(d^l - \mathring{d}^l\right) \le \sigma\end{cases}} \mathcal{P}\left(c^{l+1} - \mathring{c}^{l+1} > \tau \,|\, g^{1:l}\right)\\ &+ \mathcal{P}\left(v^l - \mathring{v}^l > \sigma\right) + \mathcal{P}\left(\tfrac{\lambda^2}{4}\left(d^l - \mathring{d}^l\right) > \sigma\right)\\ &+ \mathcal{P}\left(\mathring{c}^{l+1}(g^{1:l}) - \mathring{c}^{l+1} > \rho \,|\, v^l - \mathring{v}^l \le \sigma\right)\Bigg].\end{aligned} \tag{4.22}$$

The first term can be bounded using Corollary 4.2.6, while the following two terms are bounded by the induction hypothesis (note that d^l is 4-order-2-pseudo-Lipschitz):

$$\begin{aligned}\mathcal{P}(c^{l+1} - \mathring{c}^{l+1} > \tau) \le \inf_{\rho\in[0,\tau]}\inf_{\sigma\ge 0}\Bigg[&e^{-n\left(\frac{\tau-\rho}{\bar{C}_2^l(\sigma)}\right)^2} + ne^{-C_0 n}\\ &+ B^l\left(e^{-n\left(\frac{\sigma}{\bar{C}_2^l(\sigma,0)}\right)^2} + ne^{-C_0 n}\right) + B^l\left(e^{-n\left(\frac{\sigma}{\bar{C}_2^l(\sigma,\nu^{l+1})}\right)^2} + ne^{-C_0 n}\right)\\ &+ \mathcal{P}\left(\mathring{c}^{l+1}(g^{1:l}) - \mathring{c}^{l+1} > \rho \,|\, v^l - \mathring{v}^l \le \sigma\right)\Bigg] \qquad \forall \tau > 0,\ n \in \mathbb{N},\end{aligned} \tag{4.23}$$

where

$$\bar{C}_2^l(\sigma) = \lambda^2\sqrt{2\left(\left(3+256e^{\frac{2}{\pi}}\right)(\bar{v}_\sigma^l)^2 + \left(2+32\pi e^{\frac{2}{\pi}}\right)\sqrt{\frac{(2\bar{v}_\sigma^l)^3\bar{d}_\sigma^l}{\pi}} + \left(1+8\pi e^{\frac{2}{\pi}}\right)\bar{v}_\sigma^l\bar{d}_\sigma^l\right)} \tag{4.24}$$

for $\bar{v}_\sigma^l = \mathring{v}^l + \sigma$ and $\bar{d}_\sigma^l = \mathring{d}^l + \frac{4\sigma}{\lambda^2}$.

Let us bound the last term:

$$\begin{aligned}\mathcal{P}\left(\mathring{c}^{l+1}(g^{1:l}) - \mathring{c}^{l+1} > \rho \,\middle|\, v^l - \mathring{v}^l \le \sigma\right) &\le \mathcal{P}\left(\sup_{v\in[0,\mathring{v}^l+\sigma]}\left(\frac{1}{n}\sum_\beta \mathbb{E}_z\psi(g_\beta^{1:l}, \sqrt{v}z) - \mathring{c}^{l+1}\right) > \rho\right)\\ &\le \mathcal{P}\left(\frac{1}{n}\sum_\beta \psi_\sigma(g_\beta^{1:l}) - \mathring{c}^{l+1} > \rho\right),\end{aligned} \tag{4.25}$$

where $\psi_\sigma(z^{1:l}) = \sup_{v\in[0,\mathring{v}^l+\sigma]} \mathbb{E}_{z\sim\mathcal{N}(0,1)}\psi(z^{1:l},\sqrt{v}z)$.

Let $u^l_\sigma := \frac{1}{n}\sum_\beta \psi_\sigma(g^{1:l}_\beta)$. We would like to apply the induction hypothesis to it. For this, we need it to be pseudo-Lipschitz of order 2:

$$\begin{aligned}\psi_\sigma(z^{1:l}) - \psi_\sigma(\tilde{z}^{1:l}) &\le \sup_{v\in[0,\mathring{v}^l+\sigma]} \mathbb{E}_{z\sim\mathcal{N}(0,1)}\left(\psi(z^{1:l},\sqrt{v}z) - \psi(\tilde{z}^{1:l},\sqrt{v}z)\right)\\ &\le \lambda^2 \sup_{v\in[0,\mathring{v}^l+\sigma]} \mathbb{E}_{z\sim\mathcal{N}(0,1)}\left(\nu^{l+1} + 2\sqrt{v}|z| + \sum_j(|z^j|+|\tilde{z}^j|)\right)\sum_j|z^j-\tilde{z}^j|\\ &\le \lambda^2\left(\nu^{l+1} + \sqrt{(\mathring{v}^l+\sigma)\frac{8}{\pi}} + \sum_j(|z^j|+|\tilde{z}^j|)\right)\sum_j|z^j-\tilde{z}^j|,\end{aligned} \tag{4.26}$$

and we get the same bound for $\psi_\sigma(\tilde{z}^{1:l}) - \psi_\sigma(z^{1:l})$.

By the induction hypothesis, $\forall n\in\mathbb{N},\ \varsigma>0$

$$\mathcal{P}\left(u^l_\sigma - \mathring{u}^l_\sigma > \varsigma\right) \le B^l\left(e^{-n\left(\frac{\varsigma}{C_2^l(\varsigma,\nu^l_\sigma)}\right)^2} + ne^{-C_0 n}\right),$$

where $\nu^l_\sigma := \nu^{l+1} + \sqrt{(\mathring{v}^l+\sigma)\frac{8}{\pi}}$.

Lemma 4.2.7. *$\forall\sigma>0$ $\mathring{u}^l_\sigma \le \mathring{c}^{l+1} + D^l\lambda^2\sigma$, where $D^l = 1+\sqrt{\frac{\mathring{d}^l}{2\pi\mathring{v}^l}}$.*

This lemma together with the induction hypothesis implies $\forall n\in\mathbb{N},\ \rho>0,\ \sigma\in\left[0,\frac{\rho}{\lambda^2 D^l}\right]$

$$\begin{aligned}\mathcal{P}\left(\mathring{c}^{l+1}(g^{1:l}) - \mathring{c}^{l+1} > \rho \,\middle|\, v^l - \mathring{v}^l \le \sigma\right) &\le \mathcal{P}\left(u^l_\sigma - \mathring{c}^{l+1} > \rho\right) \le \mathcal{P}\left(u^l_\sigma - \mathring{u}^l_\sigma > \rho - \lambda^2 D^l\sigma\right)\\ &\le B^l\left(e^{-n\left(\frac{\rho-\lambda^2D^l\sigma}{C_2^l(\rho-\lambda^2D^l\sigma,\nu^l_\sigma)}\right)^2} + ne^{-C_0n}\right).\end{aligned} \tag{4.27}$$

Therefore

$$\begin{aligned}\mathcal{P}(c^{l+1} - \mathring{c}^{l+1} > \tau) \le \inf_{\rho\in[0,\tau]}\inf_{\sigma\in\left[0,\frac{\rho}{\lambda^2D^l}\right]}\Bigg[&e^{-n\left(\frac{\tau-\rho}{\bar{C}_2^l(\sigma)}\right)^2} + ne^{-C_0n}\\ &+ B^l\left(e^{-n\left(\frac{\sigma}{C_2^l(\sigma,0)}\right)^2} + ne^{-C_0n}\right) + B^l\left(e^{-n\left(\frac{\sigma}{C_2^l(\sigma,\nu^{l+1})}\right)^2} + ne^{-C_0n}\right)\\ &+ B^l\left(e^{-n\left(\frac{\rho-\lambda^2D^l\sigma}{C_2^l(\rho-\lambda^2D^l\sigma,\nu^l_\sigma)}\right)^2} + ne^{-C_0n}\right)\Bigg] \qquad \forall\tau>0,\ n\in\mathbb{N}.\end{aligned} \tag{4.28}$$

Since $\bar{C}_2^l(\sigma)$ is increasing with σ, $C_2^l(\sigma,\nu)$ is increasing with σ and ν by the induction hypothesis,

and ν_σ^l is increasing with σ, we can furter upper-bound the above:

$$\mathcal{P}(c^{l+1} - \hat{c}^{l+1} > \tau) \leq \inf_{\rho\in[0,\tau]} \inf_{\sigma\in\left[0,\frac{\rho}{\lambda^2 D^l}\right]} \Bigg[e^{-n\left(\frac{\tau-\rho}{\bar{C}_2^l\left(\frac{\rho}{\lambda^2 D^l}\right)}\right)^2} + ne^{-C_0 n} \\ +2B^l \left(e^{-n\left(\frac{\sigma}{C_2^l\left(\frac{\rho}{\lambda^2 D^l},\nu^{l+1}\right)}\right)^2} + ne^{-C_0 n} \right) + B^l \left(e^{-n\left(\frac{\rho-\lambda^2 D^l\sigma}{C_2^l\left(\rho,\nu^l{}_{\frac{\rho}{\lambda^2 D^l}}\right)}\right)^2} + ne^{-C_0 n} \right) \Bigg] \tag{4.29}$$

$\forall \tau > 0,\ n \in \mathbb{N}$. We take σ to equalize the exponents containing it:

$$\frac{\sigma}{C_2^l\left(\frac{\rho}{\lambda^2 D^l},\nu^{l+1}\right)} = \frac{\rho - \lambda^2 D^l \sigma}{C_2^l\left(\rho,\nu^l{}_{\frac{\rho}{\lambda^2 D^l}}\right)}. \tag{4.30}$$

The solution is

$$\sigma = \frac{\rho}{\lambda^2 D^l + \frac{C_2^l\left(\rho,\nu^l{}_{\frac{\rho}{\lambda^2 D^l}}\right)}{C_2^l\left(\frac{\rho}{\lambda^2 D^l},\nu^{l+1}\right)}}. \tag{4.31}$$

Plugging it into the above,

$$\mathcal{P}(c^{l+1} - \hat{c}^{l+1} > \tau) \leq \inf_{\rho\in[0,\tau]} \Bigg[e^{-n\left(\frac{\tau-\rho}{\bar{C}_2^l\left(\frac{\rho}{\lambda^2 D^l}\right)}\right)^2} + 3B^l e^{-n\left(\frac{\rho}{\lambda^2 D^l C_2^l\left(\frac{\rho}{\lambda^2 D^l},\nu^{l+1}\right) + C_2^l\left(\rho,\nu^l{}_{\frac{\rho}{\lambda^2 D^l}}\right)}\right)^2} \\ + (3B^l+1) n e^{-C_0 n} \Bigg] \qquad \forall \tau > 0,\ n \in \mathbb{N}. \tag{4.32}$$

Again, by monotonicity of $\bar{C}_2^l$ and C_2^l, we further upper-bound the above as

$$\mathcal{P}(c^{l+1} - \hat{c}^{l+1} > \tau) \leq \inf_{\rho\in[0,\tau]} \Bigg[e^{-n\left(\frac{\tau-\rho}{\bar{C}_2^l\left(\frac{\tau}{\lambda^2 D^l}\right)}\right)^2} + 3B^l e^{-n\left(\frac{\rho}{\lambda^2 D^l C_2^l\left(\frac{\tau}{\lambda^2 D^l},\nu^{l+1}\right) + C_2^l\left(\tau,\nu^l{}_{\frac{\tau}{\lambda^2 D^l}}\right)}\right)^2} \\ + (3B^l+1) n e^{-C_0 n} \Bigg] \qquad \forall \tau > 0,\ n \in \mathbb{N}. \tag{4.33}$$

We take ρ to equalize the exponents containing it:

$$\frac{\tau - \rho}{\bar{C}_2^l\left(\frac{\tau}{\lambda^2 D^l}\right)} = \frac{\rho}{\lambda^2 D^l C_2^l\left(\frac{\tau}{\lambda^2 D^l},\nu^{l+1}\right) + C_2^l\left(\tau,\nu^l{}_{\frac{\tau}{\lambda^2 D^l}}\right)}. \tag{4.34}$$

The solution is

$$\rho = \frac{\tau}{1 + \frac{\bar{C}_2^l\left(\frac{\tau}{\lambda^2 D^l}\right)}{\lambda^2 D^l C_2^l\left(\frac{\tau}{\lambda^2 D^l},\nu^{l+1}\right) + C_2^l\left(\tau,\nu^l{}_{\frac{\tau}{\lambda^2 D^l}}\right)}}. \tag{4.35}$$

Therefore

$$\tau - \rho = \frac{\tau}{1 + \frac{\lambda^2 D^l C_2^l\left(\frac{\tau}{\lambda^2 D^l}, \nu^{l+1}\right) + C_2^l\left(\tau, \nu^l_{\frac{\tau}{\lambda^2 D^l}}\right)}{\bar{C}_2^l\left(\frac{\tau}{\lambda^2 D^l}\right)}}. \tag{4.36}$$

This gives the final bound:

$$\mathcal{P}(c^{l+1} - \mathring{c}^{l+1} > \tau) \leq (3B^l + 1)\left(e^{-n\left(\frac{\tau}{C_2^{l+1}(\tau, \nu^{l+1})}\right)^2} + ne^{-C_0 n}\right) \qquad \forall \tau > 0,\ n \in \mathbb{N}, \tag{4.37}$$

where

$$C_2^{l+1}(\tau, \nu^{l+1}) = \bar{C}_2^l\left(\frac{\tau}{\lambda^2 D^l}\right) + \lambda^2 D^l C_2^l\left(\frac{\tau}{\lambda^2 D^l}, \nu^{l+1}\right) + C_2^l\left(\tau, \nu^l_{\frac{\tau}{\lambda^2 D^l}}\right). \tag{4.38}$$

□

4.3 Tail bound for a restricted class

Setup and assumptions. In this section, we consider an even more restricted class for which all nonlinearities depend only on the last generated vector. This class covers e.g. the very first forward pass of a fully-connected network without bias vectors (as they require multivariate nonlinearities, see Section 1.4).

That is, consider the following iteration:

$$g^{l+1} = A^l \hat{\phi}(g^l), \qquad c^l = \frac{1}{n}\sum_\alpha \left(\hat{\chi}(g^l_\alpha)\right)^2 \qquad \forall l \in [L], \tag{4.39}$$

where $\hat{\phi}$ and $\hat{\chi}$ are both λ-Lipschitz and monotonically non-decreasing, and $\hat{\phi}(0) = \hat{\chi}(0) = 0$. This setup is similar to the one of Hanin and Nica (2020b) but our activation functions are allowed to be any Lipschitz ones equal to zero at zero, while Hanin and Nica (2020b) allow only for (Leaky)ReLUs (this is important for their proofs).

4.3.1 Result

The theorem similar to Theorem 4.2.1 but for the restricted class above is given below:

Theorem 4.3.1. *Let $\mathring{c}^l$ be an a.s. limit of c^l. Then $\forall l \in [L],\ n \in \mathbb{N},\ \tau > 0$*

$$\mathcal{P}(c^l - \mathring{c}^l > \tau) \leq B^l\left(e^{-n\left(\frac{\tau}{C_2^l(\tau)}\right)^2} + ne^{-C_0 n}\right), \tag{4.40}$$

where $B^l = l$, $C_0 = 8e^{-\frac{1}{2}} \approx 4.85$, while C_2^l is defined recursively:

$$C_2^1(\tau^1) = \lambda^2\sqrt{2\left(3 + 64e^{1/2}\right)}, \qquad \bar{C}_2^l(\tau^l) = \lambda^2(\mathring{v}^l + \tau^l)\sqrt{2\left(3 + 64e^{1/2}\right)}, \tag{4.41}$$

$$C_2^{l+1}(\tau^{l+1}) = \lambda^2 C_2^l(\tau^{l+1}/\lambda^2) + \bar{C}_2^l(\tau^{l+1}/\lambda^2), \tag{4.42}$$

and we have the same bound for $\mathcal{P}(c^l - \mathring{c}^l < -\tau)$.

The recurrence relations defining the constants in the above theorem look much simpler compared to that of our more general theorem, Theorem 4.2.1. We study the large λ asymptotics in the following section. In this case, all recurrences get resolved easily and the corresponding asymptotic make more sense, as we will see below.

4.3.2 Asymptotics

Similar to Section 4.2, we study the asymptotics for the large Lipschitz constant λ.

We have $g^l = \Theta(\lambda^{l-1})$, $\mathring{v}^l = \Theta(\lambda^{2l})$, and $\mathring{c}^l = \Theta(\lambda^{2l})$, therefore it makes sense to take $\tau^l = t\lambda^{2l}$ for some $t > 0$. In this case, $\tau^{l+1}/\lambda^2 = \tau^l \ \forall l$. Suppose also $\mathring{v}^l = (\rho\lambda)^{2l}$ for some $\rho > 0$. This allows us, with a slight abuse of notation, to consider C_2^l and Q^l as functions of t:

$$C_2^1(t) = \lambda^2\sqrt{2\left(3+64e^{1/2}\right)}, \quad \bar{C}_2^l(t) = \lambda^{2l+2}(\rho^{2l}+t)\sqrt{2\left(3+64e^{1/2}\right)}, \quad C_2^{l+1}(t) = \lambda^2 C_2^l(t) + \bar{C}_2^l(t). \tag{4.43}$$

We will look for $C_2^l(t) = \sqrt{2\left(3+64e^{1/2}\right)}q^l(t)\lambda^{2l}$:

$$q^1(t) = 1, \qquad q^{l+1}(t) = q^l(t) + \rho^{2l} + t. \tag{4.44}$$

The solution is $q^l(t) = \frac{1-\rho^{2l}}{1-\rho^2} + (l-1)t$.

Therefore if we would like to bound the tail as $\mathcal{P}(c^l - \mathring{c}^l > \tau) \le \delta$ for some fixed $\delta \in (0,1)$, we need to have

$$n \ge 2\left(3+64e^{1/2}\right)\left(\frac{\frac{1-\rho^{2l}}{1-\rho^2}+(l-1)t}{t}\right)^2 \ln\left(\frac{l}{\delta}\right) \sim 2\left(3+64e^{1/2}\right)l^2\ln l \tag{4.45}$$

as $l \to \infty$, which is much better than $n \sim l^3$ we (erroneously) got for our Theorem 4.2.1 but still worse than $n \sim l$ of Hanin and Nica (2020b).

4.3.3 The proof

We follow the same strategy as for the proof of our Theorem 4.2.1. However, as the model is simpler, the proof becomes much cleaner.

Proof. The proof is by induction on l.

Induction base. Recall $x_\beta = \hat{\psi}(g_\beta^1) - \hat{\psi}(0)$. We have

$$\mathcal{P}(x_\beta > t) \le \mathcal{P}\left(\lambda^2|g_\beta^1|^2 > t\right) = \mathcal{P}\left(|g_\beta^1| > \frac{\sqrt{t}}{\lambda}\right) = \operatorname{erfc}\left(\frac{\sqrt{t}}{\sqrt{2}\lambda}\right) \le e^{-\frac{t}{2\lambda^2}}. \tag{4.46}$$

Therefore $\kappa = \frac{1}{2\lambda^2}$. Compare to $\kappa = \frac{1}{\lambda^2\left(4+\nu^1\sqrt{\frac{\pi}{2}}\right)}$ in the general case.

We then bound the moments:

$$\mathbb{E}\left[|x|\right] \le \mathbb{E}\left[\lambda^2|g_\beta^1|^2\right] \le \lambda^2; \tag{4.47}$$

$$\mathbb{E}\left[x^2\right] \le \mathbb{E}\left[\lambda^4 |g_\beta^1|^4\right] \le 3\lambda^4. \tag{4.48}$$

Finally, we bound quantities containing q:

$$q \le \mathbb{E}\left[x^2\right] + \tfrac{16}{\kappa^2 e^{-\kappa \mathbb{E}|x|}} \le \lambda^4 \left(3 + 64 e^{1/2}\right). \tag{4.49}$$

$$\frac{q\kappa^2}{2} \ge 8 e^{-\kappa \mathbb{E}[x]} \ge 8 e^{-1/2}; \tag{4.50}$$

$$\frac{2\mathbb{E}[x]}{q\kappa} \le \frac{1}{8}\kappa \mathbb{E}[x] e^{\kappa \mathbb{E}[x]} \le \frac{e^{1/2}}{16} < 1. \tag{4.51}$$

Induction step. Recall $x_\beta = \hat{\psi}(\sqrt{v^l} z_\beta) - \hat{\psi}(0)$, where $z \sim \mathcal{N}(0, I_n)$:

Corollary 4.3.2. *Let $\mathring{c}^{l+1}(g^l) = \mathbb{E}_{z\sim\mathcal{N}(0,1)} \hat{\psi}(\sqrt{v^l} z)$. Then*

$$\forall n \in \mathbb{N},\ \tau > 0 \qquad \mathcal{P}\left(c^{l+1} - \mathring{c}^{l+1}(g^l) > \tau \,\middle|\, g^l\right) \le e^{-n\left(\frac{\tau}{\hat{C}_2^l}\right)^2} + n e^{-C_0 n}, \tag{4.52}$$

where $\hat{C}_2^l = \lambda^2 v^l \sqrt{2\left(3 + 64 e^{1/2}\right)}$, and we have the same bound for $\mathcal{P}(c^{l+1} - \mathring{c}^{l+1}(g^l) < -\tau)$.

Proof.

$$\mathcal{P}(x_\beta > t) \le \mathcal{P}\left(\lambda^2 v^l z_\beta^2 > t\right) = \mathcal{P}\left(|z_\beta| > \frac{\sqrt{t}}{\sqrt{v^l}\lambda}\right) = \operatorname{erfc}\left(\frac{\sqrt{t}}{\sqrt{2v^l}\lambda}\right) \le e^{-\frac{t}{2v^l\lambda^2}}. \tag{4.53}$$

Therefore $\kappa = \frac{1}{2v^l\lambda^2}$, not dependent on β. Compare to $\kappa_\beta = \frac{1}{\lambda^2 \eta_\beta^l}$ in the general case, where $\eta_\beta^l = \frac{(\nu^{l+1} + 2\|g_\beta^{1:l}\|_1)^2}{\theta\left(\frac{\nu^{l+1} + 2\|g_\beta^{1:l}\|_1}{2\sqrt{2v^l}}\right)}$.

We then bound the moments:

$$\mathbb{E}\left[|x|\right] \le \mathbb{E}\left[\lambda^2 v^l |z|^2\right] \le \lambda^2 v^l; \tag{4.54}$$

$$\mathbb{E}\left[x^2\right] \le \mathbb{E}\left[\lambda^4 (v^l)^2 |z|^4\right] \le 3\lambda^4 (v^l)^2. \tag{4.55}$$

Finally, we bound quantities containing q:

$$q \le \mathbb{E}\left[x^2\right] + \tfrac{16}{\kappa^2 e^{-\kappa \mathbb{E}|x|}} \le \lambda^4 (v^l)^2 \left(3 + 64 e^{1/2}\right). \tag{4.56}$$

$$\frac{q\kappa^2}{2} \ge 8 e^{-\kappa \mathbb{E}[x]} \ge 8 e^{-1/2}; \tag{4.57}$$

$$\frac{2\mathbb{E}[x]}{q\kappa} \le \frac{1}{8}\kappa \mathbb{E}[x] e^{\kappa \mathbb{E}[x]} \le \frac{e^{1/2}}{16} < 1. \tag{4.58}$$

Plugging all of the above into Lemma 4.2.3 gives the result. □

Let us perform the actual induction step:

$$\mathcal{P}(c^{l+1} - \mathring{c}^{l+1} > \tau) \le \inf_{\rho\in[0,\tau]} \inf_{\sigma\ge 0} \Bigg[\sup_{g^l:\, \begin{cases} \mathring{c}^{l+1}(g^l) - \mathring{c}^{l+1} \le \rho, \\ v^l - \mathring{v}^l \le \sigma \end{cases}} \mathcal{P}\left(c^{l+1} - \mathring{c}^{l+1} > \tau \,|\, g^l\right) + \mathcal{P}\left(v^l - \mathring{v}^l > \sigma\right) + \mathcal{P}\left(\mathring{c}^{l+1}(g^l) - \mathring{c}^{l+1} > \rho \,|\, v^l - \mathring{v}^l \le \sigma\right) \Bigg]. \tag{4.59}$$

The first term can be bounded using the above corollary, while the following term is bounded by the induction hypothesis:

$$\mathcal{P}(c^{l+1} - \mathring{c}^{l+1} > \tau) \le \inf_{\rho\in[0,\tau]} \inf_{\sigma\ge 0} \Big[e^{-n\left(\frac{\tau-\rho}{\bar{C}_2^l(\sigma)}\right)^2} + ne^{-C_0 n} + B^l \left(e^{-n\left(\frac{\sigma}{C_2^l(\sigma)}\right)^2} + ne^{-C_0 n} \right) + \mathcal{P}\left(\mathring{c}^{l+1}(g^l) - \mathring{c}^{l+1} > \rho \,|\, v^l - \mathring{v}^l \le \sigma\right) \Big] \qquad \forall \tau > 0,\ n \in \mathbb{N}, \tag{4.60}$$

where

$$\bar{C}_2^l(\sigma) = \lambda^2(\mathring{v}^l + \sigma)\sqrt{2\left(3 + 64e^{1/2}\right)}. \tag{4.61}$$

Let us bound the last term:

$$\mathcal{P}\left(\mathring{c}^{l+1}(g^l) - \mathring{c}^{l+1} > \rho \,\middle|\, v^l - \mathring{v}^l \le \sigma\right) \le \mathrm{Ind}\left(\sup_{v\in[0,\mathring{v}^l+\sigma]} \left(\mathbb{E}_z \hat{\psi}(\sqrt{v}z) - \mathring{c}^{l+1}\right) > \rho\right) = \mathrm{Ind}\left(\mathring{u}_\sigma^l - \mathring{c}^{l+1} > \rho\right), \tag{4.62}$$

where $\mathring{u}_\sigma^l = \sup_{v\in[0,\mathring{v}^l+\sigma]} \mathbb{E}_{z\sim\mathcal{N}(0,1)} \hat{\psi}(\sqrt{v}z)$.

Lemma 4.3.3. $\forall \sigma > 0\ \mathring{u}_\sigma^l \le \mathring{c}^{l+1} + \lambda^2\sigma$.

Proof. By pseudo-Lipschitzness,

$$\hat{\psi}(\sqrt{v}z) - \hat{\psi}(\sqrt{\mathring{v}^l}z) = \left(\hat{\chi}(\sqrt{v}z) + \hat{\chi}(\sqrt{\mathring{v}^l}z)\right)\left(\hat{\chi}(\sqrt{v}z) - \hat{\chi}(\sqrt{\mathring{v}^l}z)\right) \le \lambda^2 \left|v - \mathring{v}^l\right| |z|^2. \tag{4.63}$$

Taking the expectation and the supremum over $v \in [\mathring{v}^l, \mathring{v}^l + \sigma]$, we get

$$\sup_{v\in[\mathring{v}^l,\mathring{v}^l+\sigma]} \mathbb{E}_z \hat{\psi}(\sqrt{v}z) - \mathring{c}^{l+1} \le \lambda^2\sigma. \tag{4.64}$$

At the same time, $\sup_{v\in[0,\mathring{v}^l]} \left(\hat{\psi}(\sqrt{v}z) - \hat{\psi}(\sqrt{\mathring{v}^l}z)\right) = 0$ since $\hat{\chi}$ is non-decreasing. This gives $\mathring{u}_\sigma^l - \mathring{c}^{l+1} \le \lambda^2\sigma$. □

This lemma implies

$$\mathcal{P}\left(\mathring{c}^{l+1}(g^l) - \mathring{c}^{l+1} > \rho \,\middle|\, v^l - \mathring{v}^l \le \sigma\right) = 0 \qquad \forall \sigma \in \left[0, \frac{\rho}{\lambda^2}\right]. \tag{4.65}$$

Then since $\bar{C}_l^2$ is increasing and C_l^2 is increasing by the induction hypothesis,

$$\begin{aligned}\mathcal{P}(c^{l+1} - \mathring{c}^{l+1} > \tau) &\leq \inf_{\rho \in [0,\tau]} \left[e^{-n\left(\frac{\tau-\rho}{\bar{C}_2^l(\rho/\lambda^2)}\right)^2} + B^l e^{-n\left(\frac{\rho}{\lambda^2 C_2^l(\rho/\lambda^2)}\right)^2} + (B^l+1)ne^{-C_0 n} \right] \\ &\leq \inf_{\rho \in [0,\tau]} \left[e^{-n\left(\frac{\tau-\rho}{\bar{C}_2^l(\tau/\lambda^2)}\right)^2} + B^l e^{-n\left(\frac{\rho}{\lambda^2 C_2^l(\tau/\lambda^2)}\right)^2} + (B^l+1)ne^{-C_0 n} \right]\end{aligned} \tag{4.66}$$

$\forall \tau > 0,\ n \in \mathbb{N}$. We simply take ρ from the following linear equation:

$$\frac{\tau - \rho}{\bar{C}_2^l(\tau/\lambda^2)} = \frac{\rho}{\lambda^2 C_2^l(\tau/\lambda^2)}. \tag{4.67}$$

The solution is

$$\rho = \frac{\tau}{1 + \frac{1}{\lambda^2}\frac{\bar{C}_2^l(\tau/\lambda^2)}{C_2^l(\tau/\lambda^2)}}. \tag{4.68}$$

This finally gives

$$\mathcal{P}(c^{l+1} - \mathring{c}^{l+1} > \tau) \leq B^{l+1}\left(e^{-n\left(\frac{\tau}{C_2^{l+1}(\tau)}\right)^2} + ne^{-C_0 n} \right) \qquad \forall \tau > 0,\ n \in \mathbb{N}, \tag{4.69}$$

where

$$B^{l+1} = B^l + 1, \qquad C_2^{l+1}(\tau) = \lambda^2 C_2^l(\tau/\lambda^2) + \bar{C}_2^l(\tau/\lambda^2). \tag{4.70}$$

□

5 Feature learning in L_2-regularized DNNs: Attraction/repulsion and sparsity

Authors: Arthur Jacot, Eugene Golikov, Clément Hongler, Franck Gabriel.

Link and reference: Jacot et al. (2022), published at *Neural Information Processing Systems (NeurIPS)*[1].

Contribution: experiments, discussion.

5.1 Introduction

It is generally believed that the success of deep learning hinges on the ability of deep neural networks (DNNs) to learn features that are well suited to the task they are trained on. There is however little understanding of what these features are and how they are selected by the network.

On the other hand, recent results Jacot et al. (2018) have shown that it is possible to train DNNs without feature learning. This suggests the existence of two regimes of DNNs, a kernel regime (also called lazy or NTK regime) without feature learning and an active regime where features are learned. The presence or absence of feature learning can depend on multiple factors, such as the initialization/parametrization of DNNs Chizat et al. (2019); Yang and Hu (2021); Li et al. (2021); Jacot et al. (2021b), very large depths Hanin and Nica (2020a) or large learning rate Lewkowycz et al. (2020); Cohen et al. (2021).

In this paper, we focus on the impact of L_2 regularization on feature learning in DNNs. This analysis is further motivated by recent results Gunasekar et al. (2018a,b); Chizat and Bach (2020) which show that the implicit bias of gradient descent on losses such as the cross entropy (which decay exponentially towards infinity) is essentially the same as the bias induced by L_2 regularization in DNNs.

Generally, the bias induced by the addition of L_2-regularization on the parameters θ of a model f_θ can be described by the *representation cost* $R(f) = \min_{\theta: f_\theta = f} \|\theta\|^2$, since $\min_\theta C(f_\theta) + \lambda \|\theta\|^2 = \min_f C(f) + \lambda R(f)$.

In deep linear networks, the addition of L_2 regularization on the parameters corresponds to the addition of an L_p-Schatten norm regularization to the represented matrix, with $p = 2/L$ where L is the depth of the network Gunasekar et al. (2018b); Dai et al. (2021). This implies a sparsity effect

[1] https://proceedings.neurips.cc/paper_files/paper/2022/hash/2d2f85c0f93e69cf71f58eebaebb5e8d-Abstract-Conference.html

that increases with depth L.

In non-linear networks the sparsity effect of L_2-regularization has been described for shallow networks ($L = 2$) in Bach (2017); Ongie et al. (2020); Savarese et al. (2019) or for shallow non-linear networks with added linear layers Ongie and Willett (2022). Though it seems natural that this effect should become stronger for deeper networks, to our knowledge little theoretical work has been done in this area.

5.1.1 Contributions

In this paper, we study the minima of the loss of L_2 regularized fully-connected DNNs of depth L. We propose two reformulations of the loss:

1. The first reformulation expresses the loss in terms of the representations $Z_1, \ldots, Z_L$ (the layer pre-activations of every input in the training set) of every layer of the network. This reformulation has the advantage of being local - the optimal choice of a layer Z_ℓ only depends on its neighboring layers $Z_{\ell-1}$ and $Z_{\ell+1}$. The optimal choice of representation Z_ℓ is at the balance between an attractive force (determined by the previous layer) and a repulsive force (coming from the next layer). It illustrates how the representations $Z_1, \ldots, Z_{L-1}$ interpolates between the input layer Z_0 and output layer Z_L.

2. The second reformulation expresses the loss in terms of the covariances of the representation before applying the non-linearity $K_\ell = Z_\ell^T Z_\ell$ and after the non-linearity $K_\ell^\sigma = \left(Z_\ell^\sigma\right)^T Z_\ell^\sigma$. For positively homogeneous non-linearities and when the number of neurons n_ℓ in every hidden layer ℓ is larger or equal to $N(N+1)$ for N the number of datapoints, this reformulation is an optimization of a (partially convex) loss over the covariances $(K_1, K_1^\sigma), \ldots, (K_{L-1}, K_{L-1}^\sigma)$ and the outputs Z_L, restricted to a (translated) convex cone. This reformulation does not depend on the number of neurons n_ℓ in the hidden layers (as long as $n_\ell \geq N(N+1)$).

The second reformulation implies that for positively homogeneous non-linearities such as the ReLU, as the number of neurons in the hidden layers n_ℓ increase, the global minimum of the L_2-regularized loss goes down until reaching a plateau (i.e. adding neurons does not lead to an improvement in the loss). This illustrates the sparsity effect of L_2-regularization, where the optimum reached on a very large network is equivalent to a much smaller network.

The start of the plateau hence gives a measure of sparsity of the global minimum. We show that the minimal number of neurons n_ℓ to reach this plateau is determined by a notion of rank $\mathrm{Rank}_\sigma\left(K_\ell, K_\ell^\sigma\right)$ of the covariance pairs. We show that $\mathrm{Rank}_\sigma\left(K_\ell, K_\ell^\sigma\right) \leq N(N+1)$, i.e. the plateau must start before $N(N+1)$ and that the scaling of this upper bound is tight by giving an example dataset such that at the optimum $\mathrm{Rank}_\sigma\left(K_\ell, K_\ell^\sigma\right) \geq N^2/4$. We also present other datasets where the start of the plateau is either constant or grows linearly with the number of datapoints. We also observe empirically that the plateau can start at much smaller widths for real data such as MNIST and the teacher/student setting.

5.2 Setup

We consider fully-connected deep neural networks with $L+1$ layers, numbered from 0 (the input layer) to L (the output layer), with nonlinear activation function $\sigma : \mathbb{R} \to \mathbb{R}$ (e.g. the ReLU $\sigma(x) = \max\{0, x\}$)). Each layer ℓ contains n_ℓ neurons and we denote $\mathbf{n} = (n_1, \ldots, n_L)$ the widths of

the network. Given an input dataset $\{x_1, \dots, x_N\} \subset \mathbb{R}^{n_0}$ of size N, we consider the data matrix $X = (x_0, \dots, x_n) \in \mathbb{R}^{n_0 \times N}$, and encode the activations and preactivations of the whole data set by considering the pre-activations $Z_\ell(X; \mathbf{W}) \in \mathbb{R}^{n_\ell \times N}$ and activations $Z_\ell^\sigma(X; \mathbf{W}) \in \mathbb{R}^{(n_\ell+1) \times N}$ given by:

$$\begin{aligned} Z_0^\sigma(X; \mathbf{W}) &= \begin{pmatrix} X \\ \beta \mathbf{1}_N^T \end{pmatrix} \\ Z_\ell(X; \mathbf{W}) &= W_\ell Z_{\ell-1}^\sigma(X; \mathbf{W}) \\ Z_\ell^\sigma(X; \mathbf{W}) &= \begin{pmatrix} \sigma(Z_\ell) \\ \beta \mathbf{1}_N^T \end{pmatrix}, \end{aligned}$$

where $\mathbf{W} = (W_\ell)_{\ell=1,\dots,L}$ is the collection of $n_\ell \times (n_{\ell-1}+1)$-dim weight matrices W_ℓ, $\sigma(Z_\ell)$ is obtained by applying elementwise the nonlinearity σ to the matrix Z_ℓ, and the scalar $\beta \in \mathbb{R}$ represents the amount of bias (i.e. when $\beta = 0$ there is no bias, when $\beta = 1$ this definition is equivalent to the traditional definition of bias). The output of the network is the pre-activation of the L-th layer Z_L.

We often drop the dependence on the weights $\mathbf{W}$ and on the dataset X and simply write Z_ℓ and Z_ℓ^σ.

We denote $f_{\mathbf{W}} : \mathbb{R}^{n_0} \to \mathbb{R}^{n_L}$ the *network function*, which maps an input x to the pre-activation at the last layer.

5.2.1 L_2-Regularized Loss and Representation Cost

Given a general cost functional $C : \mathbb{R}^{n_L \times N} \to \mathbb{R}$, the L_2-regularized loss of DNNs of widths $\mathbf{n}$ is

$$\mathcal{L}_{\lambda,\mathbf{n}}(\mathbf{W}) = C(Z_L(X; \mathbf{W})) + \lambda \|\mathbf{W}\|^2,$$

where $\|\mathbf{W}\|$ is the L_2-norm of $\mathbf{W}$ understood as a vector. Note that $\|\mathbf{W}\|^2 = \sum_{\ell=1}^L \|W_\ell\|_F^2$ where $\|\cdot\|_F$ denotes the Frobenius norm. From now on, we often omit to specify the widths $\mathbf{n}$ and simply write $\mathcal{L}_\lambda$.

The additional regularization cost should bias the network toward low norm solutions. This bias on the parameters leads to a bias in function space, which is described by the so-called *representation cost* $\mathcal{R}_{\mathbf{n}}(f)$ defined on functions $f : \mathbb{R}^{n_0} \to \mathbb{R}^{n_L}$:

$$\mathcal{R}_{\mathbf{n}}(f) = \min_{\mathbf{W}: f_{\mathbf{W}} = f} \|\mathbf{W}\|^2,$$

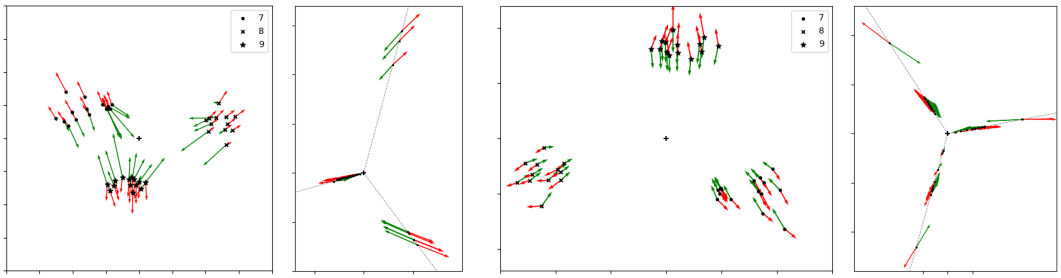
where the minimum is taken over all choices of parameters $\mathbf{W}$ of a width $\mathbf{n}$ network, with fixed β bias amount, such that the network function $f_{\mathbf{W}}$ equals f. By convention, if no such parameters exist then $\mathcal{R}_{\mathbf{n}}(f) = +\infty$.

Similarly, given an input-output pair $X \in \mathbb{R}^{n_0 \times N}$, $Y \in \mathbb{R}^{n_L \times N}$, the representation cost $R_{\mathbf{n}}(X, Y)$ is:

$$R_{\mathbf{n}}(X, Y) = \min_{\mathbf{W}: Z_L(X, \mathbf{W}) = Y} \|\mathbf{W}\|^2,$$

with again the convention that if there exists no weight $\mathbf{W}$ such that $Z_L(X, \mathbf{W}) = Y$, then $R_{\mathbf{n}}(X, Y) = +\infty$. The representation cost $R_{\mathbf{n}}$ naturally describes the bias induced by the L_2-regularized loss of DNNs since:

$$\min_{\mathbf{W}} C(Z_L(X; \mathbf{W})) + \lambda \|\mathbf{W}\|^2 = \min_Y C(Y) + \lambda R_{\mathbf{n}}(Y, X).$$



(a) Inputs ($\ell = 1$). (b) Neurons ($\ell = 1$). (c) Inputs ($\ell = 2$). (d) Neurons ($\ell = 2$).

Figure 5.1: **Attraction/Repulsion:** Visualization of the hidden representation Z_1 and Z_2 of a $L = 3$ ReLU DNN at the end of training (i.e. after $T = 20k$ steps of gradient descent on the original loss $\mathcal{L}_\lambda$) on 3 digits (7,8 and 9) of MNIST LeCun et al. (1998) along with the attraction force in green and repulsion force in red (both forces are approximated with Tikhonov regularization). For both layers, we plot in **(a)** and **(c)** the PCA of the $N = 42$ lines (each corresponding to a datapoint) and in **(b)** and **(d)** the PCA the $n_1 = n_2 = 50$ columns (each corresponding to a neuron in the layer $\ell = 1$ or $\ell = 2$). We observe a clustering of the inputs according to their digit, and of the neurons along 3 rays in grey dashed lines.

5.3 Two Reformulations of the Regularized Loss: Hidden Representation and Covariance Optimization

We now provide two reformulations of the L_2-regularized loss $\mathcal{L}_\lambda(\mathbf{W})$ and representation cost $R_{\mathbf{n}}(X, Y)$, which both put emphasis on the hidden representations Z_ℓ and how they are progressively modified throughout the neural network. The first reformulation holds for general non-linearities while the second only applies to networks with homogeneous nonlinearities.

5.3.1 Feature optimization : attraction/repulsion

The key observation is that the weights W_ℓ can be decomposed as follows:

$$W_\ell = Z_\ell \left(Z_{\ell-1}^\sigma\right)^+ + \tilde{W}_\ell,$$

where the residual matrix $\tilde{W}_\ell$ is orthogonal to $Z_{\ell-1}^\sigma$, i.e. $\tilde{W}_\ell Z_{\ell-1}^\sigma = 0$, and $(\cdot)^+$ is the Moore-Penrose pseudo-inverse. This stems from the fact that $W_\ell = W_\ell \mathrm{P}_{\mathrm{Im} Z_{\ell-1}^\sigma} + \tilde{W}_\ell$ where $\tilde{W}_\ell := W_\ell \mathrm{P}_{(\mathrm{Im} Z_{\ell-1}^\sigma)^\perp}$, and $\mathrm{P}_{\mathrm{Im} Z_{\ell-1}^\sigma}$, resp. $\mathrm{P}_{(\mathrm{Im} Z_{\ell-1}^\sigma)^\perp}$, is the orthogonal projection on $\mathrm{Im} Z_{\ell-1}^\sigma$, resp. on the orthogonal complement of $\mathrm{Im} Z_{\ell-1}^\sigma$; one concludes using the facts that $\mathrm{P}_{\mathrm{Im} Z_{\ell-1}^\sigma} = Z_{\ell-1}^\sigma \left(Z_{\ell-1}^\sigma\right)^+$ and $Z_\ell = W_\ell Z_{\ell-1}^\sigma$.

Note that the matrix $\tilde{W}_\ell$ does not affect either the hidden representations Z_ℓ nor the output Z_L. Besides, the Frobenius norm of W_ℓ can be rewritten as $\|W_\ell\|_F^2 = \|Z_\ell \left(Z_{\ell-1}^\sigma\right)^+\|_F^2 + \|\tilde{W}_\ell\|_F^2$. When minimizing the L_2-regularized cost, it is therefore always optimal to consider null residual matrices $\tilde{W}_\ell = 0$, resulting in a reformulation of the cost which only depends on the pre-activations Z_ℓ:

Proposition 5.3.1. *The infimum of* $\mathcal{L}_\lambda(\mathbf{W}) = C(Z_L(X;\mathbf{W})) + \lambda \|\mathbf{W}\|^2$, *over the parameters* $\mathrm{W} \in \mathbb{R}^P$ *is equal to the infimum of*

$$\mathcal{L}^r_\lambda(Z_1,\dots,Z_L) = C(Z_L) + \lambda \sum_{\ell=1}^{L} \left\| Z_\ell \left(Z^\sigma_{\ell-1}\right)^+ \right\|^2_F$$

over the set $\mathcal{Z}$ *of hidden representations* $\mathbf{Z} = (Z_\ell)_{\ell=1,\dots,L}$ *such that* $Z_\ell \in \mathbb{R}^{n_\ell \times N}$, $\mathrm{Im} Z^T_{\ell+1} \subset \mathrm{Im}\left(Z^\sigma_\ell\right)^T$, *with the notations* $Z^\sigma_0 = \begin{pmatrix} X \\ \beta \mathbf{1}^T_N \end{pmatrix}$ *and* $Z^\sigma_\ell = \begin{pmatrix} \sigma(Z_\ell) \\ \beta \mathbf{1}^T_N \end{pmatrix}$.

Furthermore, if $\mathbf{W}$ *is a local minimizer of* $\mathcal{L}_\lambda$ *then* $(Z_1(X;\mathbf{W}),\dots,Z_L(X;\mathbf{W}))$ *is a local minimizer of* $\mathcal{L}^r_\lambda$. *Conversely, keeping the same notations, if* $(Z_\ell)_{\ell=1,\dots,L}$ *is a local minimizer of* $\mathcal{L}^r_\lambda$, *then* $\mathrm{W} = (Z_\ell (Z^\sigma_{\ell-1})^+)_{\ell=1,\dots,L}$ *is a local minimizer of* $\mathcal{L}_\lambda$.

Note that one can also reformulate the representation cost:

$$R_{\mathbf{n}}(X,Y) = \min_{\mathbf{Z}\in\mathcal{Z}, Z_L = Y} \sum_{\ell=1}^{L} \left\| Z_\ell \left(Z^\sigma_{\ell-1}\right)^+ \right\|^2_F .$$

The representation in terms of the output and hidden representations have several interesting properties, especially when it comes to minimization:

1. The optimization becomes local in the sense that all terms and constraints depend either only on the output cost $C(Z_L)$ or on two neighboring terms (e.g. $\left\| Z_\ell \left(Z^\sigma_{\ell-1}\right)^+ \right\|^2_F$). As a result, the (projected) gradient of the loss $\mathcal{L}^r_\lambda(Z_1,...,Z_L)$ w.r.t. to Z_ℓ only depends on $Z_{\ell-1}, Z_\ell$ and $Z_{\ell+1}$. This is in contrast to the optimization of $\mathcal{L}_\lambda(\mathbf{W})$, where the gradient of $C(Z_L)$ with respect to W_ℓ depends on all parameters $W_1,\dots,W_L$.

2. The value $\left\| Z_\ell \left(Z^\sigma_{\ell-1}\right)^+ \right\|^2_F$ represents a 'multiplicative distance' between Z_ℓ and $Z^\sigma_{\ell-1}$ (in contrast to the 'additive distance' $\left\| Z_\ell - Z^\sigma_{\ell-1} \right\|^2_F$); the representation Z_ℓ therefore interpolates multiplicatively between $Z_{\ell-1}$ and $Z_{\ell+1}$. This is most obvious for linear networks (i.e. $\sigma = \mathrm{id}$ and $\beta = 0$): in this case, one can check that at any global minimizer, the covariances of the hidden layers equal $Z^T_\ell Z_\ell = X^T (X^{-T} Z^T_L Z_L X^{-1})^{\frac{\ell}{L}} X$, interpolating between the input covariance $X^T X$ and output covariance $Z^T_L Z_L$.

3. A lot of work has been done to propose biologically plausible training methods for DNNs Bengio et al. (2015); Illing et al. (2019), in contrast to backpropagation which is not local. A line of work Pehlevan et al. (2017); Obeid et al. (2019); Qin et al. (2021), propose a biologically plausible optimization technique which minimizes a cost which closely resembles our first reformulation, with the multiplicative distances $\left\| Z_\ell \left(Z^\sigma_{\ell-1}\right)^+ \right\|^2_F$ replaced by additive ones $\left\| Z_\ell - Z^\sigma_{\ell-1} \right\|^2_F$. Due to this change, there is no direct correspondence between the networks trained with this biologically plausible technique and those trained with backpropagation. If one could extend this training technique to work with multiplicative distances one could guarantee such a direct correspondence.

4. The optimization leads to an attraction-repulsion algorithm. If we optimize only on the term Z_ℓ and fix all other representations, the only two terms that depend on Z_ℓ are $\left\|Z_\ell \left(Z^\sigma_{\ell-1}\right)^+\right\|^2_F$ and $\left\|Z_{\ell+1} \left(Z^\sigma_\ell\right)^+\right\|^2_F$. The former term is attractive as it pushes the representations Z_ℓ towards the origin (and hence pushes the representations at depth ℓ of every input towards each other), especially along directions where $Z^\sigma_{\ell-1}$ is small. The latter term is repulsive as it pushes the representations Z^σ_ℓ away from the origin, especially along directions where $Z_{\ell+1}$ is large.
5. This attraction-repulsion process is similar to the Information Bottleneck theory Tishby and Zaslavsky (2015): the repulsive term ensure that Z_ℓ keeps enough information about the inputs to reconstruct $Z_{\ell+1}$, while the attractive term pushes Z_ℓ to keep as little information as possible.

The attraction and repulsion forces of of the ℓ-th layer are the derivative $\partial_{Z_\ell}\left\|Z_\ell \left(Z^\sigma_{\ell-1}\right)^+\right\|^2_F$ and $\partial_{Z_\ell}\left\|Z_{\ell+1} \left(Z^\sigma_\ell\right)^+\right\|^2_F$ which are both $n_\ell \times N$ matrices. One can visualize these forces either column by column (each column corresponding to a datapoint $i = 1, \dots, N$) or line by line (each line corresponding to a neuron $k = 1, \dots, n_\ell$). These two visualizations of the forces are presented in Figure 5.1 for the two hidden layers of a depth $L = 3$ network, projected to the 2 largest principal components of the columns resp. lines of Z_ℓ. Figures 5.1a and 5.1c illustrate how the inputs corresponding to different classes are pushed away from each other, leading to a clustering effect. Figures 5.1b and 5.1d show that the neurons naturally align along rays starting from the origin. This happens for homogeneous non-linearities, such as the ReLU in this example, because if two neurons k, k' have proportional activations, i.e. $Z_{\ell,k} = \alpha Z_{\ell,k}$ for some $\alpha \in \mathbb{R}$, then their attractive and repulsive forces will also be proportional with the same scaling α. As a result, the neuron k is stable, i.e. the attraction and repulsion cancel each other, if and only if the neuron k' is stable.

This phenomenon can be interpreted as a form of sparsity: a group of aligned neurons can be replaced by a single neuron without changing the resulting function Z_L. Can we guarantee a degree of sparsity in the hidden representations? Can we bound the number of aligned groups in a neuron? In the next section, we introduce a further reformulation of the loss which allows us to partially answer these questions.

5.3.2 Covariance learning : partial convex optimization for positively homogeneous nonlinearities

The loss of the first reformulation $\mathcal{L}^r_\lambda$ depends on the hidden representations Z_ℓ and Z^σ_ℓ only through the covariances $K_\ell = Z^T_\ell Z_\ell$ and $K^\sigma_\ell = \left(Z^\sigma_\ell\right)^T Z^\sigma_\ell$, since $\left\|Z_\ell \left(Z^\sigma_{\ell-1}\right)^+\right\|^2_F = \mathrm{Tr}\left[K_\ell \left(K^\sigma_{\ell-1}\right)^+\right]$. Hence, we provide a second reformulation expressed in terms of the tuple of covariance pairs $\mathbf{K} = ((K_1, K^\sigma_1), \dots, (K_{L-1}, K^\sigma_{L-1}))$ and the outputs Z_L. Using the notations $K^\sigma_0 = X^T X + \beta^2 \mathbf{1}_{N\times N}$ and $K_L = Z^T_L Z_L$, we define:

$$\mathcal{L}^{\mathrm{k}}_\lambda(\mathbf{K}, Z_L) = C(Z_L) + \lambda \sum_{\ell=1}^{L} \mathrm{Tr}\left[K_\ell \left(K^\sigma_{\ell-1}\right)^+\right].$$

It remains to identify the set $\mathcal{K}_{\mathbf{n}}(X)$ of covariances $\mathbf{K}$ and outputs Z_L which can be represented by a width $\mathbf{n}$ network with inputs X. For positively homogeneous nonlinearities of degree 1 such as

the ReLU (i.e. when $\sigma(\lambda x) = \lambda\sigma(x)$ for any positive λ), the set $\mathcal{K}_{\mathbf{n}}(X)$ can be expressed using the notion of conical hulls.

Definition 5.3.2. The conical hull of $\Omega \subset \mathbb{R}^d$ is the set $\text{cone}(\Omega) := \left\{\sum_{i=1}^k \alpha_i\omega_i : k \geq 0, \alpha_i \geq 0, \omega_i \in \Omega\right\}$ and its m-conical hull for $m \geq 1$ is the set $\text{cone}_m(\Omega) := \{\sum_{i=1}^m \alpha_i\omega_i : \alpha_i \geq 0, \omega_i \in \Omega\}$.

Note that by Caratheodory's theorem for conical hulls, $\text{cone}_m(\Omega) = \text{cone}(\Omega)$ for any $m \geq d$. We now proceed to the description of the set $\mathcal{K}_{\mathbf{n}}(X)$ and obtain the second formulation of the L_2 regularized loss and of the representation loss R_n in terms of covariances:

Proposition 5.3.3. *For positively homogeneous non-linearities σ, the infimum of $\mathcal{L}_\lambda(\mathbf{W}) = C(Z_L(X;\mathbf{W})) + \lambda\|\mathbf{W}\|^2$, over the parameters $\mathrm{W} \in \mathbb{R}^P$ is equal to the infimum over $\mathcal{K}_{\mathbf{n}}(X)$ of*

$$\mathcal{L}^{\mathrm{k}}_\lambda(\mathbf{K}, Z_L) = C(Z_L) + \lambda\sum_{\ell=1}^{L} \mathrm{Tr}\left[K_\ell\left(K^\sigma_{\ell-1}\right)^+\right].$$

The set $\mathcal{K}_{\mathbf{n}}(X)$ is the set of covariances $\mathbf{K} = ((K_1, K_1^\sigma), \ldots, (K_{L-1}, K_{L-1}^\sigma))$ and outputs Z_L such that for all hidden layer $\ell = 1, \ldots, L-1$:

- *the pair (K_ℓ, K_ℓ^σ) belongs to the (translated) n_ℓ-conical hull*

$$S_{n_\ell} = \text{cone}_{n_\ell}\left(\left\{\left(xx^T, \sigma(x)\sigma(x)^T\right) : x \in \mathbb{R}^N\right\}\right) + (0, \beta^2\mathbf{1}_{N\times N}),$$

- $\mathrm{Im}K_\ell \subset \mathrm{Im}K^\sigma_{\ell-1}$, *with the notation $K_0^\sigma = X^TX + \beta^2\mathbf{1}_{N\times N}$, and* $\mathrm{Im}Z_L \subset \mathrm{Im}K^\sigma_{L-1}$.

Note that one can also reformulate the representation cost:

$$R_{\mathbf{n}}(X, Y) = \min_{\mathbf{K}:(\mathbf{K},Y)\in\mathcal{K}_{\mathbf{n}}(X)} \sum_{\ell=1}^{L} \mathrm{Tr}\left[K_\ell\left(K^\sigma_{\ell-1}\right)^+\right].$$

Remark 5.3.4. In contrast to the first loss $\mathcal{L}^r_\lambda$ whose local minima were in correspondence with the local minima of the original loss $\mathcal{L}_\lambda$, the second loss $\mathcal{L}^{\mathrm{k}}_\lambda$ can in some cases have strictly less critical points and local minima. Indeed, since the map $\mathbf{W} \mapsto (\mathbf{K}, Z_L)$ is continuous, if $(\mathbf{K}(\mathrm{W}), Z_L(\mathbf{W}))$ is a local minimum, then so is $\mathbf{W}$. However, the converse is not true: we provide a counterexample in the Appendix, i.e. a set of weights $\mathbf{W}$ of a depth $L = 2$ network which is a local minimum of $\mathcal{L}_\lambda$ and such that the corresponding $(\mathbf{K}, Z_L)$ is not a local minimum of $\mathcal{L}^{\mathrm{k}}_\lambda$.

Since the dimension of the space of pairs of symmetric $N\times N$ matrices is $N(N+1)$, if $n_\ell \geq N(N+1)$ then, by the Caratheodory's theorem for conical hulls,

$$S_{n_\ell} = S := \text{cone}\left(\left\{\left(xx^T, \sigma(x)\sigma(x)^T\right) : x \in \mathbb{S}^{N-1}\right\}\right) + (0, \beta^2\mathbf{1}_{N\times N}).$$

Hence, as soon as $n_\ell \geq N(N+1)$ for all hidden layers, the set $\mathcal{K}_{\mathbf{n}}(X)$ does not depend on the list of widths $\mathbf{n}$. We denote

$$\mathcal{K}(X) = \left\{(\mathbf{K}, Z_L) \mid \forall \ell = 1, \ldots, L-1, (K_\ell, K_\ell^\sigma) \in S, \mathrm{Im}K_\ell \subset \mathrm{Im}K^\sigma_{\ell-1}, \mathrm{Im}Z_L \subset \mathrm{Im}K^\sigma_{L-1}\right\}$$

this width-independent set. The following proposition shows that for sufficiently wide networks with a positively homogeneous nonlinearity, training a deep DNN with L_2 regularization is equivalent to a partially convex optimization over a translated convex cone.

Proposition 5.3.5. *The set $\mathcal{K}(X)$ is a translated convex cone: after the suitable translation, it is equal to its conical hull. The cost $\mathcal{L}^{\mathrm{k}}_\lambda(\mathbf{K}, Z_L)$ is partially convex w.r.t. to the outputs Z_L and the pairs (K_ℓ, K_ℓ^σ), i.e. it is convex if one fixes the other parameters and let only (K_ℓ, K_ℓ^σ), or Z_L, vary.*

5.3.3 Direct optimization of the reformulations

It is natural at this point to wonder whether one could optimize directly over the representations $\mathbf{Z}$ (using the first reformulation) or over the covariances $\mathbf{K}$ and output Z_L (using the second reformulation), and whether this would have an advantage over the traditional optimization of the weights.

For the first reformulation, one can simply use the projected gradient descent with updates given by

$$\mathbf{Z}_{t+1} = P_{\mathcal{Z}}\left(\mathbf{Z}_t - \eta \nabla \mathcal{L}^r_\lambda(\mathbf{Z})\right)$$

for any projection $P_{\mathcal{Z}}$ to the constraint space $\mathcal{Z}$. For example, a projection is obtained by mapping Z_ℓ to $Z_\ell P_{\mathrm{Im} Z^\sigma_{\ell-1}}$ sequentially from $\ell = 1$ to $\ell = L$. Note however that the loss explodes as the constraints become unsatisfied so that for gradient flow, there is no need for the projections. This suggests that these projections might also be unnecessary as long as the learning rate is small enough. For more details, see Appendix C.2.1.

For the second reformulation, there is no obvious way to compute a projection to the constraint space $\mathcal{K}$: the cone S is spanned by an infinite amount of points and we do not have an explicit formula for the dual cone S^*. Frank-Wolfe optimization can be used to overcome the need for computing the projections.

However, these direct optimizations of the reformulations lead to issues of computational complexity and stability. First, the computation of the gradients $\nabla \mathcal{L}^r_\lambda(\mathbf{Z})$ and $\nabla \mathcal{L}^{\mathrm{k}}_\lambda(\mathbf{K}, Z_L)$ requires solving a linear equation of dimension N, which is very costly, in contrast to the traditional optimization of the weights $\mathbf{W}$ for which the gradient can be computed very efficiently. Second, if Z^σ_ℓ is not full-rank, the computation of its pseudo-inverse $(Z^\sigma_\ell)^+$ and the projection $P_{\mathrm{Im} Z^\sigma_\ell}$ are very unstable. Therefore, if we only have finite-precision knowledge of Z_ℓ, we cannot reliably compute $(Z^\sigma_\ell)^+$ nor $P_{\mathrm{Im} Z^\sigma_\ell}$.

Although it could be possible to solve these problems (e.g. using the Tikhonov regularization for the unstability problem) and to develop efficient algorithms to optimize both reformulations efficiently, we decided in this paper to focus on the theoretical implications of these reformulations.

5.4 Sparsity of the Regularized Optimum for Homogeneous DNNs

In this section, we assume that the non-linearity is positively homogenous. Under this assumption, the second reformulation of the loss (and of the representation cost) holds and implies the existence of a sparsity phenomenon.

First observe that as the widths $\mathbf{n}$ increase, both the global minimizer of the loss $\min_{\mathbf{W}} \mathcal{L}_\lambda(\mathbf{W})$ and the representation cost $R_{\mathbf{n}}(X, Y)$ diminish. We denote by $\mathcal{L}_{\lambda,\mathbf{n}}$ the L_2-regularized loss of DNNs with widths n. Recall that the depth L is fixed.

Proposition 5.4.1. *If $\mathbf{n} \leq \mathbf{n}'$ (in the sense that $n_\ell \leq n'_\ell$ for all ℓ and $n_0 = n'_0$ and $n_L = n'_L$), then*

$$\min_{\mathbf{W} \in \mathbb{R}^{P_{\mathbf{n}}}} \mathcal{L}_{\lambda,\mathbf{n}}(\mathbf{W}) \geq \min_{\mathbf{W} \in \mathbb{R}^{P_{\mathbf{n}'}}} \mathcal{L}_{\lambda,\mathbf{n}'}(\mathbf{W})$$

and for any $X \in \mathbb{R}^{n_0 \times N}$ and $Y \in \mathbb{R}^{n_L \times N}$, $R_{\mathbf{n}}(X, Y) \geq R_{\mathbf{n}'}(X, Y)$.

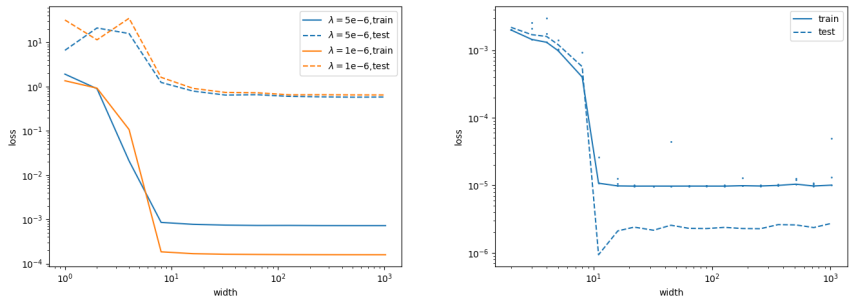


Figure 5.2: **Loss plateau:** Plots of the train loss (full lines) and test loss (dashed lines) as a function of the width for depth $L = 3$ DNNs for different datasets: (left) cross-entropy loss for a subset of MNIST ($N = 1000$) and two values of λ; (right) MSE with $\lambda = 10^{-6}$ on $N = 1000$ Gaussian inputs and outputs evaluated on a fixed teacher network of depth $L = 3$ and width 10((right). For the right plot we took (the minimum is taken over 3 independent trials, represented by the small blue dots). In both settings, the plateau appears to start around 10, much earlier than $N^2 = 10^6$. The regularization term is included in the training loss but not the test, leading to a smaller test loss on the right.

Proof. Let us assume that the parameters $\mathbf{W}^*$ are optimal for a width $\mathbf{n}$ network, the parameters can be mapped to parameters of a wider network by adding 'dead' neurons (i.e. neurons with zero incoming and outcoming weights) without changing the network function $f_{\mathbf{W}^*}$ nor the norm of the parameters $\|W\|$. □

For DNNs with positively homogeneous nonlinearities, a direct consequence of our reformulation through the hidden covariances is that both the global minimum of the loss $\mathcal{L}_\lambda$ and the representation cost $R_{\mathbf{n}}(X, Y)$ plateau for any widths $\mathbf{n}$ such that $n_\ell \geq N(N+1)$.

Proposition 5.4.2. *For any positively homogeneous nonlinearity σ, any widths $\mathbf{n}$ and $\mathbf{n}'$ such that $n_0 = n'_0$, $n_L = n'_L$ and for all $\ell = 1, \dots, L-1$, $n_\ell, n'_\ell \geq N(N+1)$, for all $\lambda > 0$, we have:*

$$\min_{\mathbf{W} \in \mathbb{R}^{P_{\mathbf{n}}}} \mathcal{L}_{\lambda,\mathbf{n}}(\mathbf{W}) = \min_{\mathbf{W} \in \mathbb{R}^{P_{\mathbf{n}'}}} \mathcal{L}_{\lambda,\mathbf{n}'}(\mathbf{W}).$$

Under the same conditions $R_{\mathbf{n}}(X, Y) = R_{\mathbf{n}'}(X, Y)$ for any $X \in \mathbb{R}^{n_0 \times N}$ and $Y \in \mathbb{R}^{n_L \times N}$,.

Proof. This follows directly from the second reformulation and the fact that by Caratheodory's theorem for conical hulls, $S_n = S$ if $n \geq N(N+1)$ (see discussion after Remark 5.3.4). □

We can therefore define a width-independent representation cost $R(X, Y)$ (which still depends on the fixed depth L) equal to the representation cost $R_{\mathbf{n}}(X, Y)$ of any sufficiently wide network.

5.4.1 Rank of the Hidden Representations

Now that we have revealed the plateau phenomenon, a natural question that we investigate in this section is when does this plateau begin. In order to do so, we introduce the notion of rank

$\text{Rank}_\sigma(K, K^\sigma)$ of a pair of Gram matrices $(K, K^\sigma) \in S$ which is the minimal number k such that

$$K = \sum_{i=1}^{k} z_i z_i^T \text{ and } K^\sigma = \sum_{i=1}^{k} \sigma(z_i)\,\sigma(z_i)^T + \beta^2 \mathbf{1}_{N\times N} \tag{5.1}$$

for some $z_1, \dots, z_k \in \mathbb{R}^N$. This notion of rank describes exactly the minimal number of neurons required to recover a set of covariances $\mathbf{K}$:

Proposition 5.4.3. *Let $(\mathbf{K}, Z_L) \in \mathcal{K}(X)$, then there are parameters $\mathbf{W}$ of a width $\mathbf{n}$ network with covariances and outputs $\mathbf{K}$ if and only if $n_\ell \geq \text{Rank}_\sigma(K_\ell, K_\ell^\sigma)$ for all $\ell = 1, \dots, L-1$.*

We can now describe the plateau $R = \{\mathbf{n} : \min_{\mathbf{W}\in\mathbb{R}^{P_\mathbf{n}}} \mathcal{L}_{\lambda,\mathbf{n}}(\mathbf{W}) = \min_{\mathbf{m}} \min_{\mathbf{W}\in\mathbb{R}^{P_\mathbf{m}}} \mathcal{L}_{\lambda,\mathbf{m}}(\mathbf{W})\}$, i.e. the set of widths $\mathbf{n}$ such that the minimum $\min_{\mathbf{W}\in\mathbb{R}^{P_\mathbf{n}}} \mathcal{L}_{\lambda,\mathbf{n}}(\mathbf{W})$ is optimal over all possible widths:

Corollary 5.4.4. *Let $K_{\min}$ be the set of covariances sequences $(\mathbf{K}, Z_L)$ which are global minima of the second reformulation. We have that $\mathbf{n} \in R$ if and only if there is a $(\mathbf{K}, Z_L) \in K_{\min}$ such that $n_\ell \geq \text{Rank}_\sigma(K_\ell, K_\ell^\sigma)$.*

Hence, the investigation of $\text{Rank}_\sigma(\cdot,\cdot)$ is crucial to understand where the plateau begins; unfortunately, it can be difficult to compute. However, from its definition and the Caratheodory's theorem for conical hulls (see our discussion after Remark 5.3.4), we have the following natural bounds:

Lemma 5.4.5. *For any pair $(K, K^\sigma) \in S$, we have* $\text{Rank}(K) \leq \text{Rank}_\sigma(K, K^\sigma) \leq N(N+1)$.

We show in the next section that the order of magnitude of the upper bound is tight. More specifically, we construct a dataset for which any global optimum satisfies $\text{Rank}_\sigma(K_1, K_1^\sigma) \geq N^2/4$. This implies that, in this example, the plateau transition occurs when the number of hidden neurons is of order $O(N^2)$. Note however that, in our numerical experiments (see Figure 5.2), the rank of the global optimum can be much smaller for more traditional dataset such as MNIST.

Remark 5.4.6. The start of the plateau measures a notion of sparsity of the learned network, since the networks learned in the plateau are equivalent to a network at the start of the plateau, i.e. large networks are equivalent in terms of their covariances and outputs $(\mathbf{K}, Z_L)$ to a (potentially much) smaller network.

Even though the set of pairs (K_ℓ, K_ℓ^σ) in the cone S that are not full rank has measure zero, the optimal representations (K_ℓ, K_ℓ^σ) always lie on the border of the cone S (since the derivative of the cost $\mathcal{L}_\lambda^k$ w.r.t. (K_ℓ, K_ℓ^σ) never vanishes) where the rank is lower. More precisely, the rank is determined by the dimension of the smallest face that contains the optimum (e.g. the pairs (K_ℓ, K_ℓ^σ) on the edges of S have rank at most 2 for example, while those on the vertices are rank 1).

We can identify different degrees of sparsity depending on how the rank of the hidden representations scales with the number of datapoints N: if the rank is $o(N)$ the covariances K_ℓ, K_ℓ^σ are low-rank (in the traditional linear sense) and for shallow networks the effective number of parameters (i.e. the number of parameters at the start of the plateau) is $o(N^2)$, if the rank is $o(\sqrt{N})$ then for deep networks the effective number of parameters $o(N)$. This could explain why very large networks with 'too many parameters' are able to generalize, since their effective number of parameters is of the order of the number datapoints. Very large networks can therefore be trained safely knowing that thanks to L_2-regularization, the network is able to recognize what is the 'right' width of the network.

5.4.2 Tightness of the Upper-Bound

In this section, we construct a pair of input and output datasets X and Y, both in $\mathbb{R}^{N\times N}$, such that for any optimal parameters $\mathbf{W}$ of a ReLU network of depth $L=2$ with no bias ($\beta=0$), the rank of the hidden representation $\mathrm{Rank}_\sigma(K_1, K_1^\sigma)$ is greater than ${}^{N^2}/_4$.

Note that one can write the decomposition (5.1) as $K = C^TC$ and $K_\sigma = B^TB$ where $C = (z_1,\dots,z_k)$ and $B = \mathrm{ReLU}(C)$ is obtained by applying elementwise the ReLU to C. Key to our construction is the fact that B is then a matrix with non-negative entries: the matrix K_σ is completely positive and $\mathrm{Rank}_\sigma(K, K^\sigma)$ can be studied using the CP-rank of K:

Definition 5.4.7. A $N\times N$ matrix A is completely positive if $A = B^TB$ for a $k\times N$ matrix B with non-negative entries. The CP-rank $\mathrm{Rank}_{\mathrm{cp}}(A)$ of a completely positive matrix A is the minimal integer k such $A = B^TB$ for a $k\times N$ matrix B with non-negative entries.

When σ is the ReLU, the kernel K_ℓ^σ is completely positive for all hidden layers ℓ, and thus

$$\max\left(\mathrm{Rank}\left(K_\ell\right), \mathrm{Rank}_{cp}\left(K_\ell^\sigma\right)\right) \leq \mathrm{Rank}_\sigma(K, K^\sigma).$$

In order to obtain the tightness of the upper bound, we proceed in two steps: first, we construct a completely positive matrix A with high CP-rank, and then construct inputs X and outputs Y such that the optimal hidden covariance $K_1 = K_1^\sigma$ for a depth $L=2$ network equals the matrix A.

As shown in Drew et al. (1994), bi-partite graphs can be used to construct matrices with high CP-rank. We refine this by showing that graphs on N vertices without cliques of 3 or more vertices lead to $N\times N$ matrices with CP-rank equal to the number of edges, and as a corollary, we construct a completely positive matrix with CP-rank equal to ${}^{N^2}/_4$.

Proposition 5.4.8. *Given a graph G with N vertices and k edges, consider the $k\times N$ matrix E with entries $E_{ev}=1$ if the vertex v is an endpoint of the edge e and $E_{ev}=0$ otherwise. The matrix $A = E^TE$ is completely positive and if the graph G contains no cliques of 3 or more vertices then $\mathrm{Rank}_{\mathrm{cp}}(A) = k$.*

Hence, to obtain a completely positive matrix of high CP-rank, it remains to find a graph with no cliques and as many edges as possible. For even N, we consider the complete bipartite graph, i.e. the graph with two groups of size $N/2$ and with edges between any two vertices iff they belong to different groups. For this graph, the matrix $B_N = E^TE$ takes the form of a block matrix:

$$B_N = \begin{pmatrix} \frac{N}{2}I_{\frac{N}{2}} & \mathbf{1}_{\frac{N}{2}\times\frac{N}{2}} \\ \mathbf{1}_{\frac{N}{2}\times\frac{N}{2}} & \frac{N}{2}I_{\frac{N}{2}} \end{pmatrix}$$

where $\mathbf{1}_{\frac{N}{2}}$ is the $N/2\times N/2$ matrix with all ones entries. Since this bipartite graph has no cliques and ${}^{N^2}/_4$ edges, from the previous proposition, we obtain $\mathrm{Rank}_{\mathrm{cp}}(B_N) = \frac{N^2}{4}$.

The following proposition shows how for any completely positive matrix (with CP-rank k) there is a dataset such that a shallow ReLU network will have a hidden representation pair $(K_1, K_1^\sigma)$of rank k:

Proposition 5.4.9. *Consider a width-*n *shallow network ($L=2$) with ReLU activation, no bias $\beta=0$, $n_0=N$, $n_1\geq N(N+1)$, input dataset $X_N = I_N$, and any output dataset Y_N such that $\left(Y_N^TY_N\right)^{\frac{1}{2}}$ is a completely positive matrix with CP-rank k.*

At any global minimum of $R_{\mathrm{n}}(X_N, Y_N)$*, we have* $\mathrm{Rank}_\sigma\left(K_1, K_1^\sigma\right) = k$*. Furthermore for* λ *small enough, at any global minimum of* $\mathcal{L}^{\mathrm{MSE}}_{\lambda,\mathrm{n}}(\mathbf{W}) = \frac{1}{N}\left\|Y(X_N;\mathbf{W}) - Y_N\right\|_F^2 + \lambda\left\|\mathbf{W}\right\|^2$*, we have* $\mathrm{Rank}_\sigma\left(K_1, K_1^\sigma\right) \geq k$*.*

By Proposition 5.4.9, with the outputs $Y_N = B_N$, the rank of the hidden representations (and the start the plateau) is larger or equal to $\frac{N^2}{4}$. This shows that the order N^2 of the bound of Lemma 5.4.5 is tight when it comes to data-agnostic bounds. However under certain assumptions on the data one can guarantee a much earlier plateau.

For example, if we instead apply Proposition 5.4.9 to a task closer to classification, where the columns of the outputs $Y_N \in \mathbb{R}^{n_L \times N}$ are one-hot vectors, then $(Y_N^T Y_N)^{\frac{1}{2}}$ is (up to permutations of the columns/lines) a block diagonal matrix with n_L constant positive blocks, which is completely positive with CP-rank equal to the number of classes n_L. This is in line with our empirical experiments in Figure 5.2 where we observe in MNIST a plateau starting roughly at a width of 10, which is the number of classes.

Another example where the structure of the data leads to an earlier plateau is when the input and output dimensions are both 1, in which case we can guarantee that the start of the plateau grows at most linearly with the number of datapoints N:

Proposition 5.4.10. *Consider shallow networks (*$L = 2$*) with scalar inputs and outputs (*$n_0 = n_2 = 1$*), a ReLU nonlinearity, and a dataset* $X, Y \in \mathbb{R}^{1\times N}$*. Both the representation cost* $R_{\mathbf{n}}(X, Y)$ *and global minimum* $\min_{\mathbf{W}} \mathcal{L}_{\lambda,\mathbf{n}}(\mathbf{W})$ *for any* $\lambda > 0$ *are independent of the width* n_1 *as long as* $n_1 \geq 4N$*.*

More generally, we propose to view the start of the plateau as an indicator of how well a certain task is adapted to a DNN architecture. An early plateau suggests that the network is able to solve the task optimally with very few neurons, in contrast to a late plateau. The fact that the optimal network requires few neurons (and hence few parameters) can be used to guarantee good generalization.

5.4.3 Conclusion

We have given two reformulations of the loss of L_2-regularized DNNs. The first works for a general non-linearity and shows how the hidden representations of the inputs $Z_1, \ldots, Z_{L-1}$ are learned to interpolate between the input and output representations, as a balance between attraction and repulsion forces for every layer. The second reformulation for homogeneous non-linearities allows us to analyze a sparsity effect of L_2-regularized DNNs, where the learned networks are equivalent to another network with much fewer neurons. This effect can be visualized by the appearance of a plateau in the minimal loss as the number of neurons grows, the earlier the plateau, the sparser the solution, since an early plateau means that very few neurons were required to obtain the same loss as a network with an infinite number of neurons. We show that this plateau cannot start later than $N(N+1)$, and then show that the order of this bound is tight by constructing a toy dataset for which the plateau starts at $N^2/4$, however, we observe that on more traditional datasets, the start of the plateau can be much earlier.

6 A Generalization bound for nearly-linear networks

Authors: Eugene Golikov.

Link and reference: Golikov (2025), published at *Transaction on Machine Learning Research (TMLR)*[1].

Contribution: this work was done solely by the author of the present thesis.

6.1 Introduction

Despite huge practical advancements of deep learning, the main object of this field, a neural network, is not yet fully understood. As we do not have a complete understanding of how neural networks learn, we are not able to answer the main question of deep learning theory: *why do neural networks generalize on unseen data?*

While the above question is valid for any supervised learning model, it is notoriously difficult to answer for neural nets. The reason is that modern neural nets have billions of parameters and as a result, huge capacity. Therefore among all parameter configurations that fit the training data, there provably exist such configurations that do not fit the held-out data well (Zhang et al., 2017). This is the reason why classical approaches for bounding a generalization gap, i.e. the difference between distribution and train errors, fall short on neural networks: such approaches bound the gap uniformly over a model class. That is, if weights for which the network performs poorly exist, we bound our trained network's performance by performance of that poor one.

As we observe empirically, networks commonly used in practice do generalize, which means that training algorithms we use (i.e. gradient descent or its variants) choose "good" parameter configurations despite the existence of poor ones. In other words, these algorithms are *implicitly biased* towards good solutions.

Unfortunately, implicit bias of gradient descent is not fully understood yet. This is because the training dynamics is very hard to integrate, or even characterize, analytically. There are two main obstacles we could identify. First, modern neural networks have many layers, resulting in a high-order weight evolution equation. Second, activation functions we use are applied to hidden representations elementwise, destroying a nice algebraic structure of stacked linear transformations.

[1] https://openreview.net/forum?id=tRpWaK3pWh

If we remove all activation functions, the training dynamics of gradient descent can be integrated analytically under certain assumptions (Saxe et al., 2013). However, the resulting model, a linear network, is as expressive as a linear model, thus loosing one of the crucial advantages of neural nets.

Idea. The idea we explore in the present paper is to consider nonlinear nets as perturbations of linear ones. We show that the original network can be approximated with a proxy-model whose parameters can be computed using parameters of a linear network trained the same way as the original one. Since the proxy-model uses the corresponding linear net's parameters, its generalization gap can be meaningfully bounded with classical approaches. Indeed, if the initial weights are fixed, the result of learning a linear network to minimize square loss on a dataset $(X \in \mathbb{R}^{d\times m}, Y \in \mathbb{R}^{d_{out}\times m})$ is determined uniquely by $YX^\top \in \mathbb{R}^{d_{out}\times d}$ and $XX^\top \in \mathbb{R}^{d\times d}$, where d, d_{out} are the input and output dimensions, and m is the dataset size. The number of parameters in these two matrices is much less than the total number of parameters in the network, making classical counting-based approaches meaningful.

Contributions. Our main contribution is a generalization bound given by Theorem 6.5.2, which is ready to apply in the following setting: (1) fully-connected networks, (2) gradient descent with vanishing learning rate (gradient flow), (3) binary classification with MSE loss. The main disadvantage of our bound is that it diverges as training time t goes to infinity. We discuss how to choose the training time in such a way that the bound stays not too large while the training risk diminishes significantly, in Section 6.5.3. We validate our bound on a simple fully-connected network trained on a downsampled MNIST dataset, and demonstrate that it becomes non-vacuous in this scenario (Section 6.6). We list advantages and disadvantages of our bound over that of existing generalization bounds in Section 6.2. We discuss the assumptions we use, as well as possible improvements of our bound, in Appendix D.1. Finally, we discuss how far the approach we choose in this work, i.e. generalization bounds based on deviation from specific proxy models, could lead us in the best case scenario (Appendix D.2).

6.2 Comparison to previous work

Advantages. The bound of our Theorem 6.5.2 has the following advantages over some other non-vacuous bounds available in the literature (e.g. Biggs and Guedj (2022); Galanti et al. (2023); Dziugaite and Roy (2017); Zhou et al. (2019)):

1. It is an *a priori* bound, i.e. getting the actual trained the model is not required for evaluating it. To the best of our knowledge, it is the first non-vacuous *a priori* bound available for neural nets. All works mentioned above, despite providing non-vacuous bounds, could be evaluated only on a trained network thus relying on the implicit bias phenomenon which is not well understood yet.

2. Related to the previous point, to the best of our knowledge, our bound is the first to incorporate the implicit bias of gradient flow explicitly. This is done by demonstrating that the model does not deviate much from a specific proxy model. The class of realizable proxy models has a small number of effective parameters. Informally, this indicates that this class is "simple"; hence the gradient flow is biased towards this simple class, and our bound exploits this property.

3. It does not require a held-out dataset to evaluate it, compared to Galanti et al. (2023) and

coupled bounds of Biggs and Guedj (2022). Indeed, if one had a held-out dataset, they could just use it directly to evaluate the trained model, thus questioning the practical utility of such bounds.

4. It does not grow with network width. In contrast, PAC-Bayesian bounds, Biggs and Guedj (2022); Dziugaite and Roy (2017); Zhou et al. (2019), might grow with width.

5. Similarly to Biggs and Guedj (2022); Galanti et al. (2023), we bound the generalization gap of the original trained model, not its proxy. In contrast, Dziugaite and Roy (2017) introduces a Gaussian noise to the learned weights, while Zhou et al. (2019) crucially relies on quantization and compression after training.

Related to the second point, we bound the generalization gap for the proxy model with a simple parameter-counting bound (similar to that of Vapnik and Chervonenkis (1971)), thus demonstrating that, contrary to a common judgement, such bounds could be useful in the context of models with many more parameters than data points.

Disadvantages. For a fair comparison, we also list the disadvantages our present bound has:

1. The bound of our Theorem 6.5.2 becomes non-vacuous only when the following two conditions hold. First, a simple counting-based generalization bound for a linear model evaluated in the same setting should be non-vacuous. Such a bound is vacuous even for binary classification on the standard MNIST dataset, but becomes non-vacuous if we downsample the images.

2. Second, the activation function has to be sufficiently close to being linear. To be specific, for a two-layered leaky ReLU neural net trained on MNIST downsampled to 7x7, one needs the negative slope to be not less than 0.99 (1 corresponds to a linear net, while ReLU corresponds to 0).

3. One may hope for the bound to be non-vacuous only for a partially-trained network, while for a fully-trained network the bound diverges.

4. Even in the most optimistic scenario, when we manage to tighten the terms of our bound as much as possible, our bound stays non-vacuous only during the early stage of training when the network has not started "exploiting" its nonlinearity yet (Appendix D.2). However, the minimal negative ReLU slope for which the bound stays non-vacuous is much smaller, 0.6.

6.3 Related work

Non-vacuous generalization bounds. While first generalization bounds date back to Vapnik and Chervonenkis (1971), the bounds which are non-vacuous for realistically large neural nets appeared quite recently. Specifically, Dziugaite and Roy (2017) constructed a PAC-Bayesian bound which was non-vacuous for small fully-connected networks trained on MNIST and FashionMNIST. The bound of Zhou et al. (2019), also of PAC-Bayesian nature, relies on quantization and compression after training. It ends up being non-vacuous for VGG-like nets trained on large datasets of ImageNet scale. The bound of Biggs and Guedj (2022) is the first non-vacuous bound that applies directly to the learned model and not to its proxy. It is not obvious whether their construction could be generalized to neural nets with more than two layers. The bound of Galanti et al. (2023) is not

PAC-Bayesian in contrast to the previous three. It is based on the notion of effective depth: it assumes that a properly trained network has small effective depth, even if it is deep. Therefore its effective capacity is smaller than its total capacity, which allows for a non-vacuous bound. See the previous section for discussion of some of the features of these bounds.

A priori and a posteriori generalization bounds. Most of the generalization bounds available in the literature are *a posteriori*. That is, some PAC-Bayesian bounds (Dziugaite and Roy, 2017; Neyshabur et al., 2018; Biggs and Guedj, 2022), as well as Rademacher complexity-based bounds (Bartlett et al., 2017), depend on norms of learned weights, or distances between the learned weights and their initializations, so they cannot be computed before these norms or distances are known. The bound of Zhou et al. (2019) depends on the trained model size after compression and quantization, which cannot be predicted before training and compression. That is, this bound implicitly relies on the fact that the training procedure prefers well-compressible models; to the best of our knowledge, this fact does not have a solid explanation yet. The bound of Galanti et al. (2023) depends on a so-called *effective depth* of a learned model, which also cannot be predicted in advance.

The only *a priori* bounds we are aware of are the classical uniform bounds based on VC-dimension and Rademacher complexity, see Vapnik and Chervonenkis (1971). These bounds grow with model expressivity, hence with both width and depth, which make them vacuous in most of the practical scenarios involving neural nets. The reason they are vacuous is also related to the fact that among all models that interpolate the data, one can often find a model that does not generalize well, see empirical results of Zhang et al. (2017). Uniform bounds bound the generalization gap in the worst case, hence if a "bad" model exists in our model class, the whole generalization bound has no chance to be good.

Overall, generalization bounds that use some complexity of the whole function class fall short due to the trade-off they impose: low complexity = provably good generalization, high complexity = maybe poor generalization. Modern neural architectures are deep and wide, hence they have high complexity, while they still generalize well. The reason is that the training procedure we often use is likely to be implicitly biased towards well-generalizing solutions. If we bound the generalization gap uniformly over the whole function class, we ignore this effect. The *a posteriori* bounds we discuss above do take some form of implicit bias into account (i.e. low weight norm, compressibility etc.). So does our bound: we exploit the fact that a nearly linear network is close to a proxy which uses only weights of a linear network. In their turn, the trained weights of a linear network depend only on a few parameters as long as their initialization is given. The difference is that our bound does not need any prior info about the learned weights (only on their initialization and the optimization procedure); hence it is *a priori*.

Linear networks training dynamics. The pioneering work that integrates the training dynamics of a linear network under gradient flow to optimize square loss is Saxe et al. (2013). This work crucially assumes that the initial weights are well aligned with the eigenvectors of the optimal linear regression weights YX^+, where (X, Y) is the train dataset. As the initialization norm approaches zero, the learning process becomes more sequential: components of the data are learned one by one starting from the strongest. Li et al. (2021) conjecture that the same happens for any initialization approaching zero excluding some set of directions of measure zero. They prove this result for the first, the strongest, component, but moving further seems more challenging. See also Yun et al. (2021); Jacot et al. (2021a). We note a recent result of Tu et al. (2024) who propose a so-called *mixed* dynamics for two-layered linear networks that does not suffer from this issue.

Nearly-linear neural nets. Our generalization gap bound decreases as activation functions get closer to linearity. While nearly-linear activations do not conform with the usual practice, nearly-linear networks have been studied before. That is, Li et al. (2022) demonstrated that when width n and depth L go to infinity with constant ratio, the hidden layer covariances at initialization admit a meaningful limit as long as the ReLU slopes behave as $1 \pm \frac{c}{\sqrt{n}}$, i.e. become closer and closer to linearity. Noci et al. (2024) explored a similar limit for Transformers (Vaswani et al., 2017). Another example is Kumar et al. (2024), who used a toy small-ϵ model to explain the phenomenon of *grokking* (Power et al., 2022) as delayed feature learning.

Linearized training of neural networks and NTK. It is important to emphasize that linear networks we consider in the present work are fundamentally different from the NTK model introduced by Jacot et al. (2018) resulted from linearized training of a neural network. That is, by a linear network we mean a network $f_\theta(x)$ who is linear in its input, x, but not necessarily in its weights θ. Whereas if we linearize the training procedure, the trained model becomes linear in θ, but not in x. Jacot et al. (2018) demonstrated that training a neural network becomes equivalent to training a kernel method under specific (non-standard) parameterization as width goes to infinity (NTK limit). Since training a kernel method is equivalent to training a random feature model with sufficiently large number of features, the corresponding training procedure is indeed linear in weights. See also Yang and Littwin (2021) for proving NTK limit for general architectures, and Chizat et al. (2019) for demonstrating linearized training for large weight initializations.

As in the NTK limit training a neural net becomes equivalent to training a kernel method, features the model uses stay the same over the whole process of training. In other words, the model does not enjoy *feature learning* when training is linearized. In contrast, even a linear network with two layers trained with gradient descent does demonstrate feature learning: see Saxe et al. (2013); Tu et al. (2024). The feature learning it exhibits is kernel alignment and a so-called *momentum effect*: the associated kernel (NTK), or equivalently, the features the model uses, aligns over strong components of the data and extends along them Tu et al. (2024). This way, these strong components get learned first, before weak components, often resulted from data noise.

We also emphasize that the proxy models we use to prove our results, while exploiting weights of a trained linear network, are not linear themselves; neither in x, nor in weights. Therefore they do not suffer from expressivity issues as linear networks (which are all functionally equivalent to linear models $x \to Wx$), while they do enjoy feature learning (since linear networks do).

Therefore our approach is quite orthogonal to NTK and generalizability of kernel methods. In terms of high-level methodology, the closest paper we could mention is Arora et al. (2019a), where they consider a two-layer NTK-parameterized MLP, and prove a generalization bound for it when width is sufficiently large. For this, they combine a generalization bound for the infinite-width limit when the kernel is constant, and a term that takes into account the fact that the kernel for finite width deviates from the limit one. Whereas what we do is we combine a generalization bound for a proxy model that exploits the weights of a linear network and a proxy deviation bound. The bound of Arora et al. (2019a) becomes good when the width is large (hence the finite-width NTK is close to the limit NTK), while our bound becomes good when the activation functions are close to be linear (hence the proxy does not deviate from the original model much).

6.4 Our approach

Before formally stating our main bound, we present the high-level approach for constructing generalization bounds we take in our work.

Generalization bound based on proxy models. Let f be a model trained on a binary classification task. We will look for a proxy model g that satisfies the following properties: (1) it is close enough to f, and (2) the generalization gap for g could be bounded well enough. Given such g, we bound the distribution risk of f as follows:

$$R[f] \leq \hat{R}_\gamma[f] + \left(R^C_\gamma[g] - \hat{R}^C_\gamma[g]\right) + \frac{1}{\gamma}\mathbb{E}_{x\sim\mathcal{D}}|f(x) - g(x)| + \frac{1}{\gamma}\frac{1}{m}\sum_{k=1}^{m}|f(x_k) - g(x_k)|, \tag{6.1}$$

where $\gamma > 0$ and we used three different risk functions:

1. Misclassification risk: $r(y, \hat{y}) = [y\hat{y} < 0]$,
2. Margin risk: $r_\gamma(y, \hat{y}) = [y\hat{y} < \gamma]$,
3. Continuous margin risk: $r^C_\gamma(y, \hat{y}) = [y\hat{y} < 0] + \left(1 - \frac{y\hat{y}}{\gamma}\right)[y\hat{y} \in [0, \gamma]]$,

giving rise to the distribution risk $R[f] = \mathbb{E}_{x,y\sim\mathcal{D}} r(y, f(x))$, and similarly $R_\gamma[f]$ and $R^C_\gamma[f]$, and its empirical counterpart $\hat{R}[f] = \frac{1}{m}\sum_{k=1}^{m} r(y_k, f(x_k))$, and similarly $\hat{R}_\gamma[f]$ and $\hat{R}^C_\gamma[f]$. Here $\mathcal{D}$ is the data distribution, and $(x_k, y_k)_{k=1}^m$ is the training dataset samled from $\mathcal{D}$ in an iid manner.

We derived Eq. (6.1) simply from the fact that $r^C_\gamma(\cdot, y)$ is $1/\gamma$-Lipschitz, and $r \leq r^C_\gamma \leq r_\gamma$, see Eq. (6.22) below for a more elaborate derivation. The first term on the right hand side of Eq. (6.1) is the empirical margin risk of the original model f. The second term is the generalization gap of the proxy-model g (with respect to continuous margin risk) which we assume to be easy to bound. The two remaining terms are deviations of g from f averaged over the whole data distribution $\mathcal{D}$ and the dataset, divided by γ. The parameter γ controls the trade-off between the first term and the other three: the first term increases with γ (eventually reaching 1), while the other three decrease.

Proxy models for a nearly-linear network. The problem now boils down to finding a good proxy g that approximates the trained model f well, and whose generalization gap could be well-bounded. Let $f^\epsilon_\theta(x)$ be a neural network with parameters θ, input $x \in \mathbb{R}^d$, scalar output, and activation functions $\phi^\epsilon(z) = z + \epsilon\psi(z)$, ψ is a positively homogeneous map (e.g. ReLU). That is, $f^0_\theta(x)$ is linear in x (but not necessarily in θ), and ϵ characterizes the deviation from linearity.

Let $f^\epsilon_{\theta^\epsilon}$ be the trained model whose generalization gap we would like to bound, and let $f^0_{\theta^0}$ be a linear model trained the same way. We consider the following proxies: $g_1(x) = f^\epsilon_{\theta^0}(x)$ and $g_2(x) = g_1(x) + \left(f^\epsilon_{\theta^\epsilon}(X) - g_1(X)\right)X^+x$, where $X \in \mathbb{R}^{d\times m}$ is a matrix with rows $(x_k)_{k=1}^m$, and X^+ is its Moore-Penrose pseudo-inverse.

In words, the first proxy is a nonlinear network that uses weights of the trained linear one, while the second proxy results from linearly correcting the first one on the train dataset. These proxies deviate from the original model as follows: $|g_\kappa(x) - f^\epsilon_{\theta^\epsilon}(x)| = O(\epsilon^\kappa)$ for $\kappa \in \{1, 2\}$ and any given $x \in \mathbb{R}^d$.

Under certain assumptions, θ^0, the trained linear model weights, depend only on $YX^+ \in \mathbb{R}^d$, where $Y \in \mathbb{R}^{1\times d}$ is a row-vector of targets. Since g_1 is fully described by θ^0, the class of models

realizable by it has only d parameters. Since g_2 is merely g_1 plus a linear correction, its respective class of models has only $d + d = 2d$ parameters. In both cases, the number of parameters the proxy has is much smaller than the total number of weights of the network, which is $(d+1)n + (L-2)n^2$. Then classical uniform bound techniques (Vapnik and Chervonenkis, 1971) result in a generalization bound for proxy models that grows as $\sqrt{\frac{\kappa d}{m}}$. This bound depends neither on width of the network, nor on its depth.

6.5 Main result

Notations. For integer $L \geq 0$, $[L]$ denotes the set $\{1, \ldots, L\}$. For integers $l \leq L$, $[l, L]$ denotes the set $\{l, \ldots, L\}$. For a vector x, $\|x\|$ denotes its Euclidean norm, while $\|x\|_1$ denotes its l_1-norm. For a matrix X, $\|X\|$ denotes its maximal singular value, while $\|X\|_F$ denotes its Frobenius norm.

6.5.1 Setup

Model. The model we study is a fully-connected LeakyReLU network with L layers:

$$f_\theta^\epsilon(x) = W_L x_{\theta,L-1}^\epsilon(x), \qquad x_{\theta,0}^\epsilon(x) = x, \qquad x_{\theta,l}^\epsilon(x) = \phi^\epsilon\left(W_l x_{\theta,l-1}^\epsilon(x)\right) \quad \forall l \in [L-1], \tag{6.2}$$

where $\theta \in \mathbb{R}^N$ denotes the vector of all weights, i.e. $\theta = \mathrm{cat}(\{\mathrm{vec}(W_l)\}_{l=1}^L)$, $W_l \in \mathbb{R}^{n_l \times n_{l-1}}$ $\forall l \in [L]$, and ϕ^ϵ is a Leaky ReLU with a negative slope $1 - \epsilon$, that is:

$$\phi^\epsilon(x) = x - \epsilon \min(0, x). \tag{6.3}$$

Since we are going to consider only binary classification in the present work, we take $n_L = 1$. We also define $d = n_0$ to denote the input dimension.

Data. Data points (x, y) come from a distribution $\mathcal{D}$. We assume all x from $\mathcal{D}$ to lie on a unit ball, $\|x\| \leq 1$, and $y \in \{-1, 1\}$. During the training phase, we sample a set of m points iid from $\mathcal{D}$ to form a dataset (X, Y), where $X \in \mathbb{R}^{d \times m}$ and $Y \in \mathbb{R}^{1 \times m}$.

Training. Assuming $\mathrm{rk}\, X = d$ (which implies $m \geq d$, i.e. the data is abundant), we train our model on (X, Y) with gradient flow to optimize square loss on whitened data, i.e. on $(\tilde{X}, Y)$ for $\tilde{X} = \Sigma_X^{-1/2} X$, where $\Sigma_X = \frac{1}{m} X X^\top \in \mathbb{R}^{d \times d}$ is an empirical feature correlation matrix. That is,

$$\frac{dW_l^\epsilon(t)}{dt} = -\frac{\partial \left(\frac{1}{2m} \left\| Y - f_{\theta^\epsilon(t)}^\epsilon(\tilde{X}) \right\|_F^2 \right)}{\partial W_l} \quad \forall l \in [L]. \tag{6.4}$$

Note that $\tilde{X}\tilde{X}^\top = m I_d$.

Inference. To conform with the above training procedure, we take the model output at a point x to be $f_\theta^\epsilon\left(\Sigma^{-1/2} x\right)$, where Σ is a (distribution) feature correlation matrix: $\Sigma = \mathbb{E}_{(x,y)\sim\mathcal{D}}(x x^\top) \in \mathbb{R}^{d \times d}$. We assume this matrix to be known; in practice, we could substitute it with Σ_X.

Performance measure. Recall the definitions of r, r_γ, and r_γ^C from Section 6.4. Slightly abusing the notation, we define the empirical (train) risk on the dataset (X, Y) and the distribution risk of the model trained for time t as

$$\hat{R}^\epsilon(t) = \frac{1}{m}\sum_{k=1}^{m} r\left(y_k, f^\epsilon_{\theta^\epsilon(t)}(\Sigma^{-1/2}x_k)\right), \qquad R^\epsilon(t) = \mathbb{E}_{(x,y)\sim\mathcal{D}} r\left(y, f^\epsilon_{\theta^\epsilon(t)}(\Sigma^{-1/2}x)\right). \tag{6.5}$$

We define $R^\epsilon_\gamma(t)$ and $R^{C,\epsilon}_\gamma(t)$ analogously to $R^\epsilon(t)$, and $\hat{R}^\epsilon_\gamma(t)$ and $\hat{R}^{C,\epsilon}_\gamma(t)$ analogously to $\hat{R}^\epsilon(t)$.

6.5.2 Generalization bound

We will need the following assumption on the training process:

Assumption 6.5.1. $\forall t \geq 0 \ \left\|\left(Y - f^\epsilon_{\theta^\epsilon(t)}(\tilde{X})\right)\tilde{X}^\top\right\|_F^2 \leq \left\|\left(Y - f^\epsilon_{\theta^\epsilon(0)}(\tilde{X})\right)\tilde{X}^\top\right\|_F^2$.

Note that $\left\|Y - f^\epsilon_{\theta^\epsilon(t)}(\tilde{X})\right\|_F^2 \leq \left\|Y - f^\epsilon_{\theta^\epsilon(0)}(\tilde{X})\right\|_F^2$ since we minimize the loss monotonically with gradient flow. This implies that the above assumption holds automatically, whenever $\tilde{X}$ is a (scaled) orthogonal matrix (which happens when $m = d$). It is easy to show that it provably holds for a linear network ($\epsilon = 0$), see Appendix D.5, and we found this assumption to hold empirically for all of our experiments with nonlinear networks too, see Fig. 6.7.

We are now ready to formulate our main result:

Theorem 6.5.2. *Fix $\beta, \gamma > 0$, $t \geq 0$, $\delta \in (0,1)$, $\epsilon \in [0,1]$, and $\kappa \in \{1,2\}$. Let p be the floating point arithmetic precision (32 by default). Under the setting of Section 6.5.1 and Assumption 6.5.1, for any weight initialization satisfying $\|W_l^\epsilon(0)\| \leq \beta \ \forall l \in [L]$, w.p. $\geq 1-\delta$ over sampling the dataset (X, Y),*

$$R^\epsilon(t) \leq \hat{R}^\epsilon_\gamma(t) + \Upsilon_\kappa + \frac{\Delta_{\kappa,\beta}(t)\epsilon^\kappa}{\gamma}, \tag{6.6}$$

for the terms in the rhs defined as

$$\Upsilon_\kappa = \sqrt{\frac{\kappa p d \ln 2 + \ln(1/\delta)}{2m}}, \qquad \Delta_{\kappa,\beta}(t) = \Phi_\kappa v_\beta(t) u_\beta^{L-1}(t), \tag{6.7}$$

where

$$\Phi_\kappa = \begin{cases} L\sqrt{d} + 1 + (L-1)\rho, & \kappa = 1; \\ (L-1)\left[(L+1+(L-1)\rho)\sqrt{d} + 2(1+(L-1)\rho)\right], & \kappa = 2, \end{cases} \tag{6.8}$$

where $\rho = \frac{\|W_1^\epsilon(0)\|_F}{\|W_1^\epsilon(0)\|}$ is the square root of the stable rank of the input layer at initialization, and the definitions of u_β and v_β are given below.

Define

- $s = \left\|YX^\top\Sigma_X^{-1/2}\right\|$; $\quad \bar{1} = 1 + \rho\beta^L$;
- *For a given $r \geq 0$, $\bar{s}_r = (1-\epsilon)(s + \rho\beta^L) + \epsilon\sqrt{r}(L-1)(1+\rho\beta^L)$;*
- $\bar{s} = \bar{s}_1$; $\quad \hat{s} = \frac{L}{1+(L-1)\rho}\bar{s}$; $\quad \bar{\beta} = \beta\frac{\bar{s}_\rho}{\bar{s}_1}$.

We have

$$u_\beta(t) = \begin{cases} \bar{\beta} e^{\bar{s}t}, & L = 2; \\ \left(\bar{\beta}^{2-L} - (L-2)\bar{s}t\right)^{\frac{1}{2-L}}, & L \geq 3; \end{cases} \tag{6.9}$$

$$v_\beta(t) = \frac{L-1}{L} \hat{s}^{\frac{2-L}{L}} u_\beta^{L-1}(t)[w(u_\beta(t)) - w(\beta)] e^{\frac{u_\beta^L(t)}{\hat{s}}}, \tag{6.10}$$

where [2]

$$w(u) = -\frac{\bar{1}}{\bar{s}} \Gamma\left(\frac{2-L}{L}, \frac{u^L}{\hat{s}}\right) - \rho \frac{\hat{s}}{\bar{s}} \Gamma\left(\frac{2}{L}, \frac{u^L}{\hat{s}}\right). \tag{6.11}$$

This theorem gives an *a priori* bound for the generalization gap $R^\epsilon - \hat{R}^\epsilon_\gamma$, i.e. it could be computed without performing the actual training.

Dimension dependency. Our bound does not depend on width n_l $\forall l \in [L-1]$, in contrast to the bounds of Dziugaite and Roy (2017); Zhou et al. (2019); Biggs and Guedj (2022). However, both penalty terms of Theorem 6.5.2, Υ_κ and $\Delta_{\beta,\kappa}(t)$, grow as $\sqrt{d}$ with the input dimension.

6.5.3 Choosing the training time

As we see, $\Delta_{\kappa,\beta}(t)$ diverges super-exponentially as $t \to \infty$ for $L = 2$ and as $t \to \frac{\bar{\beta}^{2-L}}{(L-2)\bar{s}}$ for $L \geq 3$, so the bound eventually becomes vacuous. For the bound to make sense, we should be able to find t small enough for $\Delta_{\kappa,\beta}(t)$ to stay not too large, and at the same time, large enough for the training risk $\hat{R}^\epsilon_\gamma(t)$ to get considerably reduced.

In the present section, we are going to demonstrate that for values of t which correspond to partially learning the dataset (i.e. for which $\hat{R}^\epsilon_\gamma(t) \in (0,1)$), the last term of the bound, $\Delta_{\kappa,\beta}(t)/\gamma$, admits a finite limit as $\beta \to 0$.[3]

We do not know how $\hat{R}^\epsilon_\gamma(t)$ decreases with t in our case. However, when the network is linear, this can be computed explicitly when either the weight initialization is properly aligned with the data, or the initialization norm β vanishes. We are going to perform our whole analysis for $L = 2$ in the main, and defer the case $L \geq 3$ to Appendix D.4.2.

That is, consider an SVD: $Y\tilde{X}^+ = \frac{1}{\sqrt{m}} Y X^\top (X X^\top)^{-1/2} = P S Q^\top$, where P and Q are orthogonal and S is diagonal; note that $S_{11} = s$. Observe also that $s = \|Y\tilde{X}^+\| \leq \|Y\| \|\tilde{X}^+\| \leq 1$ since $y = \pm 1$. Saxe et al. (2013)[4] have demonstrated for linear nets ($\epsilon = 0$) and $L = 2$ that when the weight initialization is properly aligned with P and Q[5], $\|W_1^0(t)\| = \|W_2^0(t)\| = \bar{u}(t)$, where $\bar{u}$ satisfies[6]

$$\frac{d\bar{u}(t)}{dt} = \bar{u}(t)(s - \bar{u}^2(t)), \qquad \bar{u}(0) = \beta, \tag{6.12}$$

which gives the solution in implicit form:

$$t = \int \frac{d\bar{u}}{\bar{u}(s - \bar{u}^2)} = \frac{1}{2s} \ln\left(\frac{\bar{u}^2(t)(s - \beta^2)}{\beta^2(s - \bar{u}^2(t))}\right). \tag{6.13}$$

[2] Γ is an upper-incomplete gamma-function defined as $\Gamma(s,x) = \int_x^\infty t^{s-1} e^{-t}\, dt$.

[3] In the linear case ($\epsilon = 0$), this corresponds to the saddle-to-saddle regime of Jacot et al. (2021a).

[4] Saxe et al. (2013) assumed $X X^\top = I_d$ and $\|Y X^\top\| = s$, while not introducing the factor $\frac{1}{m}$ as we do in Eq. (6.4). It is easy to see that the our gradient flow has exactly the same dynamics as the one studied by Saxe et al. (2013).

[5] That is, when $W_2(0) = P \bar{S}_2^{1/2} R^\top$, $W_1(0) = R \bar{S}_1^{1/2} Q^\top$, where R is orthogonal, and $\bar{S}_1$ and $\bar{S}_2$ are constructed from S by adding or removing zero rows and columns.

[6] Our $\bar{u}$ corresponds to $u^{1/(N_l - 1)}$ of Saxe et al. (2013).

One could resolve $\bar{u}$ explicitly to get

$$\bar{u}(t) = \frac{se^{2st}}{e^{2st} - 1 + s/\beta^2}. \tag{6.14}$$

Saxe et al. (2013) observed this expression to hold with good precision for random (not aligned) initializations when β is small enough. This is supported by further works (Li et al., 2021; Jacot et al., 2021a): when a linear network is initialized close to the origin and the initial weights do not lie on a "bad" subspace, the gradient flow nearly follows the same trajectory as studied by Saxe et al. (2013).

Plugging $\bar{u}^2(t) = \alpha s$ into Eq. (6.13), we get the time required for a linear network to learn a fraction α of the strongest mode:

$$t^*_\alpha(\beta) = \frac{1}{2s} \ln\left(\frac{\alpha(s-\beta^2)}{(1-\alpha)\beta^2}\right). \tag{6.15}$$

Clearly, the learning time $t^*_\alpha(\beta)$ diverges whenever $\beta \to 0$, or $\alpha \to 1$.

Since $t^*_\alpha(\beta)$ is the time sufficient to learn a network for $\epsilon = 0$, we suppose it also suffices to learn a nonlinear network. Thus, we are going to evaluate our bound at $t = t^*_\alpha$. Since we need $\hat{R}_\gamma(t^*_\alpha) < 1$ for the bound to be non-vacuous, we should take γ small relative to α. We consider $\gamma = \alpha^\nu/q$ for $\nu, q \geq 1$.

Since the linear network learning time $t^*_\alpha(\beta)$ is correct for almost all initialization only when β vanishes, we are going to work in the limit of $\beta \to 0$. Since we need $\alpha \in (\beta^2/s, 1)$, otherwise the linear training time is negative, we take $\alpha = \frac{r}{s}\beta^\lambda$ for $\lambda \in (0, 2]$ and $r > 1$.

Let us compute the quantities which appear in our bounds, at $t = t^*_\alpha(\beta)$:

$$u = \bar{\beta}\left(\frac{\alpha(s-\beta^2)}{(1-\alpha)\beta^2}\right)^{\frac{\bar{s}}{2s}} = \frac{\bar{s}_\rho}{\bar{s}_1} r^{\frac{\bar{s}}{2s}} \beta^{1+\frac{\bar{s}}{s}\left(\frac{\lambda}{2}-1\right)}(1 + O(\beta)), \tag{6.16}$$

where we omitted the argument $t^*_\alpha(\beta)$ for brevity.

Since $\Gamma(0, x) = -\operatorname{Ei}(-x) = -\ln x + O_{x\to 0}(1)$ and $\Gamma(1, x) = e^{-x} = O_{x\to 0}(1)$, we have $w(\beta) = \frac{\bar{1}}{\bar{s}} \ln(\beta^2) + O(1)$. Similarly, whenever $1 + \frac{\bar{s}}{s}\left(\frac{\lambda}{2} - 1\right) > 0$, we have $u = o(\beta)$, and $w(u) = \frac{\bar{1}}{\bar{s}} \ln\left(\beta^{2+\frac{\bar{s}}{s}(\lambda-2)}\right) + O(1)$.

Consider first $\lambda < 2$ and suppose $\nu = 1$. In this case, $w(u) - w(\beta) = \frac{\bar{1}}{s}\left(\frac{\lambda}{2} - 1\right)\ln(\beta^2) + O(1)$, and

$$\frac{2uv}{\gamma} = \frac{\bar{s}_\rho^2}{\bar{s}_1^2} \frac{q r^{\frac{\bar{s}}{s}-1}\bar{1}}{\beta^{\left(\frac{\bar{s}}{s}-1\right)(2-\lambda)}} \left(\frac{\lambda}{2} - 1\right) \ln(\beta^2)(1 + O(1/\ln\beta)). \tag{6.17}$$

Since $\bar{s} \geq s$, this expression diverges as $\beta \to 0$. Note that we get the same order divergence even also for $1 + \frac{\bar{s}}{s}\left(\frac{\lambda}{2} - 1\right) \leq 0$. Clearly, for $\nu > 1$, we get even faster divergence.

On the other hand, if $\lambda = 2$ then $w(u) - w(\beta) = \frac{\bar{1}}{\bar{s}} \ln\left(\frac{\bar{s}_\rho^2}{\bar{s}_1^2} r^{\frac{\bar{s}}{s}}\right) + O(\beta)$, and

$$\frac{2uv}{\gamma} = \frac{s^\nu \bar{s}_\rho^2}{\bar{s}_1^3} q r^{\frac{\bar{s}}{s}-\nu}\bar{1} \ln\left(\frac{\bar{s}_\rho^2}{\bar{s}_1^2} r^{\frac{\bar{s}}{s}}\right) \beta^{2(1-\nu)}(1 + O(\beta)). \tag{6.18}$$

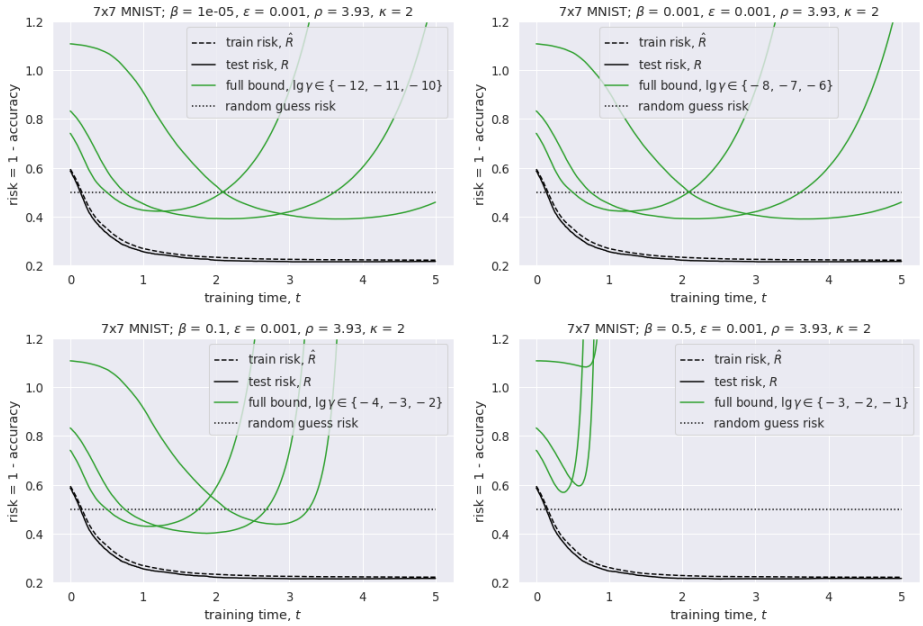


Figure 6.1: We consider 7x7 binary MNIST, $L = 2$, $\kappa = 2$, $\epsilon = 0.001$, and vary β. The bound of Theorem 6.5.2 converges as β vanishes and increases as β grows. The bound stays non-vacuous for a small enough β and a properly choosen γ. We consider $\gamma = \beta^2/q$ for $q \in \{1, 10, 100\}$.

We get a finite limit only for $\nu = 1$, i.e. when $\gamma \propto \alpha$. In this case, the last term of the bound Eq. (6.6) becomes

$$\frac{\Delta_{\kappa,\beta}\epsilon^\kappa}{\gamma} = \frac{\bar{1}s\bar{s}_\rho^2}{2\bar{s}_1^3} q r^{\frac{\bar{s}}{s}-1} \Lambda_\kappa \ln\left(\frac{\bar{s}_\rho^2}{\bar{s}_1^2} r^{\frac{\bar{s}}{s}}\right) \epsilon^\kappa (1 + O(\beta)). \tag{6.19}$$

Summing up, we expect the empirical risk $\hat{R}_\gamma(t^*_\alpha)$ to be smaller than $\frac{1}{2}$ for large enough q (i.e. for small γ compared to α), and the last term to stay finite as $\beta \to 0$ and vanish as ϵ^κ. Therefore as long as Υ_κ is not large enough (i.e. when κpd is small compared to m), the overall bound becomes non-vacuous for small enough ϵ at least at the (linear) training time t^*_α. We are going to evaluate our bound empirically in the upcoming section.

6.6 Experiments

Setup. We consider an L-layer bias-free fully-connected network of width 64 and train it to classify 0-4 versus 5-9 digits of MNIST (i.e. $m = 60000$). In order to approximate the gradient flow dynamics, we run gradient descent with learning rate 0.001. By default, we take $L = 2$, the floating point precision to be $p = 32$, downsample the images to 7x7, and initialize the layers randomly in a standard Pytorch way (plus, we rescale the weights to match the required layer norm β). For some experiments, we consider deeper networks, half-precision $p = 16$, downsample not so aggresively, or enforce the input layer weight matrix to have rank 1.

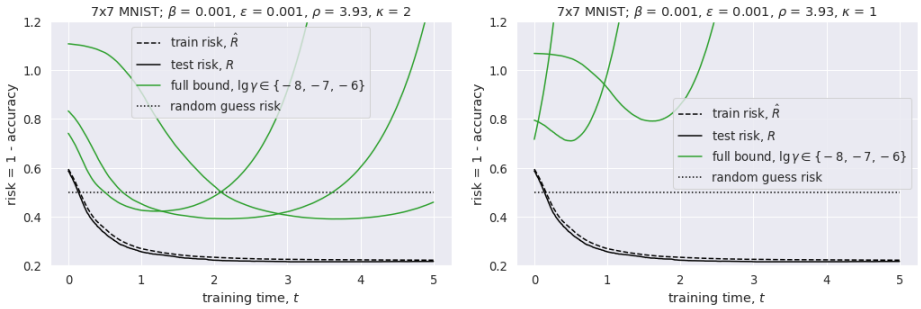


Figure 6.2: We consider 7x7 binary MNIST, $L = 2$, $\beta = 0.001$, $\epsilon = 0.001$, and compare different kappas of Theorem 6.5.2. The bound for $\kappa = 2$ is much stronger than that for $\kappa = 1$.

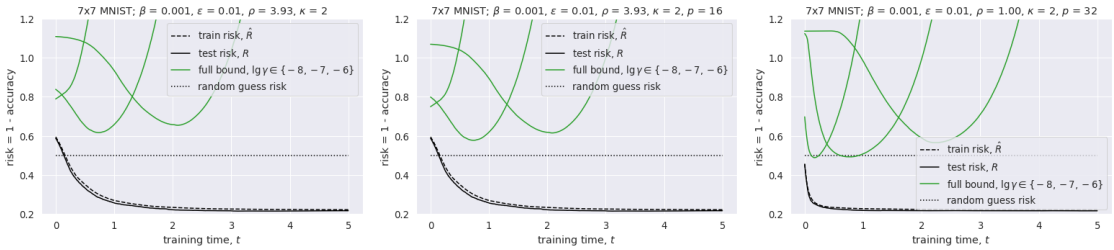


Figure 6.3: We consider 7x7 binary MNIST, $L = 2$, $\beta = 0.001$, $\epsilon = 0.01$, $\kappa = 2$, and vary the stable rank at initialization ρ and floating point precision p. Initializing the input layer with a rank one matrix considerably improves the bound. Moreover, it also improves the convergence speed.

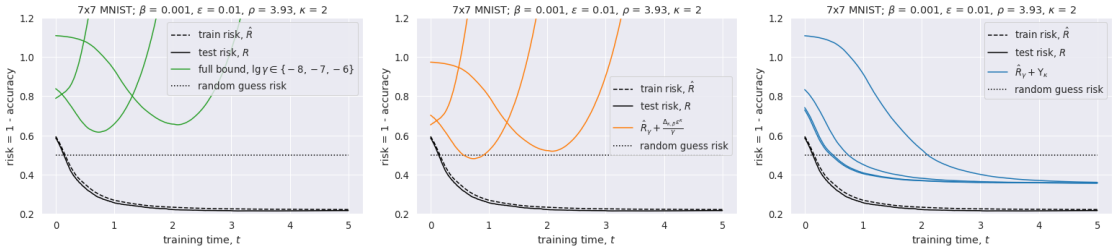


Figure 6.4: We consider 7x7 binary MNIST, $L = 2$, $\beta = 0.001$, $\epsilon = 0.01$, $\kappa = 2$, and compare different components of the bound. The left figure corresponds to the full bound, while for the central one we forget about the generalization gap bound for the proxy model Υ_κ, and for the rightmost one, we forget about the deviation term $\frac{\Delta_{\kappa,\beta}\epsilon^\kappa}{\gamma}$. We see that both terms are of the same order; one therefore has to work on reducing both in order to reduce the overall bound.

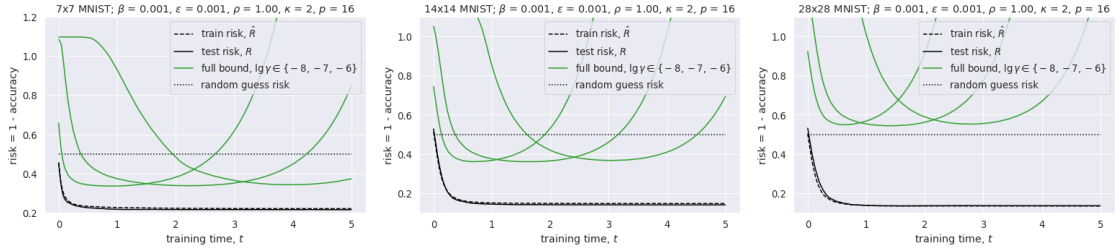


Figure 6.5: We consider binary MNIST, $L = 2$, $\beta = 0.001$, $\epsilon = 0.001$, $\kappa = 2$, $p = 16$, rank one input layer initialization, and vary image dimensions. In this "gentle" scenario, the bound stays non-vacuous for 14x14 MNIST, and only slightly exceeds the random guess risk for the full-sized, 28x28 MNIST.

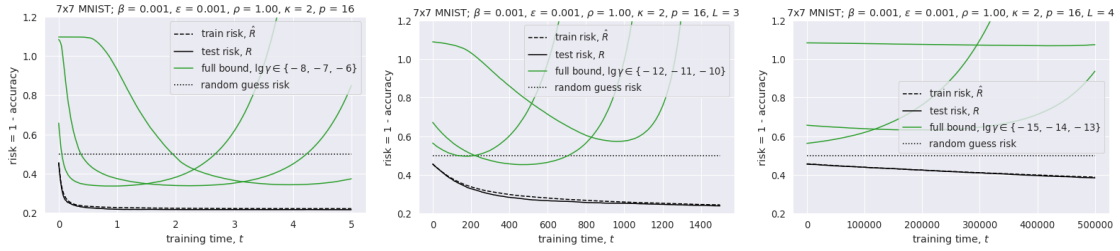


Figure 6.6: We consider binary 7x7 MNIST, $\beta = 0.001$, $\epsilon = 0.001$, $\kappa = 2$, $p = 16$, rank one input layer initialization, and vary depth. Even in this "gentle" scenario, the bound gets considerably worse with depth.

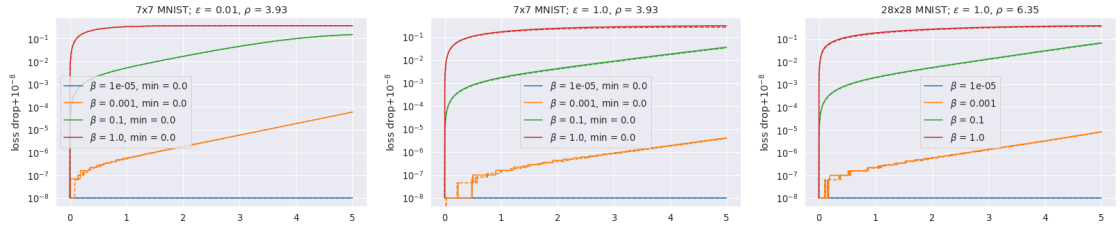


Figure 6.7: We consider binary MNIST, $L = 2$, $\beta = 0.001$, $\epsilon = 0.001$, and measure the differences $\mathcal{L}(0) - \mathcal{L}(t)$ (solid lines) and $\mathcal{L}_X(0) - \mathcal{L}_X(t)$ (dashed lines). Here $\mathcal{L}(t) = \frac{1}{2m} \left\| Y - f^{\epsilon}_{\theta^{\epsilon}(t)}(\tilde{X}) \right\|_F^2$ is the loss at time t, while $\mathcal{L}_X(t) = \frac{1}{2m} \left\| \left(Y - f^{\epsilon}_{\theta^{\epsilon}(t)}(\tilde{X}) \right) \tilde{X}^{\top} \right\|_F^2$ is the "projected" loss at time t. While $\mathcal{L}(t)$ should clearly decrease for gradient descent with small enough steps, it is not *a priori* clear that $\mathcal{L}_X(t)$ also does. As we see from the plots, it does for β large and small, and for ϵ up to 1, which corresponds to conventional ReLU activations. These results validate our Assumption 6.5.1. Note that we added a small quantity 10^{-8} in order to make zero visible.

Observations:

- When we take $\kappa = 2$ and $\beta \leq 0.1$, the bound becomes nonvacuous up to $\epsilon = 0.001$ (Fig. 6.1) and even up to $\epsilon = 0.01$ if we enforce $\rho = 1$ (Fig. 6.3);
- We can have a non-vacuous bound also for 14x14 MNIST if we take $\epsilon = 0.001$, half-precision ($p = 16$), and enforce $\rho = 1$ (Fig. 6.5); the bound slightly exceeds the random guess risk for the full-sized, 28x28 MNIST;
- The bound for $\kappa = 2$ is much stronger than that for $\kappa = 1$ (Fig. 6.2);
- The bound improves and converges as β vanishes (Fig. 6.1);
- Assumption 6.5.1 holds empirically (Fig. 6.7);
- The bound improves if we enforce $\rho = 1$; this also results in faster convergence (Fig. 6.3);
- The bound slightly improves for the half floating point precision (Fig. 6.3);
- For $\epsilon = 0.01$ at the training time, all components of our bound are of the same order, while for larger ϵ, the last one starts to dominate (Fig. 6.4);
- The bound deteriorates consideraly when we increase depth (Fig. 6.6).

6.7 Proof of the main result

6.7.1 Proof of Theorem 6.5.2 for $\kappa = 1$

We start with approximating our learned network $f^{\epsilon}_{\theta^{\epsilon}}$ with $f^{\epsilon}_{\theta^0}$, i.e. with a nonlinear network that uses the weights learned by the linear one. This approximation deviates from the original network the following way: $\forall x \in \mathbb{R}^d, \epsilon \in [0,1]$,

$$\frac{1}{\epsilon}\left\|f^{\epsilon}_{\theta^{\epsilon}}(x) - f^{\epsilon}_{\theta^0}(x)\right\| = \frac{1}{\epsilon}\left\|\int_0^{\epsilon} \frac{\partial f^{\epsilon}_{\theta^{\tau}}(x)}{\partial \tau}\, d\tau\right\| \leq \sup_{\tau\in[0,\epsilon]}\left\|\frac{\partial f^{\epsilon}_{\theta^{\tau}}(x)}{\partial \tau}\right\| \leq \sup_{\tau\in[0,\epsilon]}\sum_{l=1}^{L}\left\|\frac{\partial W_l^{\tau}}{\partial \tau}\right\|\prod_{k\neq l}\|W_k^{\tau}\|\|x\| \tag{6.20}$$

since we use Leaky ReLUs. We omitted the argument t for brevity. We get a similar deviation bound on the training dataset:

$$\frac{1}{\epsilon}\left\|f^{\epsilon}_{\theta^{\epsilon}}(\tilde{X}) - f^{\epsilon}_{\theta^0}(\tilde{X})\right\| \leq \sup_{\tau\in[0,\epsilon]}\left\|\frac{\partial W_1^{\tau}\tilde{X}}{\partial \tau}\right\|_F \prod_{k=2}^{L}\|W_k^{\tau}\| + \sup_{\tau\in[0,\epsilon]}\sum_{l=2}^{L}\left\|\frac{\partial W_l^{\tau}}{\partial \tau}\right\|\|W_1^{\tau}\tilde{X}\|_F \prod_{k\in[2:L]\setminus\{l\}}\|W_k^{\tau}\|. \tag{6.21}$$

We complete the above deviation bound with bounding weight norms and norms of weight derivatives:

Lemma 6.7.1. *Under Assumption 6.5.1,* $\forall \tau \in [0,1], t \geq 0$ $\frac{1}{\sqrt{m}}\|W_1^{\tau}(t)\tilde{X}\|_F \leq \rho u(t)$, $\frac{1}{\sqrt{m}}\left\|\frac{\partial W_1^{\tau}\tilde{X}}{\partial \tau}(t)\right\|_F \leq v(t)$, *and* $\forall l \in [L]$ $\|W_l^{\tau}(t)\| \leq u(t)$, $\left\|\frac{\partial W_l^{\tau}}{\partial \tau}(t)\right\| \leq v(t)$ *for* u, v *defined in Theorem 6.5.2.*

See Section 6.7.3 and Appendix D.4.1 for the proof.

Now we can relate the risk of the original model with the risk of the approximation. Since $r \le r_\gamma^C \le r_\gamma$ and $r_\gamma^C(\cdot, y)$ is $1/\gamma$-Lipschitz for any fixed y,

$$\begin{aligned} R(f^\epsilon_{\theta^\epsilon}) - \hat{R}_\gamma(f^\epsilon_{\theta^\epsilon}) &\le R^C_\gamma(f^\epsilon_{\theta^\epsilon}) - \hat{R}^C_\gamma(f^\epsilon_{\theta^\epsilon}) \\ &\le R^C_\gamma(f^\epsilon_{\theta^0}) - \hat{R}^C_\gamma(f^\epsilon_{\theta^0}) + \frac{1}{\gamma}\mathbb{E}\,\|f^\epsilon_{\theta^\epsilon}(\tilde{x}) - f^\epsilon_{\theta^0}(\tilde{x})\| + \frac{1}{\gamma m}\left\|f^\epsilon_{\theta^\epsilon}(\tilde{X}) - f^\epsilon_{\theta^0}(\tilde{X})\right\|_1, \end{aligned} \tag{6.22}$$

where the expectation is over the data distribution $\mathcal{D}$, and $\tilde{x} = \Sigma^{-1/2}x$ is the actual input of the network.

As for the last term, the deviation on the train dataset, we use that $\|z\|_1 \le \sqrt{m}\|z\|$ for any $z \in \mathbb{R}^m$, and Eq. (6.21). As for the deviation on the test dataset, due to Eq. (6.20) and Lemma 6.7.1, in order to bound the last term, it suffices to bound $\mathbb{E}\,\|\tilde{x}\|$. Since $\Sigma = \mathbb{E}\,[xx^\top]$, we get

$$\mathbb{E}\,\|\Sigma^{-1/2}x\|^2 = \mathbb{E}\,\left[x^\top\Sigma^{-1}x\right] = \mathbb{E}\,\operatorname{tr}\left[xx^\top\Sigma^{-1}\right] = \operatorname{tr}[I_d] = d. \tag{6.23}$$

This gives $\mathbb{E}\,\|\Sigma^{-1/2}x\| \le \sqrt{\mathbb{E}\,\|\Sigma^{-1/2}x\|^2} \le \sqrt{d}$.

The first two terms is a generalization gap of the proxy-model. We use a simple counting-based bound:

$$R^C_\gamma\left(f^\epsilon_{\theta^0(t)}\right) - \hat{R}^C_\gamma\left(f^\epsilon_{\theta^0(t)}\right) \le \sqrt{\frac{\ln|\mathcal{F}^\epsilon_1(t;\theta(0))| - \ln\delta}{2m}}, \tag{6.24}$$

w.p. $\ge 1-\delta$ over sampling the dataset (X, Y), where $\mathcal{F}^\epsilon_1(t;\theta(0))$ denotes the set of functions representable with $f^\epsilon_{\theta^0(t)}$ for a given initial weights $\theta(0)$, where $\theta^0(t)$ is a result of running the gradient flow Eq. (6.4) for time t. As long as we work with finite precision, this class is finite:

Lemma 6.7.2. $\forall\theta(0), \epsilon, t \ge 0$, $|\mathcal{F}^\epsilon_1(t;\theta(0))| \le 2^{pd}$.

Proof. Since we run our gradient flow Eq. (6.4) on whitened data to optimize squared loss, the initial weights are fixed, and the network we train is linear, the resulting weights depend only on $Y\tilde{X}^\top$ which has d parameters. Since each function in our class is completely defined with the resulting weights, and each weight occupies p bits, we get the above class size. □

We could have used a classical VC-dimension-based bound instead (Vapnik and Chervonenkis, 1971). However, we found it to be numerically larger compared to the counting-based bound above.

This finalizes the proof of Theorem 6.5.2 for $\kappa = 1$.

6.7.2 Proof of Theorem 6.5.2 for $\kappa = 2$

What changes for $\kappa = 2$ is a proxy-model. Consider the following:

$$\tilde{f}^\epsilon_{\theta^0,\theta^\epsilon}(x) = f^\epsilon_{\theta^0}(x) + \left(f^\epsilon_{\theta^\epsilon}(\tilde{X}) - f^\epsilon_{\theta^0}(\tilde{X})\right)\tilde{X}^+x. \tag{6.25}$$

That is, we take the same proxy-model as before, but we add a linear correction term. This correction term aims to fit the proxy model to the original one, $f^\epsilon_{\theta^\epsilon}$, on the training dataset $\tilde{X}$. We prove the following lemma in Appendix D.3.1:

Lemma 6.7.3. *Under the premise of Lemma 6.7.1, $\forall t \geq 0$ $\forall \epsilon \in [0,1]$ $\forall x \in \mathbb{R}^d$ we have*

$$\left\| f^{\epsilon}_{\theta^{\epsilon}(t)}(x) - \tilde{f}^{\epsilon}_{\theta^0(t),\theta^{\epsilon}(t)}(x) \right\| \leq (L-1)(L+1+\rho(L-1))u^{L-1}(t)v(t)\|x\|\epsilon^2; \tag{6.26}$$

$$\left\| f^{\epsilon}_{\theta^{\epsilon}(t)}(\tilde{X}) - \tilde{f}^{\epsilon}_{\theta^0(t),\theta^{\epsilon}(t)}(\tilde{X}) \right\| \leq 2(L-1)(1+\rho(L-1))u^{L-1}(t)v(t)\sqrt{m}\epsilon^2. \tag{6.27}$$

The deviation now scales as ϵ^2 instead of ϵ.

What remains is to bound the size of $\mathcal{F}^{\epsilon}_2(t;\theta(0))$, which denotes the set of functions representable with our new proxy-model $\tilde{f}^{\epsilon}_{\theta^0(t),\theta^{\epsilon}(t)}$ for given initial weights $\theta(0)$. Since $\tilde{f}^{\epsilon}_{\theta^0(t),\theta^{\epsilon}(t)}$ is a sum of $f^{\epsilon}_{\theta^0(t)}$ and a linear model, its size is at most 2^{pd} times larger:

$$|\mathcal{F}^{\epsilon}_2(t;\theta(0))| \leq |\mathcal{F}^{\epsilon}_1(t;\theta(0))|2^{pd} \leq 2^{2pd}. \tag{6.28}$$

This finalizes the proof of Theorem 6.5.2 for $\kappa = 2$.

6.7.3 Proof of Lemma 6.7.1

Below, we are going to present the proof only for $L = 2$, and defer the case of $L \geq 3$ to Appendix D.4.1.

Weight norms

Let us expand the weight evolution Eq. (6.4):

$$\frac{dW^{\tau}_1}{dt} = \left[D^{\tau} \odot W^{\tau,\top}_2 \Xi^{\tau}\right] \tilde{X}^{\top}, \qquad \frac{dW^{\tau}_2}{dt} = \Xi^{\tau}\left[D^{\tau} \odot W^{\tau}_1 \tilde{X}\right]^{\top}, \tag{6.29}$$

where we define

$$\Xi^{\tau} = \frac{1}{m}\left(Y - W^{\tau}_2\left[D^{\tau} \odot W^{\tau}_1 \tilde{X}\right]\right), \tag{6.30}$$

and $D^{\tau} = (\phi^{\epsilon})'\left(W^{\tau}_1 \tilde{X}\right)$ is a $n \times m$ matrix with entries equal to 1 or $1-\tau$. We express it as $D^{\tau} = (1-\tau)1_{n\times m} + \tau\Delta$, where Δ is a $n \times m$ 0-1 matrix.

Let us bound the evolution of weight norms:

$$\frac{d\|W^{\tau}_1\|}{dt} \leq \left\|\frac{dW^{\tau}_1}{dt}\right\| \leq (1-\tau)\|W^{\tau}_2\|\left\|\Xi^{\tau}\tilde{X}^{\top}\right\| + \tau\left\|\Delta \odot W^{\tau,\top}_2 \Xi^{\tau}\right\|\left\|\tilde{X}\right\|. \tag{6.31}$$

Since multiplying by a 0-1 matrix elementwise does not increase Frobenius norm, we get

$$\left\|\Delta \odot W^{\tau,\top}_2 \Xi^{\tau}\right\| \leq \left\|\Delta \odot W^{\tau,\top}_2 \Xi^{\tau}\right\|_F \leq \|W^{\tau}_2\|\|\Xi^{\tau}\|_F. \tag{6.32}$$

Noting that Ξ^{τ} and $\Xi^{\tau}\tilde{X}^{\top}$ are row matrices, this results in

$$\frac{d\|W^{\tau}_1\|}{dt} \leq \left((1-\tau)\|\Xi^{\tau}\tilde{X}^{\top}\| + \tau\|\Xi^{\tau}\|\|\tilde{X}\|\right)\|W^{\tau}_2\|. \tag{6.33}$$

By a similar reasoning,

$$\frac{d\|W^{\tau}_2\|}{dt} \leq (1-\tau)\|\Xi^{\tau}\tilde{X}^{\top}\|\|W^{\tau}_1\| + \tau\|\Xi^{\tau}\|\|W^{\tau}_1\tilde{X}\|_F. \tag{6.34}$$

Consider the following system of ODEs:

$$\frac{dg_1(t)}{dt} = \bar{s}^2 g_2(t), \quad \frac{dg_2(t)}{dt} = g_1(t), \qquad g_1(0) = \beta\bar{s}_\rho; \quad g_2(0) = \beta. \tag{6.35}$$

We then make use of the following lemma which we prove in Appendix D.3.2:

Lemma 6.7.4. $\|\Xi_\tau\| = \|\Xi_\tau\|_F \leq \frac{1}{\sqrt{m}}(1+\rho\beta^L)$. *If we additionally take Assumption 6.5.1 then* $\|\Xi_\tau\tilde{X}^\top\| = \|\Xi_\tau\tilde{X}^\top\|_F \leq s + \rho\beta^L$.

This lemma implies $g_1(t) \geq (1-\tau)(s+\rho\beta^2)\|W_1^\tau(t)\| + \tau(1+\rho\beta^2)\|W_1^\tau(t)\tilde{X}\|_F$ and $g_2(t) \geq \|W_2^\tau(t)\|$.

The above system of ODEs could be solved analytically:

$$g_1(t) = \beta\sqrt{\bar{s}_\rho - \bar{s}_1}\cosh\left(\bar{s}t + \tanh^{-1}\left(\frac{\bar{s}_1}{\bar{s}_\rho}\right)\right), \qquad g_2(t) = \sqrt{\bar{\beta}^2 - \beta^2}\sinh\left(\bar{s}t + \tanh^{-1}\left(\frac{\bar{s}_1}{\bar{s}_\rho}\right)\right), \tag{6.36}$$

where $\bar{\beta} = \beta\frac{\bar{s}_\rho}{\bar{s}_1}$. This gives the bound for the input layer weight norms:

$$\|W_1^\tau(t)\| \leq \beta + \bar{s}\int_0^t g_2(t)\,dt = \beta - \bar{\beta} + \sqrt{\bar{\beta}^2 - \beta^2}\cosh\left(\bar{s}t + \tanh^{-1}\left(\frac{\bar{s}_1}{\bar{s}_\rho}\right)\right). \tag{6.37}$$

For further analysis, we will need simpler-looking bounds. Consider a looser bound: $g_2(t) \leq u(t)$ and $g_1(t) \leq \bar{s}u(t)$, where

$$\frac{du(t)}{dt} = \bar{s}u(t), \qquad u(0) = \beta\frac{\bar{s}_\rho}{\bar{s}_1}. \tag{6.38}$$

This ODE solves as $u(t) = \bar{\beta}e^{\bar{s}t}$.

So, we have $\|W_2^\tau(t)\| \leq u(t)$. As for the input layer weights, we get

$$\|W_1^\tau(t)\| \leq \beta + \bar{s}\int_0^t u(t)\,dt = \beta - \bar{\beta} + u(t) \tag{6.39}$$

Similarly, $\frac{1}{\sqrt{m}}\|W_1^\tau(t)\tilde{X}\|_F = \beta\rho - \bar{\beta} + u(t)$. As we do not want additive terms, we bound from above: $\|W_1^\tau(t)\| \leq u(t)$ and $\frac{1}{\sqrt{m}}\|W_1^\tau(t)\tilde{X}\| \leq \rho u(t)$.

Norms of weight derivatives

In Appendix D.3.3, we follow the same logic to demonstrate that $\left\|\frac{dW_1^\tau}{d\tau}\right\|$, $\frac{1}{\sqrt{m}}\left\|\frac{dW_1^\tau}{d\tau}\tilde{X}\right\|_F$, and $\left\|\frac{dW_2^\tau}{d\tau}\right\|$ are all bounded by the same v which satisfies

$$\frac{dv}{dt} = v\left[(1+\rho)u^2 + \bar{s}\right] + u\left[\bar{1} + \rho u^2\right], \quad v(0) = 0. \tag{6.40}$$

It is a linear ODE in $v(t)$, and $u(t)$ is given: $u(t) = \bar{\beta}e^{\bar{s}t}$. We solve it in Appendix D.3.4 to get $v(t)$ from Theorem 6.5.2.

6.8 Conclusion

We have derived a novel generalization bound for LekyReLU networks. Our bound could be evaluated before the actual training and does not depend on network width. Our bound becomes non-vacuous for partially-trained nets with activation functions close to being linear.

7 Conclusions

This chapter summarizes the contributions of the present thesis, as well as directions for future work. Please, see Chapter 1 for more details.

7.1 Contributions

- Chapter 2 provides a detailed survey on NTK theory: one of the infinite-width limit theories for neural networks.
- In Chapter 3, we prove a generalized version of the Master theorem, the main limit theorem of Tensor Programs. Compared to the old theorem Theorem 1.4.1, the new one Theorem 1.4.2 is valid for non-Gaussian weight initializations, and establishes convergence in L^p for any p, in addition to almost sure convergence.
- In Chapter 4, we establish a tail bound for the limits predicted by the Master Theorem. To the best of our knowledge, no tail bounds applicable to generic Tensor Programs have been available in the literature by the moment of writing this thesis.
- In Chapter 5, we study minima of a L_2-regularized loss function for fully-connected networks. By reformulating this loss function in terms of covariance matrices, we demonstrate that adding new neurons on top of $N(N+1)$ for N being the dataset size, cannot improve the loss. This suggests that the infinite-width limit studied in the previous chapters could be achieved already for finite width in some cases.
- Finally, in Chapter 6, we present a novel generalization bound that is (1) a-priori, i.e. computable before training the network, and (2) non-vacuous when activation functions are close to being linear. To the best of our knowledge, this is the first non-vacuous a-priori generalization bound available in the literature.

7.2 Future work

- Our non-Gaussian Master Theorem, Theorem 1.4.2, could be improved in several directions. First, generalizing it to non-smooth nonlinearities would greatly improve applicability, as popular activation functions like ReLU result in non-smooth nonlinearities in the corresponding

Tensor Programs. Second, our preliminary experimental results suggest that existence of only four moments of the matrix entry distribution suffices for the limit to hold. Third, we could prove a Master Theorem for rotationally invariant weight initialization, which are not iid and include popular orthogonal initializations, in the spirit of corresponding works in Approximate Message Passing (Dudeja et al., 2023; Wang et al., 2024).

- As discussed in Section 1.5, the tail bound we present in Chapter 4 is quite loose in its dependence on depth and on the Lipschitz constant of nonlinearities. Moreover, the present bound is currently proven only for a restricted class of Tensor Programs. We believe that the technique of Hanin and Nica (2020b) could be well-adapted to general Tensor Programs.

- The work of Hanin and Nica (2020b) demonstrates that the very first forward pass of a fully-connected network admits a meaningful limit when depth and width go to infinity proportionally. We conjecture that under suitable conditions on nonlinearities, the same result holds for generic Tensor Programs. As the depth of a Tensor Program expressing a neural network training process is proportional to the neural network depth times the number of training steps, the typical Tensor Program depth must be at least comparable to a typical neural nework width. Therefore the proportional limit might provide a better approximation to real-sized neural networks than (any of) infinite width limits (e.g. μP or NTK). We henceforth see it as an important direction for future work.

- Our loss landscape analysis for L_2-regularized neural nets proves that the global minimum value of the regularized loss cannot be improved when the width exceeds a certain threshold. This threshold grows quadratically with the dataset size. This result deals with global minima directly without analysing the optimization procedure. It would be interesting to investigate if the optimization becomes easier when the width grows beyond this threshold. It would also be interesting to investigate if there is any relation to a *linear* threshold which guarantees that all local minima are global for *unregularized* neural nets (see e.g. Yu and Chen (1995); Nguyen and Hein (2017); Nguyen (2019b, 2021)).

- The generalization bound presented in Chapter 6 could be improved in several ways. First, the deviation bound is not optimal and could be tighten. Second, the generalization bound for proxy-models is quite naive; we believe that better bounds are possible with a thorough analysis. That is, since our proxy-models utilize the trained weights of a linear network, while the latter could be obtained explictly (Saxe et al., 2013), we should be able to get a tight generalization bound. This is analogous to kernels methods for which the model is integrable, and an explicit generalization bound is given e.g. in Bordelon et al. (2024).

- Lastly, we present a proxy-model that approximates the original model up to quadratic terms; whether a higher-order approximation of low complexity exists or not, is an open question. To be specific, we are looking for a proxy lying in a parametric class with $O_{n\to\infty}(1)$ parameters. An affirmative answer would result in better generalization bounds, and also provide a good approximative model for neural networks in the beginning of training (as the deviation diverges with time). A negative answer would indicate that even if we take the implicit bias of gradient descent into account, the class of models realizable by a wide neural network cannot be close up to $O(\epsilon^3)$ to any simple model class.

A Appendix: Non-Gaussian Tensor Programs

A.1 Smoothing by Bump Function

Lemma A.1.1. *Let $\phi : \mathbb{R}^k \to \mathbb{R}$. Then for any $\delta > 0$ there exists $\phi^\delta \in C^\infty(\mathbb{R}^k)$ such that*

1. *For any $x \in \mathbb{R}^k$,*
$$|\phi(x) - \phi^\delta(x)| \le \sup_{x_1, x_2 \in \bar{B}_\delta(x)} |\phi(x_1) - \phi(x_2)|,$$
where $\bar{B}_\delta(x)$ is a closed δ-vicinity of x;

2. *If ϕ is polynomially bounded then ϕ^δ is polynomially smooth;*

3. *If ϕ is Lipschitz with a Lipschitz constant λ, then ϕ^δ is also Lipschitz with a Lipschitz constant λ and*
$$\sup_{x \in \mathbb{R}^k} |\phi(x) - \phi^\delta(x)| \le 2\lambda\delta.$$

Proof. Let ζ be a standard bump function with k arguments:

$$\zeta(z) = Ce^{-\frac{1}{1-\|z\|_2^2}} \mathbb{I}(\|z\|_2 \le 1). \tag{A.1}$$

It is a non-negative $C^\infty(\mathbb{R}^k)$ function that is supported supported on a unit ball. We choose the constant C so that ζ sums to one.

Claim 1 Let $\zeta^\delta(x) \stackrel{\text{def}}{=} (1/\delta)^k \zeta(x/\delta)$. Set $\phi^\delta(x) \stackrel{\text{def}}{=} [\phi * \zeta^\delta](x)$. Then

$$\begin{aligned}
\phi^\delta(x) &= \int_{\mathbb{R}^k} \zeta^\delta(y)\phi(x-y)\,dy \\
&= (1/\delta)^k \int_{\mathbb{R}^k} \zeta(y/\delta)\phi(x-y)\,dy \\
&= \int_{\mathbb{R}^k} \zeta(z)\phi(x - z\delta)\,dz.
\end{aligned}$$

We have for any $x \in \mathbb{R}^k$,

$$\begin{aligned}
|\phi^\delta(x) - \phi(x)| &= \left| \int_{\mathbb{R}^k} \zeta(z) \left(\phi(x - z\delta) - \phi(x)\right) \, dz \right| \\
&\le \left| \int_{\|z\|_2 \le 1} \zeta(z) \, dz \right| \sup_{\|z\|_2 \le 1} |\phi(x - z\delta) - \phi(x)| \\
&\le \sup_{x_1, x_2 \in \bar{B}_\delta(x)} |\phi(x_1) - \phi(x_2)|.
\end{aligned}$$

Claim 2 Suppose ϕ is polynomially bounded, i.e. $|\phi(x)| \le C(1 + \|x\|_p^p)$ for some $p \ge 1$. For any $r \ge 0$ and any $j_1, \ldots, j_r \in [k]$,

$$\begin{aligned}
\partial_{j_1,\ldots,j_r} \phi^\delta(x) &= (1/\delta)^{k + \sum_{i=1}^r j_i} \int_{\mathbb{R}^k} \partial_{j_1,\ldots,j_r} \zeta((x - y)/\delta) \phi(y) \, dy \\
&= (1/\delta)^{k + \sum_{i=1}^r j_i} \int_{\|y\|_2 \le \delta} \partial_{j_1,\ldots,j_r} \zeta(y/\delta) \phi(x - y) \, dy \\
&= (1/\delta)^k \int_{\|z\|_2 \le 1} \partial_{j_1,\ldots,j_r} \zeta(z) \phi(x - z\delta) \, dz.
\end{aligned}$$

Therefore,

$$\begin{aligned}
\left|\partial_{j_1,\ldots,j_r} \phi^\delta(x)\right| &\le (1/\delta)^k \sup_{\|z\|_2 \le 1} |\phi(x - z\delta)| \left| \int_{\|z\|_2 \le 1} \partial_{j_1,\ldots,j_r} \zeta(z) \, dz \right| \\
&\le C(1/\delta)^k \sup_{\|z\|_2 \le 1} (1 + \|x - z\delta\|_p^p) \le C(1/\delta)^k \sup_{\|z\|_2 \le 1} (1 + 2^p \|x\|_p^p + 2^p \delta^p \|z\|_p^p) \\
&\le C(1/\delta)^k \sup_{\|z\|_2 \le 1} (1 + 2^p \|x\|_p^p + 2^p \delta^p p^p \|z\|_\infty^p) \le C(1/\delta)^k (1 + 2^p \|x\|_p^p + (2p\delta)^p),
\end{aligned}$$

which implies that $\partial_{j_1,\ldots,j_r} \phi^\delta$ is polynomially bounded.

Claim 3 Suppose ϕ is Lipschitz with a Lipschitz constant λ. Let us check that ϕ^δ is also Lipschitz with a Lipschitz constant λ:

$$\begin{aligned}
|\phi^\delta(x_1) - \phi^\delta(x_2)| &= \left| \int_{\mathbb{R}^k} \zeta(z) \left(\phi(x_1 - z\delta) - \phi(x_2 - z\delta)\right) \, dz \right| \\
&\le \lambda |x_1 - x_2| \int_{\mathbb{R}^k} \zeta(z) \, dz = \lambda |x_1 - x_2|.
\end{aligned}$$

Then for any $x \in \mathbb{R}^k$,

$$|\phi(x) - \phi^\delta(x)| \le \sup_{x_1, x_2 \in \bar{B}_\delta(x)} |\phi(x_1) - \phi(x_2)| \le \lambda \sup_{x_1, x_2 \in \bar{B}_\delta(x)} \|x_1 - x_2\|_2 \le 2\lambda\delta. \tag{A.2}$$

□

A.2 Additional Applications of Theorem 3.3.7

TP3 demonstrated the following applications of Theorem 3.3.4: the semi-circle law for Gaussian orthogonal ensembles (GOE), the Marchenko-Pastur law for Gaussian Wishart ensembles, and the free independence principle for neural nets with Gaussian initialization. Here we generalize all the above results to non-Gaussian weight initializations.

A.2.1 Semicircle Law for Non-Gaussian Wigner Ensembles

The semicircle law for GOE proven in Yang (2020b) is the following result:

Theorem A.2.1. *For each $n \geq 1$, define the random matrix $A = W + W^\top$ for iid Gaussian matrix $W \in \mathbb{R}^{n\times n}$, $W_{\alpha\beta} \sim \mathcal{N}(0, 1/2n)$. Let $\lambda_1, \ldots, \lambda_n \in \mathbb{R}$ be the eigenvalues of A and $\mu_n = \frac{1}{n}\sum_{\alpha=1}^n \delta_{\lambda_\alpha}$ be their empirical distribution. Then $\mu_n \xrightarrow{a.s.} \mu_{sc}$, where μ_{sc} is the distribution with density $\propto \sqrt{4-x^2}$.*

Here by almost sure (weak) convergence of measures we mean that for every compactly supported, continuous $\phi : \mathbb{R} \to \mathbb{R}$, as $n \to \infty$, we have

$$\frac{1}{n}\sum_{\alpha=1}^n \phi(\lambda_\alpha) \xrightarrow{\text{a.s.}} \mathbb{E}_{\lambda\sim\mu_{sc}} \phi(\lambda). \tag{A.3}$$

In order to be able to apply Tensor Programs machinery to prove this result, Yang (2020b) used the moment method which states that it suffices to prove almost sure convergene of moments of μ_n:

$$\mathbb{E}_{\lambda\sim\mu_n} \lambda^r = \frac{1}{n}\sum_{\alpha=1}^n \lambda_\alpha^r = n^{-1}\operatorname{tr}(A^r) \xrightarrow{\text{a.s.}} \mathbb{E}_{\lambda\sim\mu_{sc}} \lambda^r \quad \forall r \in \mathbb{N}. \tag{A.4}$$

The trace can be expressed as $\operatorname{tr}(A^r) = \mathbb{E}_z[z^\top A^r z]$ for $z_\alpha \sim \mathcal{N}(0,1)$ iid for each $\alpha \in [n]$. For Gaussian z, $A^r z$ can be expressed as a vector in the following Tensor Program:

$$g^1 = z, \quad g^2 = Wz, \quad g^3 = W^\top z, \tag{A.5}$$

$$g^{2k} = W(g^{2k-1} + g^{2k-2}), \quad g^{2k+1} = W^\top(g^{2k-1} + g^{2k-2}) \quad \forall k \in [2, r]. \tag{A.6}$$

Here each nonlinearity simply adds two vectors. Note that these nonlinearities are linear polynomials. Then we get

$$\begin{aligned} n^{-1}\operatorname{tr}(A^r) &= \frac{1}{n}\mathbb{E}_{z\sim\mathcal{N}(0,1)}[z^\top A^r z] \\ &= \frac{1}{n}\sum_{\alpha=1}^n \mathbb{E}_{g_\alpha^1\sim\mathcal{N}(0,1)}\left[g_\alpha^1\left(g_\alpha^{2r} + g_\alpha^{2r+1}\right)\right] = \frac{1}{n}\sum_{\alpha=1}^n \mathbb{E}_{g_\alpha^1\sim\mathcal{N}(0,1)}\psi(g_\alpha^1, \ldots, g_\alpha^{2r+1}), \end{aligned} \tag{A.7}$$

where $\psi(x_1, \ldots, x_{2r+1}) = x_1(x_{2r} + x_{2r+1})$ — a quadratic polynomial.

The only thing that prevents us from directly applying Theorem 3.3.4 to the expression above is conditional expectation. Yang (2020b) proves the following conditional theorem:

Theorem A.2.2 (Gaussian Conditional Master Theorem, Yang (2020b))**.** *Consider Setup 3.3.3 and assume ψ to be quadratically bounded and all the nonlinearities to be linearly bounded. Let S be any σ-subalgebra of the σ-algebra generated by the random sampling of the program. Then, as $n \to \infty$,*

$$\frac{1}{n}\sum_{\alpha=1}^{n} \mathbb{E}_S \psi\left(g_\alpha^1, \dots, g_\alpha^M, c^1, \dots, c^M\right) \xrightarrow{\text{a.s.}} \mathring{\Psi}, \tag{A.8}$$

where $\mathring{\Psi}$ is the same as in Theorem 3.3.2.

We had to come back to the original form of Master Theorems (see Theorem 3.3.2) since there is a distinction between nonlinearities (which have to be linearly bounded) and test functions ψ, which could be quadratically bounded.

The above conditional theorem implies $n^{-1}\operatorname{tr}(A^r) \xrightarrow{\text{a.s.}} \mathring{\Psi}$ for certain $\mathring{\Psi}$. Comparing with Eq. (A.4), what remains to establish Theorem A.2.1, is to prove that $\mathring{\Psi}$ equals to the r-th moment of the semicircle law. One can find the proof in Yang (2020b); we do not reproduce it here.

Non-Gaussian Conditional Master Theorem. Consider Setup 3.3.6 and let S be a subset of initial vectors. Then for any scalar c^i in the program, $\mathbb{E}\,|\mathbb{E}_S c^i - \mathring{c}^i|^p \le \mathbb{E}\,|c^i - \mathring{c}^i|^p$, which converges to zero by Theorem 3.3.7. Therefore $\mathbb{E}_S c^i$ converges to $\mathring{c}^i$ in L^p.

Consider now the same interpolation between Gaussian and non-Gaussian weights $A^l(t)$ as in the proof of Theorem 3.3.7, see Section 3.5. Theorem 3.5.2 then gives $\sup_t \mathbb{E}\,|\mathbb{E}_S c^i(t)|^p \le \sup_t \mathbb{E}\,|c^i(t)|^p = O(n^{-p/2})$ for any $p \in [1,\infty)$. Following the argument in Section 3.5, we get $\mathbb{E}_S c^i(1) - \mathbb{E}_S c^i(0) \xrightarrow{\text{a.s.}} 0$. Assuming the conditions of the above Gaussian theorem, Theorem A.2.2, namely, that ϕ^i is quadratically bounded, while ϕ^j for all $j < i$ are linearly bounded, we get $\mathbb{E}_S c^i(0) \xrightarrow{\text{a.s.}} \mathring{c}^i$, and therefore $\mathbb{E}_S c^i(1) \xrightarrow{\text{a.s.}} \mathring{c}^i$:

Theorem A.2.3 (Non-Gaussian Conditional Master Theorem, ours)**.** *Consider Setup 3.3.6 and let S be a subset of initial vectors. Then every scalar c^i conditioned on S converges to the same $\mathring{c}^i$ as in Theorem 3.3.4 in L^p for any $p \in [1,\infty)$:*

$$\mathbb{E}_S c^i \xrightarrow{L^p} \mathring{c}^i \quad \forall p \in [1,\infty). \tag{A.9}$$

If moreover all the nonlinearities are linearly bounded then for any quadratically bounded polynomially smooth ψ,

$$\frac{1}{n}\sum_{\alpha=1}^{n} \mathbb{E}_S \psi\left(g_\alpha^1, \dots, g_\alpha^M, c^1, \dots, c^M\right) \xrightarrow{\text{a.s.}} \mathring{\Psi} \tag{A.10}$$

as $n \to \infty$, where $\mathring{\Psi}$ is the same as in Theorem 3.3.2.

Since the program used to compute the moments conforms not only Setup 3.3.3 but also a more restrictive Setup 3.3.6, we get a full analogue of Theorem A.2.1:

Theorem A.2.4. *For each $n \ge 1$, define the random matrix $A = W + W^\top$ for W being an $n \times n$ matrix with iid entries with zero mean, variance $1/(2n)$, and with all higher moments existing. Let $\lambda_1, \dots, \lambda_n \in \mathbb{R}$ be the eigenvalues of A and $\mu_n = \frac{1}{n}\sum_{\alpha=1}^{n} \delta_{\lambda_\alpha}$ be their empirical distribution. Then $\mu_n \xrightarrow{\text{a.s.}} \mu_{sc}$, where μ_{sc} is the distribution with density $\propto \sqrt{4-x^2}$.*

Relaxing Boundedness. Could we relax the assumption on linear and quadratic boundedness in our Theorem A.2.3? Inspired by Central Limit Theorem and our matrix derivative bound below, Lemma A.10.6, we conjecture the following moment bound that quantifies the rate of L^p convergence:

Conjecture A.2.5. *Under Setup A.2.6 below, $\mathbb{E}\,|c^i - \mathring{c}^i|^p = O(n^{-p/2})$ for any scalar c^i in the program and the corresponding almost sure limit $\mathring{c}^i$, and any $p \in [1, \infty)$.*

Setup A.2.6. *Consider Setup 3.3.6 but replace 5) with 5*) for any initial scalar c^i and any $p \in [1, \infty)$, $\mathbb{E}\,|c^i - \mathring{c}^i|^p = O(n^{-p/2})$, where $\mathring{c}^i$ is the almost sure limit of c^i which exists due to 2).*

If the above conjecture holds, we will get a similar bound for scalars conditioned on S, i.e. $\mathbb{E}\,|\mathbb{E}_S c^i - \mathring{c}^i|^p = O(n^{-p/2})$, which will imply $\mathbb{E}_S c^i \xrightarrow{\text{a.s.}} \mathring{c}^i$ by Borel-Cantelli lemma:

Conjecture A.2.7. *Consider Setup A.2.6 and let S be a subset of initial vectors. Then every scalar c^i conditioned on S converges to the same $\mathring{c}^i$ as in Theorem 3.3.4 almost surely and in L^p for any $p \in [1, \infty)$:*

$$\mathbb{E}_S c^i \xrightarrow{\text{a.s. \& } L^p} \mathring{c}^i \quad \forall p \in [1, \infty). \tag{A.11}$$

We leave proving Conjecture A.2.5 for future work.

A.2.2 Marchenko-Pastur Law for Non-Gaussian Wishart Ensembles

Using exactly the same machinery as for the semi-circle law, Yang (2020b) proves the Marchenko-Pastur law for Wishart ensembles, i.e. for matrices of the form $AA^\top$, where A is an $m \times n$ Gaussian matrix with zero mean and variance n^{-1}, and $m/n \to \rho \in (0, \infty)$ as $n \to \infty$. Our Theorem 3.3.7 (with a remark on programs with variable dimensions, see Section 3.3) gives a similar result with no assumptions on Gaussianity of A:

Theorem A.2.8 (Ours)**.** *For each $n \geq 1$, let A be an $m \times n$ matrix with iid entries with zero mean, variance $1/n$, and with all higher moments existing. Let $\lambda_1, \ldots, \lambda_n \in \mathbb{R}$ be the eigenvalues of $AA^\top$ and $\mu_n = \frac{1}{n}\sum_{\alpha=1}^n \delta_{\lambda_\alpha}$ be their empirical distribution. Let m go to infinity as $n \to \infty$ such that $m/n \to \rho \in (0, \infty)$. Then $\mu_n \xrightarrow{a.s.} \mu_{mp}$, where μ_{mp} has "density" $p_{mp}(x)$ given below:*

$$p_{mp}(x) = \max(0, 1 - \rho^{-1})\delta(x) + \frac{1}{\rho 2\pi x}\sqrt{(b-x)(x-a)}1_{x\in[a,b]}, \tag{A.12}$$

where $\delta(x)$ is the Dirac Delta, $a = (1 - \sqrt{\rho})^2$, and $b = (1 + \sqrt{\rho})^2$.

A.2.3 Free Indepedence Principle for Tensor Programs with Non-Gaussian Weights

Using the same machinery again, Yang (2020b) proves Free Independence Principle for Tensor Programs:

Theorem A.2.9 (Yang (2020b))**.** *Consider Setup 3.3.3 and assume all nonlinearities to be linearly bounded. Let $\mathcal{D}$ denote the collection of diagonal matrices formed from bounded coordinatewise images of vectors in the program:*

$$\mathcal{D} = \left\{\text{diag}\left(\psi\left(g^1, \ldots, g^M\right)\right) : \ \psi : \mathbb{R}^M \to \mathbb{R} \text{ is bounded}\right\}. \tag{A.13}$$

Then $\mathcal{D}$, along with the random matrix collections $\{A, A^\top\}$ for all matrices in the program are almost surely asymptotically free as $n \to \infty$.

Here by almost sure asymptotical freeness we mean the following:

Definition A.2.10. Fix k. Consider collections of random matrices $\mathcal{W}_n^1, \dots, \mathcal{W}_n^k \subseteq \mathbb{R}^{n\times n}$ for each $n \geq 1$, of constant cardinalities (with n). We say $\mathcal{W}_n^1, \dots, \mathcal{W}_n^k$ are almost surely asymptotically freely independent (or just almost surely asymptotically free), if

$$n^{-1} \operatorname{tr}\left(\prod_{i=1}^{k} \left(P_i(\mathcal{W}_n^{j_i}) - \tau_i I\right)\right) \xrightarrow{\text{a.s.}} 0, \tag{A.14}$$

where $\tau_i = n^{-1} \operatorname{tr}(P_i(\mathcal{W}_n^{j_i}))$, P_i is a non-commutative polynomial in $|\mathcal{W}_n^{j_i}|$ variables, $j_1, \dots, j_k \in [k]$ are indices with no two adjacent j_i equal, and $\{P_i\}_i$, $\{j_i\}_i$ independent of n.

The proof strategy for Theorem A.2.9 is very similar to the one of Theorem A.2.1. The trace in each τ_i can be expressed as an expectation $\mathbb{E}_{z\sim\mathcal{N}(0,1)}[z^\top P_i(\mathcal{W}_n^{j_i})z]$ and $P_i(\mathcal{W}_n^{j_i})z$ can be expressed as a vector in a program with linearly bounded nonlinearities (that's the reason why we require ψ in the definition of $\mathcal{D}$ to be bounded). Therefore each τ_i converges by Theorem A.2.2, and therefore can be thought as a scalar in a new Tensor Program. The "outer" trace in Eq. (A.14) can be expressed as conditional expectation using the same trick, and the corresponding Eq. (A.14) again converges by Theorem A.2.2. We refer the reader to Yang (2020b) for details.

Simply replacing Theorem A.2.2 in the above reasoning with its non-Gaussian analogue, Theorem A.2.3, we get a non-Gaussian analogue of Theorem A.2.9:

Theorem A.2.11 (Ours)**.** *Consider Setup 3.3.6 and assume all nonlinearities to be linearly bounded. Let $\mathcal{D}$ denote the collection of diagonal matrices formed from bounded polynomially smooth coordinatewise images of vectors in the program:*

$$\mathcal{D} = \left\{\operatorname{diag}\left(\psi\left(g^1, \dots, g^M\right)\right) : \ \psi : \mathbb{R}^M \to \mathbb{R} \text{ is bounded and polynomially smooth}\right\}. \tag{A.15}$$

Then $\mathcal{D}$, along with the random matrix collections $\{A, A^\top\}$ for all matrices in the program are almost surely asymptotically free as $n \to \infty$.

We had to require all nonlinearities to be linearly bounded and the functions ψ in the definition of digonal matrices $\mathcal{D}$ to be bounded in order to apply a conditional theorem, Theorem A.2.2 or Theorem A.2.3. Note that our original non-Gaussian Master Theorem, Theorem 3.3.7, gives also convergence in L^1, which implies convergence of (full, non-conditional) expectations without requiring linearly bounded nonlinearities[1]. Convergence of full expectations allows us to prove more traditional asymptotic freeness in expectation in our non-Gaussian case:

Definition A.2.12. Fix k. Consider collections of random matrices $\mathcal{W}_n^1, \dots, \mathcal{W}_n^k \subseteq \mathbb{R}^{n\times n}$ for each $n \geq 1$, of constant cardinalities (with n). We say $\mathcal{W}_n^1, \dots, \mathcal{W}_n^k$ are asymptotically freely independent in expectation (or just asymptotically free), if

$$n^{-1}\mathbb{E}\left[\operatorname{tr}\left(\prod_{i=1}^{k} \left(P_i(\mathcal{W}_n^{j_i}) - \tau_i I\right)\right)\right] \to 0, \tag{A.16}$$

where $\tau_i = n^{-1}\mathbb{E}\operatorname{tr}(P_i(\mathcal{W}_n^{j_i}))$, P_i is a non-commutative polynomial in $|\mathcal{W}_n^{j_i}|$ variables, $j_1, \dots, j_k \in [k]$ are indices with no two adjacent j_i equal, and $\{P_i\}_i$, $\{j_i\}_i$ independent of n.

[1] This proves Conjecture A.4 of Yang (2020b) for polynomially smooth nonlinearities

Theorem A.2.13 (Ours)**.** *Consider Setup 3.3.6. Let $\mathcal{D}$ denote the collection of diagonal matrices formed from polynomially smooth (not necessarily bounded) coordinatewise images of vectors in the program:*

$$\mathcal{D} = \left\{ \operatorname{diag}\left(\psi\left(g^1, \ldots, g^M\right)\right) : \ \psi : \mathbb{R}^M \to \mathbb{R} \text{ is polynomially smooth} \right\}. \tag{A.17}$$

Then $\mathcal{D}$, along with the random matrix collections $\{A, A^\top\}$ for all matrices in the program are asymptotically free in expectation as $n \to \infty$.

We use the same tensor program to compute $c^i = \frac{1}{n}\sum_{\alpha\in[n]} z^i_\alpha (P_i(\mathcal{W}^{j_i}_n) z^i)_\alpha$ required to compute the traces $\tau_i = \mathbb{E}_{z^i\sim\mathcal{N}(0,I)} c^i$. Since we now rely on a theorem that does not require linearly bounded nonlinearities, the nonlinearities of the program at hand and ψ in the definition of $\mathcal{D}$ can be polynomially smooth. We embed the programs for $P_i(\mathcal{W}^{j_i}_n)z$ into the "outer" program that computes the scalar $\frac{1}{n}\sum_{\alpha\in[n]} \bar{z}_\alpha \left(\prod_{i=1}^k \left(P_i(\mathcal{W}^{j_i}_n) - c_i I\right)\bar{z}\right)_\alpha$. Taking the expectation over all randomness and applying Theorem A.2.3 gives Eq. (A.16).

A.3 Non-Gaussian Master Theorem for Lipschitz Nonlinearities

One of the limitations of our Theorem 3.3.7 is smoothness requirement. In the present section, we prove a similar result that assumes nonlinearities to be Lipschitz but not necessarily smooth:

Setup A.3.1. *Consider Setup 3.3.6, but replace 3*) and 4*) with the following: 3**) all matrices A^i have iid entries drawn from distributions with zero mean, variance n^{-1}, and a fourth moment bounded by $\nu_4 n^{-2}$ for some $\nu_4 > 0$; 4**) all the nonlinearities ϕ are Lipschitz (but not necessarily smooth).*

Theorem A.3.2 (Non-Gaussian Master Theorem for Lipschitz nonlinearities)**.** *Consider Setup A.3.1. Then, as $n \to \infty$, every scalar c^i converges to the same $\mathring{c}^i$ as in Theorem 3.3.4 almost surely. If, moreover, we require the entries of matrices A^i to be drawn from sub-Gaussian distributions, we get convergence in L^p for any $p \in [1, \infty)$.*

The following result is an immediate consequence of the above theorem:

Corollary A.3.3 (Convergence to GP at initialization)**.** *Consider a neural network $f(\xi) = \frac{1}{\sqrt{n}} V \Phi(\xi)$ with output layer weights V and embedding $\Phi(\xi)$ on input ξ, and where n is the dimension of the Tensor Program we describe next. Suppose 1) $\Phi(\xi)$ can be expressed as a concatenation of a constant (wrt n) number of vectors $x^i \in \mathbb{R}^n$ (c.f. Eq. (3.4)) from a Tensor Program T in Setup A.3.1, 2) entries of the matrices A^i in T are drawn from sub-Gaussian distributions, and 3) V's entries are initialized iid from a sub-Gaussian distribution with mean 0 and variance 1, and are independent from the random objects of the program T.*

Then at initialization, as width tends to infinity, $f(\cdot)$ converges weakly to a Gaussian Process (GP).

Proof. Our goal is to show that for any integer $B > 0$, any batch of inputs $\xi_1, \ldots, \xi_B$ of size B, and any bounded Lipschitz function $\psi : \mathbb{R}^B \to \mathbb{R}$, as $n \to \infty$, $\mathbb{E}\left[\psi\left(f(\xi_1), \ldots, f(\xi_B)\right)\right]$ converges to $\mathbb{E}_Z \psi(Z)$ for Z being a B-dimensional Gaussian that can be expressed as the marginal of a fixed Gaussian process.

Suppose for simplicity that $\Phi(\xi)$ is a single vector x^i from the program T and V is a row-vector of size n. Fix B, a batch of inputs $\xi_1, \dots, \xi_B$, and a bounded Lipschitz function $\psi : \mathbb{R}^B \to \mathbb{R}$. Let T_B be a concatenation of tensor programs that are used to express each $\Phi(\xi_k)$ for $k \in [B]$. Choose $i_1, \dots, i_B$ such that $\Phi(\xi_k)$ is expressed by the vector x^{i_k} in T_B for each $k \in [B]$.

Consider a tensor program $\tilde{T}_B$, augmented with a new matrix $W \in \mathbb{R}^{n\times n}$, that extends T_B: If T_B has size M (i.e., ends at vector g^M), then 1) $\tilde{T}_B$ generates $g^{M+k} \leftarrow W x^{i_k}$ for each $k = 1, \dots, B$ on top of T_B, and 2) $\tilde{T}_B$ generates the scalar $c \stackrel{\text{def}}{=} c^{M+B+1} \leftarrow \frac{1}{n}\sum_{\alpha=1}^n \psi(g_\alpha^{M+1}, \dots, g_\alpha^{M+B})$. The matrix W will be sampled independently from all objects in T_B and to have each row be an iid copy of $\frac{1}{\sqrt{n}}V$. Then

$$(f(\xi_1), \dots, f(\xi_B)) \stackrel{\text{d}}{=} (g_\alpha^{M+1}, \dots, g_\alpha^{M+B})$$

for any $\alpha \in [n]$. In addition, $(g_\alpha^{M+1}, \dots, g_\alpha^{M+B})$ is iid in α conditioned on T_B.

Finally, we apply the Master Theorem to the scalar c. It is easy to check that this $\tilde{T}_B$ satisfies the conditions of Theorem A.3.2 above and therefore $c \to \mathring{c}$ in mean, where $\mathring{c}$ is the limit given by the Gaussian theorem, Theorem 3.3.4, if we assume that all matrix-like weight tensors in $\Phi(\cdot)$, as well as the vector V, had iid entries from the corresponding Gaussian equivalents.

Convergence in mean implies convergence of expectations, therefore $\mathbb{E}\, c \to \mathring{c}$ as $n \to \infty$. However,

$$\mathbb{E}\, c = \frac{1}{n}\sum_{\alpha=1}^n \mathbb{E}\left[\mathbb{E}\left[\psi\left(g_\alpha^{M+1}, \dots, g_\alpha^{M+B}\right) \mid T_B\right]\right] = \mathbb{E}\,\psi\left(g_1^{M+1}, \dots, g_1^{M+B}\right)$$
$$= \mathbb{E}\,\psi\left(f(\xi_1), \dots, f(\xi_B)\right)$$

since the entries of each g^{i_k} are iid conditioned on all random objects of T_B. Examining the explicit form of $\mathring{c}$ given in Yang (2020b), we conclude that when V does not depend on random objects from T_B, the limit is an expectation of $\mathbb{E}_Z \psi(Z)$ for a certain multvariate Gaussian Z.

The above proof extends straightforwardly to the case when T is a concatenation of a finite number of x^i from T. □

Proof of Theorem A.3.2. Let λ be the maximal Lipschitz constant among all ϕ^i in the program. Fix some $\delta > 0$. Let $\phi^{i,\delta}$ be the corresponding polynomially smooth Lipschitz functions given by Lemma A.1.1. WLOG, we substitute δ with $\delta/2\lambda$ so that $\sup_{x\in\mathbb{R}^{2(i-1)}} |\phi^{i,\delta}(x) - \phi^i(x)| \le \delta$.

Define the "smoothed" version of the given Tensor Program:

$$g_\alpha^{i,\delta} \leftarrow \sum_{\beta=1}^n W_{\alpha\beta}^i x_\beta^{i,\delta}, \quad c^{i,\delta} \leftarrow \frac{1}{n}\sum_{\beta=1}^n x_\beta^{i,\delta}, \quad \text{where } x_\alpha^{i,\delta} = \phi^i(g_\alpha^{1,\delta}, \dots, g_\alpha^{i-1,\delta}; c^{1,\delta}, \dots, c^{i-1,\delta}). \tag{A.18}$$

for $i \in [M_0 + 1, M]$, where all W^i are the same as in the original program. All input vectors and scalars coincide with the original program: $g^{i,\delta} = g^i$, $c^{i,\delta} = c^i$ for $i \in [M_0]$.

We consider the same weight interpolation $A^l(t)$ as in the main: $A^l(0)$ corresponds to Gaussian weights $\tilde{A}^l$ with the same mean and variance as A^l, while $A^l(1)$ corresponds to the original weights A^l. Consequently, $g^{i,\delta}(t)$ denotes a vector obtained using $A^1(t), \dots, A^L(t)$, and similarly for $c^{i,\delta}(t)$.

Lemma A.3.4. *Under premise of Theorem A.3.2, for any $i \in [M_0+1, M]$, there exists a polynomial P^i such that for any $n \ge 1$, any $\delta > 0$, and any $t \in [0, 1]$,*

$$\left\|x^i(t) - x^{i,\delta}(t)\right\|_2 \le \sqrt{n}\delta P^i\left(\lambda\|A^1(t)\|_{op}, \dots, \lambda\|A^L(t)\|_{op}\right). \tag{A.19}$$

Proof. We will drop the t-argument in the proof to lighten the exposition. We prove the statement by induction on i.

Induction base: since $g^1 = g^{1,\delta}, \dots, g^{M_0} = g^{M_0,\delta}$,

$$\begin{aligned}&\left\|x^{M_0+1}(t) - x^{M_0+1,\delta}(t)\right\|_2\\ &= \left\|\phi^{M_0+1}(g^1,\dots,g^{M_0};c^1,\dots,c^{M_0}) - \phi^{M_0+1,\delta}(g^{1,\delta},\dots,g^{M_0,\delta};c^{1,\delta},\dots,c^{M_0,\delta})\right\|_2 \le \sqrt{n}\delta. \quad \text{(A.20)}\end{aligned}$$

Here $P^{M_0+1} \equiv 1$; in particular, it does not depend on t.

Suppose the induction hypothesis holds for i. Then we get the following for $i+1$:

$$\begin{aligned}&\left\|x^{i+1} - x^{i+1,\delta}\right\|_2\\ &= \left\|\phi^i(g^1,\dots,g^i;c^1,\dots,c^i) - \phi^{i,\delta}(g^{1,\delta},\dots,g^{i,\delta};c^{1,\delta},\dots,c^{i,\delta})\right\|_2\\ &\le \left\|\phi^i(g^1,\dots,g^i;c^1,\dots,c^i) - \phi^{i,\delta}(g^1,\dots,g^i;c^1,\dots,c^i)\right\|_2\\ &\quad + \left\|\phi^{i,\delta}(g^1,\dots,g^i;c^1,\dots,c^i) - \phi^{i,\delta}(g^{1,\delta},\dots,g^{i,\delta};c^{1,\delta},\dots,c^{i,\delta})\right\|_2\\ &\le \sqrt{n}\delta + \sqrt{\sum_{\alpha=1}^n \left|\phi^{i,\delta}(g^1_\alpha,\dots,g^i_\alpha;c^1,\dots,c^i) - \phi^{i,\delta}(g^{1,\delta}_\alpha,\dots,g^{i,\delta}_\alpha;c^{1,\delta},\dots,c^{i,\delta})\right|^2},\end{aligned}$$

where the last inequality holds due to the approximation property of ϕ^δ.

$$\begin{aligned}&\sum_{\alpha=1}^n \left|\phi^{i,\delta}(g^1_\alpha,\dots,g^i_\alpha;c^1,\dots,c^i) - \phi^{i,\delta}(g^{1,\delta}_\alpha,\dots,g^{i,\delta}_\alpha;c^{1,\delta},\dots,c^{i,\delta})\right|^2\\ &\qquad \le \lambda^2 \sum_{\alpha=1}^n \sum_{i'=M_0+1}^i \left(\left|g^{i'}_\alpha - g^{i',\delta}_\alpha\right|^2 + |c^{i'} - c^{i',\delta}|^2\right)\\ &= \lambda^2 \sum_{i'=M_0+1}^i \left(\|g^{i'} - g^{i',\delta}\|_2^2 + n|c^{i'} - c^{i',\delta}|^2\right) \le \lambda^2 \sum_{i'=M_0+1}^i \left(\|W^{i'}\|_{op}^2 + 1\right) \left\|x^{i'} - x^{i',\delta}\right\|_2^2\\ &\qquad \le n\delta^2\lambda^2 \sum_{i'=M_0+1}^i \left(\|W^{i'}\|_{op}^2 + 1\right) \left(P^{i'}(\lambda\|A^1\|_{op},\dots,\lambda\|A^L\|_{op})\right)^2 \quad \text{(A.21)}\end{aligned}$$

by induction hypothesis. Plugging this expression back, we get

$$\begin{aligned}\left\|x^{i+1} - x^{i+1,\delta}\right\|_2 &\le \sqrt{n}\delta \left(1 + \lambda \sqrt{\sum_{i'=M_0+1}^i \left(\|W^{i'}\|_{op}^2 + 1\right) \left(P^{i'}(\lambda\|A^1\|_{op},\dots,\lambda\|A^L\|_{op})\right)^2}\right)\\ &\le \sqrt{n}\delta \left(1 + \lambda \sum_{i'=M_0+1}^i \left(\|W^{i'}\|_{op} + 1\right) P^{i'}(\lambda\|A^1\|_{op},\dots,\lambda\|A^L\|_{op})\right). \quad \text{(A.22)}\end{aligned}$$

Define $j_{i'}$ such that $W^{i'}$ equals $A^{j_{i'}}$ or its transposition for every $i' \in [M_0+1, M]$. Define the new polynomial P^{i+1} as

$$P^{i+1}(x_1,\dots,x_L) = 1 + \lambda \sum_{i'=M_0+1}^i \left(x_{j_{i'}} + 1\right) P^{i'}(x_1,\dots,x_L), \quad \text{(A.23)}$$

which does not depend on t since neither of $P^{i'}$ for $i' \leq i$ do. This proves the induction. □

The above lemma implies that for any $t \in [0, 1]$,

$$|c^i(t) - c^{i,\delta}(t)| = \frac{1}{n}\|x^i(t) - x^{i,\delta}(t)\|_1 \leq \frac{1}{\sqrt{n}}\|x^i(t) - x^{i,\delta}(t)\|_2 \\ \leq \delta P^i(\lambda\|A^1(t)\|_{op}, \ldots, \lambda\|A^L(t)\|_{op}). \quad \text{(A.24)}$$

Almost Sure Convergence. We have

$$|c^i(1) - c^i(0)| \leq |c^i(1) - c^{i,\delta}(1)| + |c^{i,\delta}(1) - c^{i,\delta}(0)| + |c^i(0) - c^{i,\delta}(0)|. \quad \text{(A.25)}$$

The second term goes almost surely to zero as $n \to \infty$ for any $\delta > 0$ by our "smooth" Master Theorem, Theorem 3.3.7.

Bounds for the first and the last term are given by Eq. (A.24), with $t = 1$ and $t = 0$, respectively. Both bounds have almost sure limits due to the following result:[2]

Theorem A.3.5 (Yin et al. (1984))**.** *Let $\{x_{ij}\}_{i,j=1}^{\infty}$ be an infinite matrix of iid random variables with zero mean and a finite fourth moment, and let X_n be the matrix formed by its top-left $n \times n$ block. Then, as $n \to \infty$,*

$$\frac{1}{n}\|X_n\|_{\text{op}}^2 \xrightarrow{a.s.} \mathbb{E}\left(x_{11}^2\right). \quad \text{(A.26)}$$

Therefore almost surely, there exists $C > 0$ such that for any $\delta > 0$,

$$\limsup_{n\to\infty} |c^i(1) - c^i(0)| \leq C\delta. \quad \text{(A.27)}$$

Hence $|c^i(1) - c^i(0)| \xrightarrow{\text{a.s.}} 0$ as $n \to \infty$. Since $c^i(0) \xrightarrow{\text{a.s.}} \mathring{c}^i$ by Gaussian Master Theorem, Theorem 3.3.4 (which applies in the Lipschitz case), we get the same almost sure limit for $c^i(1)$.

L^p Convergence. For any $p \in [1, \infty)$, computing the p-th moment for both sides of Eq. (A.25),

$$\mathbb{E}\,|c^i(1) - c^i(0)|^p \leq 3^p\mathbb{E}\,|c^i(1) - c^{i,\delta}(1)|^p + 3^p\mathbb{E}\,|c^{i,\delta}(1) - c^{i,\delta}(0)|^p + 3^p\mathbb{E}\,|c^i(0) - c^{i,\delta}(0)|^p. \quad \text{(A.28)}$$

The second term goes to zero as $n \to \infty$ for any $\delta > 0$ by our "smooth" Master Theorem, Theorem 3.3.7 (see also the reasoning in Section 3.5). The first and the last terms are bounded as follows:

$$\mathbb{E}\,|c^i(1) - c^{i,\delta}(1)|^p \leq \delta^p\mathbb{E}\left[\left(P^i(\lambda\|A^1\|_{op}, \ldots, \lambda\|A^L\|_{op})\right)^p\right], \quad \text{(A.29)}$$

$$\mathbb{E}\,|c^i(0) - c^{i,\delta}(0)|^p \leq \delta^p\mathbb{E}\left[\left(P^i(\lambda\|\tilde{A}^1\|_{op}, \ldots, \lambda\|\tilde{A}^L\|_{op})\right)^p\right], \quad \text{(A.30)}$$

which are both bounded uniformly wrt n by $C\delta^p$ for some constant C independent of n since all $A^1, \ldots, A^L$ and $\tilde{A}^1, \ldots, \tilde{A}^L$ are sub-Gaussian and therefore have moments which are uniformly bounded wrt n:

[2] In the theorem, the n-th matrix is formed by the top-left $n \times n$ block of an infnite matrix with iid elements, forming a sequence with mutually dependent elements. At the same time, in the canonical definition of tensor programs (see Yang (2019b), Appendix H), the matrices are sampled independently. However, we can safely assume the same sampling strategy as in the theorem. The reason is that as long as Gaussian Master theorem holds, our results do not depend on how exactly we sample weights since we use only their moments. At the same time, the proof of Gaussian Master theorem we rely on (see Yang (2020b)) also depends merely on weight moments.

Lemma A.3.6. *Let X be an $n \times n$ matrix with iid sub-Gaussian entries with zero mean. Then for any $p \in [1, \infty)$, there exists $C > 0$ such that for any $n \geq 1$, $\mathbb{E}\left(\|X\|_{\text{op}}^p\right) \leq CK^p n^{p/2}$, where $K = \inf\{t > 0 : \mathbb{E}\, e^{X_{11}^2/t^2} \leq 2\}$.*

Proof. We are going to use the following tail bound:

Theorem A.3.7 (Theorem 4.4.5 of Vershynin (2018))**.** *Under premise of the above lemma, there exists $C' > 0$ such that for any $n \geq 1$ and any $t > 0$,*

$$\Pr\left(\|X\|_{op} > C'K\left(\sqrt{2n} + t\right)\right) \leq e^{-t^2}, \tag{A.31}$$

where K is defined the same way as in the lemma.

Let $\tilde{C} = 2^{1/p-1}C'$. We get:

$$\begin{aligned}\mathbb{E}\left(\|X\|_{op}^p\right) = \int_0^\infty \Pr\left(\|X\|_{op}^p > t\right) dt = \int_0^\infty \Pr\left(\|X\|_{op} > t^{1/p}\right) dt \\ \leq \tilde{C}^p K^p (2n)^{p/2} + \int_{\tilde{C}^p K^p (2n)^{p/2}}^\infty \Pr\left(\|X\|_{op} > t^{1/p}\right) dt. \end{aligned} \tag{A.32}$$

Let us bound the second integral. By concavity of the root, $(a+b)^{1/p} = 2^{1/p}((a+b)/2)^{1/p} \geq 2^{1/p-1}(a^{1/p} + b^{1/p})$. Therefore

$$\begin{aligned}\int_{C'^{,p} K^p (2n)^{p/2}}^\infty \Pr\left(\|X\|_{op} > t^{1/p}\right) dt \leq \int_0^\infty \Pr\left(\|X\|_{op} > (t + \tilde{C}^p K^p (2n)^{p/2})^{1/p}\right) dt \\ \leq \int_0^\infty \Pr\left(\|X\|_{op} > 2^{1/p-1}(t^{1/p} + \tilde{C}K\sqrt{2n})\right) dt \\ = 2^{p-1}(C'K)^p \int_0^\infty \Pr\left(\|X\|_{op} > C'K(u^{1/p} + \sqrt{2n})\right) du \leq 2^{p-1}(C'K)^p \int_0^\infty e^{-u^{2/p}}\, du. \end{aligned} \tag{A.33}$$

The integral converges for any $p \in [1, \infty)$ and does not depend on n. Therefore $\mathbb{E}\left(\|X\|_{op}^p\right) \leq \tilde{C}^p K^p (2n)^{p/2} + \bar{C}K^p$ for some new constant $\bar{C}$. Taking $C = \tilde{C}^p 2^{p/2} + \bar{C}$ gives the result. □

Therefore

$$\limsup_{n\to\infty} \mathbb{E}\,|c^i(1) - c^i(0)|^p \leq 2C\delta^p. \tag{A.34}$$

Taking infimum over $\delta > 0$ gives simply $\lim_{n\to\infty} \mathbb{E}\,|c^i(1) - c^i(0)|^p = 0$, which means that $c^i(1) - c^i(0)$ converges to zero in L^p.

Since all of our nonlinearities are Lipschitz, they are linearly bounded. For such nonlinearities, Gaussian Master Theorem guarantees also convergence in L^p for any $p \in [1, \infty)$, by a straightforward extension of Theorem A.2 of Yang (2020b).[3] Therefore

$$\limsup_{n\to\infty} \mathbb{E}\,|c^i(1) - \mathring{c}^i|^p \leq 2^p \limsup_{n\to\infty} \mathbb{E}\,|c^i(1) - c^i(0)|^p + 2^p \limsup_{n\to\infty} \mathbb{E}\,|c^i(0) - \mathring{c}^i|^p = 0, \tag{A.35}$$

[3] Suppose we are in the Gaussian case, Setup 3.3.3. As shown in the proof of Theorem A.2 of Yang (2020b), when all the nonlinearities are linearly bounded, each vector g^j for $j \in [1, M]$ has bounded l_2 norm with high probability. From this follows that c^i is bounded with high probability, and so is $(c^i)^p$ for any fixed $p \in [1, \infty)$. This precisely means that there exist $C, c > 0$ such that $\Pr((c^i)^p > r) \leq Ce^{-crn}$ for any $r > C$. Decompose $(c^i)^p$ as $(c^i)^p = (c^i)^p \mathbb{I}((c^i)^p \leq C) + (c^i)^p \mathbb{I}((c^i)^p > C)$. The first term converges by Gaussian Master Theorem and dominated convergence, while the second one converges in mean since $\mathbb{E}\left((c^i)^p \mathbb{I}((c^i)^p > C)\right) \leq \int_C^\infty Ce^{-crn}\, dr = \frac{C}{cn} e^{-cCn} \to 0$ as $n \to \infty$.

which means that $c^i(1)$ converges to $\mathring{c}^i$ in L^p.

□

A.4 Tensor Program Formulation Equivalence

The Tensor Program series Yang (2019b, 2020a,b); Yang and Littwin (2021); Yang and Hu (2021); Yang et al. (2021) defines a Tensor Program as a pair of initial state and a sequence of commands. The initial state consists of variables of three different types: A, G, and C, which correspond to matrices, vectors, and scalars in Eq. (3.4), respectively. In its body, the program can also generate X-vars (that correspond to a vector x in Eq. (3.4)), which are size-n vectors but with a different meaning compared to G-vars. The G- and C-vars in the initial state are called input variables.

Each command takes some variables from the state, generates a new variable, and appends it to the state. In the most general version of a Tensor Program, NETSOR$\top^+$, the following commands are available:

- Trsp. Input: $A : \mathsf{A}$. Output $A^\top : \mathsf{A}$.
- MatMul. Input: $A : \mathsf{A}$, $x : \mathsf{X}$. Output: $Ax : \mathsf{G}$.
- Nonlin$^+$. Input: $g^1 : \mathsf{G}, \ldots, g^k : \mathsf{G}$, $c^1 : \mathsf{C}, \ldots, c^l : \mathsf{C}$. Output: $x : \mathsf{X}$, where $x_\alpha = \phi(g^1_\alpha, \ldots, g^k_\alpha, c^1, \ldots, c^l)$ $\forall \alpha \in [n]$.
- Moment. Input: $g^1 : \mathsf{G}, \ldots, g^k : \mathsf{G}$, $c^1 : \mathsf{C}, \ldots, c^l : \mathsf{C}$. Output: $c : \mathsf{C}$, where $c = \frac{1}{n}\sum_{\alpha=1}^n \phi(g^1_\alpha, \ldots, g^k_\alpha, c^1, \ldots, c^l)$.

It is easy to see that the iteration of Eq. (3.4) can be expressed as a NETSOR$\top^+$ program above. Note that in all Master Theorems, we care only about vectors of type G and scalars. Let us now show that for any NETSOR$\top^+$ program, we can construct an iteration in the form of Eq. (3.4) that generates the same set of G- and C-vars.

Note that G-vars can only be generated by MatMul. Each such MatMul uses a single X-var which can be generated only by Nonlin$^+$. For the m-th G-var, we therefore get the following iteration:

$$g^m_\alpha = \sum_{\beta=1}^n W^m_{\alpha\beta}\phi^m(g^1_\beta, \ldots, g^{k_m}_\beta; c^1, \ldots, c^{l_m}),$$

where W^m can be any A-var (or its transpose). By generating "placeholder" G- or C-vars (say, zeros), we can assume WLOG that $k_m = l_m = m - 1$. Moreover, we can generate a G-var and a C-var at the same time using the same function ϕ^m as a subsequent nonlinearity may not depend on one of them if not necessary. This gives us an iteration of the form of Eq. (3.4).

A.5 Comparison with Chen and Lam (2021)

The work of Chen and Lam (2021) considers the following iteration:

$$\tilde{x}^2_\alpha = \phi^2\left(\sum_\delta \tilde{A}_{\alpha\beta}x^1_\beta\right), \quad \tilde{x}^m_\alpha = \phi^m\left(\sum_\delta \tilde{A}_{\alpha\beta}\tilde{x}^{m-1}_\beta, \tilde{x}^{m-2}_\alpha, \ldots, \tilde{x}^2_\alpha, x^1_\alpha\right) \quad \forall m > 2\ \forall \alpha \in [n], \tag{A.36}$$

where $\tilde{A}$ is a sum of two **symmetric** $n \times n$ matrices: one has iid sub-Gaussian entries with zero mean and variance n^{-1}, while the other is a deterministic matrix divided by n. Denote the first matrix as $\tilde{Z}$, and the second as X/n, so $\tilde{A} = \tilde{Z} + X/n$. Let $A = Z + X/n$, where Z is a symmetric iid Gaussian matrix with zero mean and variance n^{-1}. Consider the corresponding iteration:

$$x_\alpha^2 = \phi^2\left(\sum_\delta A_{\alpha\beta} x_\beta^1\right), \quad x_\alpha^m = \phi^m\left(\sum_\delta A_{\alpha\beta} x_\beta^{m-1}, x_\alpha^{m-2}, \dots, x_\alpha^2, x_\alpha^1\right) \quad \forall m > 2\ \forall \alpha \in [n], \tag{A.37}$$

The main result of Chen and Lam (2021) follows:

Theorem A.5.1 (Chen and Lam (2021)). *Take $M \geq 2$ and a function ψ with M arguments. Suppose ψ and all ϕ^m for $m \in [2, M]$ are Lipschitz. Then*

$$\lim_{n\to\infty} \left| \frac{1}{n} \sum_{\alpha=1}^{n} \left(\psi(x_\alpha^1, \dots, x_\alpha^M) - \psi(\tilde{x}_\alpha^1, \dots, \tilde{x}_\alpha^M)\right)\right| = 0 \tag{A.38}$$

in probability.

In this formulation, $\frac{1}{n}\sum_{\alpha=1}^n \psi(x_\alpha^1, \dots, x_\alpha^M)$ does not converge to $\mathbb{E}_{z\sim\mathcal{N}(\mu,\Sigma)}\psi(z)$ for some μ and Σ since $x_\alpha^1, \dots, x_\alpha^M$ are images of nonlinear functions. Therefore the above formulation does not allow for a clear Gaussian process interpretation as in our Corollary 3.4.3 or Corollary A.3.3.

One can show equivalence of our iteration (3.4) and the above iteration in a certain scenario. Namely for the above, let X be and identity matrix and assume x^1 is an elementwise image of a Gaussian vector. For our iteration (3.4), let $M_0 = 1$, let us generate a "twin" variable for each vector, where the only difference is that one uses A^m, while the other uses $A^{m,\top}$, and let each nonlinearity use a sum of x_α^j, $\bar{x}_\alpha^j$, and c^j for each $j \in [m-1]$, where $\bar{x}^j$ is a twin variable for x^j. Equivalence is then showed by reorganizing nonlinearities of the two variants.

Apart from equivalence in the above scenario, the use of iteration (A.36) for expressing deep learning computations is very limited. First, iteration (A.36) assumes weight matrices to be symmetric, while such weight initializations are rarely used in practice. Second, iteration (A.36) applies the same weight matrix each time. While some layers in neural nets can indeed share weights, assuming that all layers share weights strictly limits us to vanilla recurrent neural nets, drawing out even simplest feedforward nets. Third, the form $A = Z + X/n$ has no clear interpretation in typical neural network computations. And last, iteration (A.36) assumes only one input variable. In terms of neural network computations, this limits us to a single input, no bias vectors, and no output layer. It automatically draws out the ability to express a backward pass (and therefore learning process). Moreover, we cannot hope even to show Gaussian process behavior (as in Corollary 3.4.3) as it requires evaluating the network on a batch of inputs of any finite size.

While our proof is based on similar weight interpolation idea as the proof of Theorem A.5.1, certain crucial details are different. Obviously, the exact interpolation function is different: we use a trigonometric interpolation scheme (Eq. (3.7)) while they use a root-linear scheme ($\tilde{a}\sqrt{t} + a\sqrt{1-t}$). While Theorem A.5.1 requires everything to be Lipschitz, our Theorem 3.3.7 allows for certain smooth polynomially bounded nonlinearities and ψ-functions. While restricting oneself to Lipschitz nonlinearities still allows to express a forward pass of a, say, ReLU net, it does not allow for expressing its backward pass since the derivative of a ReLU is not Lipschitz (and not even continuous). Moreover, computing a kernel (NNGP or NTK, see Corollary 3.4.3 and Corollary 3.4.4) requires a quadratic ψ-function. Finally, they prove only convergence in probability, while our Theorem 3.3.7 states almost sure convergence and convergence in L^p for any p, which is much stronger.

A.6 Proof of Lemma 3.5.3

We prove Lemma 3.5.3, recalled here for convenience.

Lemma 3.5.3 (Interpolation Properties)**.** *For any matrix entry $a(t) = A^i_{\alpha\beta}(t)$ of a program in Setup 3.3.6:*

1. $\mathbb{E}\,\dot{a}(t) = \mathbb{E}\,a(t)\dot{a}(t) = 0$ *for all* t
2. *For any integers $j, k \geq 0$ with sum $\ell = j + k$,*

$$\sup_t \mathbb{E}\,|a(t)^j \dot{a}(t)^k| \leq \pi^\ell \nu_\ell n^{-\ell/2},$$

where ν_ℓ is the scaled moment bound in Setup 3.3.6.

Beyond our TP setting, our proof shows that these two properties hold for any interpolation $a(t) = a_1 \cos\frac{\pi}{2}t + a_2 \sin\frac{\pi}{2}t$ between centered random variables a_1 and a_2 with identical variance $1/n$ and satisfying the scaled moment bound $\mathbb{E}\,|a_1|^k, \mathbb{E}\,|a_2|^k \leq \nu_k / n^{-k/2}$ for all $k \geq 3$.

Proof. Write

$$a(t) = \tilde{a}\cos\frac{\pi}{2}t + a\sin\frac{\pi}{2}t$$

where $\tilde{a}$ is the Gaussian random variable and a is the non-Gaussian random variable. Then

$$\dot{a}(t) = \frac{\pi}{2}(-\tilde{a}\sin\frac{\pi}{2}t + a\cos\frac{\pi}{2}t).$$

Then $\mathbb{E}\,\dot{a}(t) = 0$ follows from the zero-mean property of $\tilde{a}$ and a. Likewise, straightforward calculation using the independence between $\tilde{a}$ and a shows $\mathbb{E}\,a(t)\dot{a}(t) = 0$.

To show Item 2, first note that, for any t,

$$|a(t)|, \frac{2}{\pi}|\dot{a}(t)| \leq |\tilde{a}| + |a|.$$

Then

$$\mathbb{E}\,|a(t)^j \dot{a}(t)^k| \leq \left(\frac{\pi}{2}\right)^k \sum_{i=0}^{k} \binom{\ell}{i} \mathbb{E}\,|\tilde{a}|^i |a|^{\ell-i}.$$

By Hölder's inequality,

$$\mathbb{E}\,|\tilde{a}|^i |a|^{\ell-i} \leq (\mathbb{E}\,|\tilde{a}|^\ell)^{\frac{i}{\ell}} (\mathbb{E}\,|a|^\ell)^{\frac{\ell-i}{\ell}} \leq \nu_\ell n^{-\ell/2}.$$

Therefore,

$$\mathbb{E}\,|a(t)^j \dot{a}(t)^k| \leq \left(\frac{\pi}{2}\right)^k \sum_{i=0}^{k} \binom{\ell}{i} \nu_\ell n^{-\ell/2} = \left(\frac{\pi}{2}\right)^k 2^\ell \nu_\ell n^{-\ell/2} = \pi^\ell \nu_\ell n^{-\ell/2}.$$

□

A.7 Technical Preliminaries

A.7.1 L^p Norm

For any vector $v \in \mathbb{R}^k$ and $p \geq 1$, we write $\|v\|_p$ to denote its L^p norm $\|v\|_p \stackrel{\text{def}}{=} \sqrt[p]{|v_1|^p + \cdots + |v_k|^p}$. When $p = 2$, we will just write $\|v\| = \|v\|_2$ when there's no cause for confusion. The following L^p norm bound is standard.

Lemma A.7.1. *For any vector $b \in \mathbb{R}^k$, if $p \leq q$, then*

$$\|b\|_q \leq \|b\|_p \leq k^{\frac{1}{p}-\frac{1}{q}} \|b\|_q$$

We will use the following trivial but useful fact repeatedly. It follows trivially from Lemma A.7.1.

Lemma A.7.2. *For any integer $m \geq 0$ and reals $a_i \in \mathbb{R}$, $i \in [k]$,*

$$\left| \sum_{i=1}^k a_i \right|^m \leq k^{m-1} \sum_{i=1}^k |a_i|^m .$$

A.7.2 Multisets

Definition A.7.3 (Multiset)**.** A *multiset* is a set allowing multiple occurences of the same element, e.g., $\{1, 1, 1, 2, 2, 3\}$ (which is not equal to $\{1, 2, 3\}$ as a multiset). We will use capital italic font to denote multisets, such as $\mathcal{P}$. Thus, e.g., when we write $\sum_{p \in \mathcal{P}} f(p)$, p could take the same value multiple times. The number of elements in $\mathcal{P}$, counting multiplicity, is denoted $|\mathcal{P}|$. The set of *unique* elements is denoted $\text{uniq}(\mathcal{P})$.

Definition A.7.4 (Partition of multiset)**.** A *partition* τ of a multiset $\mathcal{P}$ expresses $\mathcal{P}$ as a disjoint union of multisets. Concretely, τ is a multiset $\{\mathcal{P}_1, \ldots, \mathcal{P}_k\}$ such that $\mathcal{P} = \bigsqcup_i \mathcal{P}_i$, and $|\tau| = k$ is the number of sets in the partition. For example, $\mathcal{P} = \{1, 1, 1, 2, 2, 3\} = \{1, 1\} \sqcup \{2, 2\} \sqcup \{1, 3\}$, so that $\tau = \{\{1, 1\}, \{2, 2\}, \{1, 3\}\}$ is a partition of $\mathcal{P}$. As another example, $\mathcal{P} = \{1\} \sqcup \{1\} \sqcup \{1\} \sqcup \{2\} \sqcup \{2\} \sqcup \{3\}$ is also a partition, with repeated sets, which emphasizes the fact that τ is allowed to be a multiset.

Obviously, the typical notion of a partition of a *set* is just a special case of the above concept applied to sets.

Definition A.7.5 (Partition induced by multiset)**.** Given a multiset $\mathcal{P}$, we can count the multiplicity of each element in $\mathcal{P}$ and list them in nonincreasing order, say $\pi_1 \geq \pi_2 \geq \cdots \geq \pi_k \geq 1$ where $\sum_{i=1}^k \pi_i = |\mathcal{P}|$. Then the multiset $\pi = \{\pi_i\}_{i=1}^k$ form a *partition* $\pi(\mathcal{P})$ of the integer $|\mathcal{P}|$, which we call the *partition induced by* $\mathcal{P}$, and we denote $|\pi| \stackrel{\text{def}}{=} k$, the *size of the partition*. Again, note that $\pi(\mathcal{P})$ does not contain any information about the identity of elements in $\mathcal{P}$, only their counts.

For example, $\mathcal{P} = \{\bullet, \bullet, \bullet, \star, \star, \times\}$ would induce the partition $|\mathcal{P}| = 6 = 3 + 2 + 1$ because 3 is the multiplicity of $\bullet$, 2 is the multiplicity of $\star$, and 1 is the multiplicity of $\times$.

Note that the partitions of integer n are equivalent to the partitions of the multiset $\underbrace{\{1, \ldots, 1\}}_{n}$.

Definition A.7.6 (Multiset from vector)**.** A multiset can be obtained from a vector by forgetting the order of the vector's entries, e.g., $\{1,1,1,2,2,3\}$ can obtained this way from the vector $(1,2,1,3,2,1)$ or $(2,3,1,1,2,1)$. We will use capital bold font to denote such vectors in the context of multisets, such as $\mathbf{P} = (P_1, P_2, \ldots)$, and such obtained multiset would be written as $\tilde{\mathbf{P}} = \{P_1, P_2, \ldots\}$.

Definition A.7.7 ($N_{\mathcal{P}}$)**.** For any fixed multiset $\mathcal{P}$, we can ask how many distinct vectors $\mathbf{P}$ are there such that $\mathcal{P} = \tilde{\mathbf{P}}$. The answer is the multinomial coefficient $N_{\mathcal{P}} \stackrel{\text{def}}{=} \binom{|\mathcal{P}|}{\pi} = \binom{|\mathcal{P}|}{\pi_1;\cdots;\pi_k}$, where $\pi = \pi(\mathcal{P})$ is the partition induced by $\mathcal{P}$. Note that $N_{\mathcal{P}}$ depends on $\mathcal{P}$ only through π.

For instance, for $\mathcal{P} = \{\bullet, \bullet, \bullet, \star, \star, \times\}$, we have $N_{\mathcal{P}} = \binom{6}{3;2;1}$.

A.7.3 Monomials

Given a vector $\boldsymbol{x} = (x_1, \ldots, x_k) \in \mathbb{R}^k$ and a vector $\mathbf{P} \in [k]^r$ for some integer $r \geq 0$, we can form the monomial $\boldsymbol{x}^{\mathbf{P}} \stackrel{\text{def}}{=} \prod_{i=1}^{r} x_{P_i}$. Thus, if 1 appears in $\mathbf{P}$ t times, then $\boldsymbol{x}^{\mathbf{P}}$ contains a factor of x_1^t.

If we also have a multiset $\mathcal{P}$ taking values in $[k]$, then we can also form the monomial $\boldsymbol{x}^{\mathcal{P}} \stackrel{\text{def}}{=} \prod_{\alpha \in \mathcal{P}} x_\alpha$. In this notation, we would have $\boldsymbol{x}^{\mathbf{P}} = \boldsymbol{x}^{\tilde{\mathbf{P}}}$.

A.7.4 Tensors

Given vectors $x \in \mathbb{R}^k, y \in \mathbb{R}^l$, their tensor $x \otimes y$ is the vector in $\mathbb{R}^{k \times l}$ with entries $(x \otimes y)_{(i,j)} = x_i y_j$, where the index ranges over $(i,j) \in [k] \times [l]$. We will use angle brackets $\langle -, - \rangle$ to denote standard inner product on all Euclidean spaces, so that

$$\langle x \otimes y, x' \otimes y' \rangle = \langle x, x' \rangle \langle y, y' \rangle \tag{A.39}$$

if $x, x' \in \mathbb{R}^k, y, y' \in \mathbb{R}^l$.

We can tensor a vector $x \in \mathbb{R}^k$ with itself $p-1$ times to form a *tensorial power* $x^{\otimes p} \in \mathbb{R}^{k^p}$ of x. Its entries would be indexed by elements $\mathbf{P}$ of $[k]^p$, with $(x^{\otimes p})_{\mathbf{P}} = x^{\mathbf{P}}$. Like in Eq. (A.39), we have

$$\langle x^{\otimes p}, y^{\otimes p} \rangle = \langle x, y \rangle^p.$$

for any x, y in the same Euclidean space.

This is particularly relevant as we will often encounter the expression $(\sum_\alpha x_\alpha)^p$ which can be written as

$$\left(\sum_\alpha x_\alpha \right)^p = \langle 1, x \rangle^p = \langle 1^{\otimes p}, x^{\otimes p} \rangle = \sum_{\mathbf{P} \in [k]^p} x^{\mathbf{P}}. \tag{A.40}$$

By Definition A.7.7, we can further rewrite this as

$$\left(\sum_\alpha x_\alpha \right)^p = \sum_{\mathcal{P}} N_{\mathcal{P}} x^{\mathcal{P}} \tag{A.41}$$

where $\mathcal{P}$ ranges over all multisets of size $|\mathcal{P}| = p$ taking value in $[k]$.

Tensorial Cauchy-Schwarz and Holder's Inequalities Below, we will always use $\|x\|_r$ to denote the L^r norm of the tensor x as a vector (and in particular will never speak of the "operator norm" of x), and $\|x\|$ denotes the L^2 norm as usual. It's easy to see the following identity holds for any vectors x, y and any $r \geq 1$:

$$\|x \otimes y\|_r = \|x\|_r \|y\|_r. \tag{A.42}$$

We therefore have the tensorial versions of Cauchy-Schwarz and Holder's Inequalities, for example,

$$\langle x \otimes y, x' \otimes y' \rangle \leq \|x\|_p \|y\|_p \|x'\|_q \|y'\|_q \tag{A.43}$$

for any $p^{-1} + q^{-1} = 1$.

A.7.5 Higher Order Differentiation and Taylor Expansion

Scalar Function For a smooth multivariate scalar function $f(x) = f(x_1, \ldots, x_k)$ and a vector $\mathbf{P} \in [k]^r$ with multiset $\mathcal{P} \stackrel{\text{def}}{=} \tilde{\mathbf{P}}$, we write

$$\partial_x^{\mathbf{P}} f(x) = \partial_x^{\mathcal{P}} f(x) \stackrel{\text{def}}{=} \frac{\partial^r}{\partial x_{P_1} \cdots \partial x_{P_r}} f(x).$$

We also define $\nabla_x^p f(x) \in \mathbb{R}^{k^p}$ as the pth order tensor containing all order-p derivatives of f evaluated at x:

$$(\nabla_x^p f(x))_{\mathbf{P}} = \partial_x^{\mathbf{P}} f(x).$$

This is a highly symmetric tensor as $(\nabla_x^p f(x))_{\mathbf{P}} = (\nabla_x^p f(x))_{\mathbf{Q}}$ if $\tilde{\mathbf{P}} = \tilde{\mathbf{Q}}$ as multisets. In all notations, we will drop the subscript, e.g., $\partial^{\mathbf{P}} = \partial^{\mathbf{P}}$, when the variables of the differentiation are clear from context.

Note that we have used a superscript notation for $\partial^{\mathbf{P}}$ and $\partial^{\mathcal{P}}$ to emphasize that they denote a "product of (1st order) partial derivatives operators." In particular, $\partial^{\mathbf{P}} f(x)$ and $\partial^{\mathcal{P}} f(x)$ should be distinguished from the vector of 1st order partial derivatives $(\frac{\partial}{\partial x_{P_i}} f(x))_{i=1}^r$. They will always be scalar quantities when f is scalar, whereas we always use ∇^p for when we want to think of the collection of (higher order) partial derivatives as a vector or tensor.

In this notation, f's Taylor expansion around $x = 0$ can be written as

$$f(0) + \langle \nabla f(0), x \rangle + \frac{1}{2!} \langle \nabla^2 f(0), x^{\otimes 2} \rangle + \cdots + \frac{1}{p!} \langle \nabla^p f(0), x^{\otimes p} \rangle + \cdots .$$

Of course, in general this Taylor expansion is not exact unless f is analytic in the appropriate neighborhood. For our purpose, the Taylor expansion with remainder is more useful, as it is exact:

Lemma A.7.8 (Taylor expansion with remainder)**.** *If $f : \mathbb{R}^k \to \mathbb{R}$, $f(x) = f(x_1, \ldots, x_k)$ is C^{p+1} (i.e., $\nabla^p f(x)$ has continuous partial derivatives), then*

$$f(x) = f(0) + \langle \nabla f(0), x \rangle + \cdots + \frac{1}{p!} \langle \nabla^p f(0), x^{\otimes p} \rangle + R_{p+1},$$

where the remainder R_{p+1} is

$$R_{p+1} \stackrel{def}{=} \frac{1}{p!} \int_0^1 (1-t)^p \langle \nabla^{p+1} f(tx), x^{\otimes (p+1)} \rangle dt$$

Vector Function For the vector case $f : \mathbb{R}^k \to \mathbb{R}^l, f(x) = (f_1(x), \dots, f_l(x))$, all of the above considerations apply to the components of f in parallel. For example,

$$\partial^{\mathbf{P}} f(x) = \partial^{\mathcal{P}} f(x) \stackrel{\text{def}}{=} (\partial^{\mathcal{P}} f_1(x), \dots, \partial^{\mathcal{P}} f_l(x)) \in \mathbb{R}^l$$
$$\nabla^p f(x) \stackrel{\text{def}}{=} (\nabla^p f_1(x), \dots, \nabla^p f_l(x)) \in \mathbb{R}^{l \times k^p}$$

A.7.6 Higher Order Chain Rule

Lemma A.7.9 (Hardy (2006))**.** *Suppose $f : \mathbb{R}^l \to \mathbb{R}, y : \mathbb{R}^k \to \mathbb{R}^l$ are C^k and consider their composition $f(y) = f(y(x_1, \dots, x_k))$. Then for any vector $\mathbf{P} \in [k]^r$,*

$$\partial_x^{\mathbf{P}} f(y) = \sum_{\tau} \langle \nabla_y^{|\tau|} f(y), \bigotimes_{\mathcal{P} \in \mathbf{P}[\tau]} \partial_x^{\mathcal{P}} y \rangle \tag{A.44}$$

where τ ranges over partitions of $\{1, \dots, r\}$ and $\mathbf{P}[\tau]$ is the partition $\{\{P_i : i \in \mathcal{S}\} : \mathcal{S} \in \tau\}$ of $\tilde{\mathbf{P}}$.

Eq. (A.44) may look somewhat scary at first, but when we apply it, the range of τ (in the outer sum) and the range of $\mathcal{P}$ (in the inner tensor product) are both $O(1)$-sized, as $n \to \infty$. So when we want to bound $\partial_x^{\mathbf{P}} f(y)$ (which is the only time we will use Eq. (A.44)), what we end up caring about is only the individual norms of $\nabla_y^{|\tau|} f(y)$ and $\partial_x^{\mathcal{P}} y$. May this fact assuage the reader intimidated by the density of Eq. (A.44).

Nevertheless, let us take some time to digest Eq. (A.44).

To unpack the tensor notation, notice that 1) $\nabla_y^{|\tau|} f(y) \in \mathbb{R}^{l^{|\tau|}}$, 2) $\partial_x^{\mathcal{P}} y = (\partial_x^{\mathcal{P}} y_1, \dots, \partial_x^{\mathcal{P}} y_l) \in \mathbb{R}^l$, so 3) $\bigotimes_{\mathcal{P} \in \mathbf{P}[\tau]} \partial_x^{\mathcal{P}} y$, being a tensor over $|\mathbf{P}[\tau]| = |\tau|$ vectors, is in $\mathbb{R}^{l^{|\tau|}}$ as well. So, for example, when $|\tau| = 1$, there is only one partition, consisting of the whole of $[k]$, and $\mathbf{P}[\tau] = \{\tilde{\mathbf{P}}\}$, so that the corresponding term in Eq. (A.44) is

$$\langle \nabla_y f(y), \partial_x^{\mathbf{P}} y \rangle = \sum_{j=1}^{l} \frac{\partial f}{\partial y_j} \frac{\partial^r y_j}{\partial x^{\mathbf{P}}}.$$

As a helping example, consider the case when $r = 2$ and $\mathbf{P} = (1, 2)$. Then we have

$$\partial_x^{\mathbf{P}} f(y) = \partial_{x_1} \partial_{x_2} f(y) = \partial_{x_1} \left(\langle \nabla_y f(y), \partial_{x_2} y \rangle \right) = \langle \nabla_y f(y), \partial_{x_1} \partial_{x_2} y \rangle + \langle \nabla_y^2 f(y), \partial_{x_1} y \otimes \partial_{x_2} y \rangle.$$

Here the first term corresponds to $\tau = \{[k]\}$; in this case, $\mathbf{P}[\tau] = \{\tilde{\mathbf{P}}\} = \{\{x_1, x_2\}\}$. The second term corresponds to $\tau = \{\{1\}, \{2\}\}$; in that case, $\mathbf{P}[\tau] = \{\{x_1\}, \{x_2\}\}$.

Note that Eq. (A.44) is not equivalent to the look-alike equation where we allow τ to range over partitions of the multiset $\tilde{\mathbf{P}}$ and letting $\mathcal{P}$ in the inner tensor product range over $\mathcal{P} \in \tau$. This alternative equation would lead to too few terms in the sum over τ, as is apparent when one considers $\mathbf{P} = (1, \dots, 1)$.

Applying Eq. (A.42) to Eq. (A.44), we get

Lemma A.7.10. *In the same setting as in Lemma A.7.9, we have*

$$|\partial_x^{\mathbf{P}} f(y)| \le \sum_{\tau} \|\nabla_y^{|\tau|} f(y)\| \cdot \prod_{\mathcal{P} \in \mathbf{P}[\tau]} \|\partial_x^{\mathcal{P}} y\|. \tag{A.45}$$

We will in particular need to apply this repeatedly to the case when $f(y) = \prod_j y_j$:

Lemma A.7.11. *Suppose $y : \mathbb{R}^k \to \mathbb{R}^l$ is C^k, with $y(x)$ denoting the value of y on input x. Then for any vector $\mathbf{P} \in [k]^r$,*

$$\left\| \partial_x^{\mathbf{P}} \prod_j y_j \right\| \le \sum_{t=0}^{l-1} \sqrt{l^{l-t-1} \|y\|_{2t}^{2t}} \cdot \sum_{\tau} \prod_{\mathcal{P} \in \mathbf{P}[\tau]} \|\partial_x^{\mathcal{P}} y\|. \tag{A.46}$$

where τ ranges over partitions of $\{1, \dots, r\}$ of size $l - t$ and $\mathbf{P}[\tau]$ is as in Lemma A.7.9. Here $\|y\|_0^0$ is by convention defined to be l.

The proof below actually shows the l^{l-t-1} in Eq. (A.46) can be refined to the falling factorial $(l-1)\cdots(t+1)$ but we will not need this. Also note that there are no partitions of size $> r$, so that the summand vanishes when $t < l - r$.

Proof. For $f(y) = \prod_j y_j$, by Eq. (A.45), we just need to show

$$\|\nabla_y^{|\tau|} f(y)\|^2 \le l^{|\tau|-1} \|y\|_{2(l-|\tau|)}^{2(l-|\tau|)}.$$

Each nonzero entry of $\nabla_y^{|\tau|} f(y)$ has the form y^S for some size-$(l - |\tau|)$ subset (i.e., having no duplicates) S of $[l]$. By power-mean inequality, $|y^S|^2 \le \frac{1}{|S|} \sum_{j \in S} |y_j|^{2|S|}$. Taking sum over all possible S of size $(l - |\tau|)$, we have

$$\sum_S |y^S|^2 \le \binom{l}{|\tau|} \frac{1}{l} \sum_{j=1}^{l} |y_j|^{2(l-|\tau|)}$$

Finally, each y^S appears $|\tau|!$ times in $\nabla_y^{|\tau|} f(y)$, so

$$\|\nabla_y^{|\tau|} f(y)\|^2 = |\tau|! \sum_S |y^S|^2 \le l^{|\tau|-1} \sum_{j=1}^{l} |y_j|^{2(l-|\tau|)}$$

as desired. □

A.7.7 Smoothness Profile of A Function

Definition A.7.12 (Smoothness Profile)**.** Consider a smooth (C^∞) function $f : \mathbb{R}^k \to \mathbb{R}$. If there is a sequence $\mathcal{C}$ of positive reals $(C_1, p_1), (C_2, p_2), \dots$ such that for any $\mathbf{P} \in [k]^r$, we have $\left|\partial^{\mathbf{P}} f(x)\right| \le C_r(1 + |x_1|^{p_r} + \cdots + |x_k|^{p_r})$, then we say $\mathcal{C}$ is a *smoothness profile*, or just *profile* for short, of f.

Obviously, f is polynomially smooth (Definition 3.3.5) iff it has a profile.

As one could guess, the "expected smoothness" of a program's vectors (under perturbation of its matrices) will depend on the nonlinearities ϕ^i of the program *only* through the profiles of ϕ^i. Thus, there are smoothness guarantees that are uniform over all programs whose nonlinearities have the same profiles.

A.8 The Basic Moment Argument

Throughout this work, we need to study many different sums of random variables $X = \sum_{\alpha=1}^{n} x_\alpha$. In particular, we will repeatedly use a basic argument to bound its moment $\mathbb{E}\, X^p$ for even p, which is formally codified in Lemma A.8.4 at the end of this section. But to motivate this eventual formulation, it's nice to start with a simple example.

Simple Motivating Example The simplest example would be when x_α are iid random variables that don't depend on n (but eventually we will need to cover the case of general non-iid distributions depending on n, with small correlations among x_α). Then for even p, by Eq. (A.41),

$$\mathbb{E}\, X^p = \sum_{\mathcal{P}} N_{\mathcal{P}} \mathbb{E}\, x^{\mathcal{P}} \tag{A.47}$$

where $\mathcal{P}$ ranges over all multisets of size $|\mathcal{P}| = p$ taking value in $[n]$ and $N_{\mathcal{P}}$ is the multinomial coefficient defined in Definition A.7.7. In our case, with each x_α being iid, it's clear that $\mathbb{E}\, x^{\mathcal{P}} = \mathbb{E}\, x^{\mathcal{P}'}$ if the multisets $\mathcal{P}, \mathcal{P}'$ have the samed induced partition: $\pi(\mathcal{P}) = \pi(\mathcal{P}')$. For example, $\mathbb{E}\, x_1 x_1 x_2 x_3 = \mathbb{E}\, x_1 x_1 x_3 x_4 = \mathbb{E}\, x_4 x_4 x_n x_{n-1}$ because the corresponding multisets all have the same induced partition $\{2, 1, 1\}$ of 4.

For any fixed partition τ of p, we can ask how many multisets $\mathcal{P}$ are there with $\pi(\mathcal{P}) = \pi$. The answer is quite straightforward but requires a bit of explanation notationally: we think of τ as another multiset and *take its induced partition* $\pi(\tau)$ of the integer $|\tau|$; then the answer is the multinomial coefficient $\binom{n}{\pi(\tau)}$. For the above example of $\tau = \{2, 1, 1\}$, we have $\pi(\tau) = \{1, 2\}$, because a) 1 in $\pi(\tau)$ is the multiplicity of 2 in τ and b) 2 in $\pi(\tau)$ is the multiplicity of 1 in τ. So there are $\binom{n}{1;2}$ multisets $\mathcal{P}$ with $\pi(\mathcal{P}) = \pi$.

Therefore, we can further rewrite Eq. (A.47) as a sum over partitions τ of integer p:

$$\mathbb{E}\, X^p = \sum_{\tau} \binom{n}{\pi(\tau)} N_{\mathcal{P}} \mathbb{E}\, x^{\mathcal{P}} \tag{A.48}$$

where for every τ, $\mathcal{P}$ is any choice of multiset with $\pi(\mathcal{P}) = \tau$. The advantage of this representation is that $\mathbb{E}\, X^p$ is now clearly a polynomial in n. This is because 1) τ ranges over a set that does not depend on n; 2) $N_{\mathcal{P}}$ does not depend on n; and 3) the distribution of x_α does not depend on n by our assumption. Thus, *as $n \to \infty$, we can obtain a bound on X^p by deriving the leading term of this polynomial.* Because $\binom{n}{\pi(\tau)} = \Theta(n^{|\tau|})$ (e.g., $\binom{n}{\pi(\{1,1,2\})} = \binom{n}{2;1} = \Theta(n^3)$), this means focusing on the the partitions τ with the largest $|\tau|$. In general, of course, this is just $\tau = \{1, \ldots, 1\}$ where $|\tau| = p$, corresponding to monomials like $x_1 x_2 \cdots x_p$. But in the cases we are concerned with, $\mathbb{E}\, x^{\mathcal{P}}$ for this τ will vanish, so we need to look at lower order terms.

For example, assume further that x_α has mean 0 and variance 1. Then $\mathbb{E}\, x^{\mathcal{P}} = 0$ for any $\mathcal{P}$ that contains an element appearing with multiplicity 1 (e.g., $\mathbb{E}\, x_1 x_2 x_2 = 0$ because it's equal to $(\mathbb{E}\, x_1)(\mathbb{E}\, x_2^2) = 0 \cdot (\mathbb{E}\, x_2^2) = 0$). Therefore, the leading term in the polynomial Eq. (A.47) corresponds exactly to the partition $\tau = \{2, \ldots, 2\}$ (where $|\tau| = p/2$) of p (corresponding to monomials like $\mathbb{E}\, x_1^2 \cdots x_{p/2}^2$ which is equal to 1 by our variance-1 assumption), and any other τ must either have smaller $|\tau|$ or it has $\mathbb{E}\, x^{\mathcal{P}} = 0$. Therefore, in this case, we deduce

$$\mathbb{E}\, X^p = \Theta(n^{p/2}), \quad \text{as } n \to \infty. \tag{A.49}$$

We can of course say more precise things about $\mathbb{E}\, X^p$ under these assumptions, which we record below.

Proposition A.8.1. *If $x_\alpha, \alpha \in [n]$, are sampled iid from a distribution with kth moment ν_k, then for any even integer p,*

$$\mathbb{E}\left(\sum_{\alpha=1}^{n} x_\alpha\right)^p = \sum_{\tau} \binom{n}{\pi(\tau)}\binom{p}{\tau}\nu^\tau \tag{A.50}$$

summing over partitions τ of integer p and any n.

Here $\nu^\tau = \prod_{k\in\tau} \nu_k$ following the multiset notation in Appendix A.7.3.
Slightly more generally, we have

Proposition A.8.2. *If $x_\alpha, \alpha \in [n]$, are sampled independently and the distribution of x_α has kth moment bounded above by ν_k, then for any even integer p,*

$$\mathbb{E}\left(\sum_{\alpha=1}^{n} x_\alpha\right)^p \le \sum_{\tau} \binom{n}{\pi(\tau)}\binom{p}{\tau}\nu^\tau \tag{A.51}$$

summing over partitions τ of integer p and any n.

General Case Now, when we consider x_α that are not iid and may correlate amongst themselves, Eq. (A.47) will continue to hold but in general Eq. (A.48) will not. Furthermore, $\mathbb{E}\, x^{\mathcal{P}}$ in the general case can also depend on n. So even if there's enough symmetry to obtain Eq. (A.48), we no longer have a polynomial expression of $\mathbb{E}\, X^p$ in n.

Nevertheless, most of the time, our objective is to show that $\mathbb{E}\, X^p$ is $O(1)$ as $n \to \infty$, and the above arguments can be easily adapted to work given the following condition:

Definition A.8.3 (Small Moment Condition)**.** Consider a sequence x of random vectors $(x(n) \in \mathbb{R}^n)_{n=1}^\infty$. We say x satisfies the *Small Moment Condition (SMC)* if for every even integer p, there exists a constant C such that

$$\mathbb{E}\, x(n)^{\mathcal{P}} \le C n^{-|\mathrm{uniq}(\mathcal{P})|} \tag{A.52}$$

for every multiset $\mathcal{P}$ of size p and for any n, where $\mathrm{uniq}(\mathcal{P})$ is the set of unique elements of $\mathcal{P}$.

Lemma A.8.4. *Consider a sequence x of random vectors $(x(n) \in \mathbb{R}^n)_{n=1}^\infty$. If x satisfies the Small Moment Condition (Definition A.8.3), then for any real $p \ge 1$,*

$$\mathbb{E}\left|\sum_{\alpha=1}^{n} x(n)_\alpha\right|^p = O(1) \quad \text{as } n \to \infty. \tag{A.53}$$

Here the constant in $O(1)$ depends only on p and the constant C in Definition A.8.3.

Proof. Note that, by the power-mean inequality, it suffices to prove this for even integer p. Let $X(n) \stackrel{\text{def}}{=} \sum_{\alpha=1}^n x(n)_\alpha$. Below we suppress the argument (n) notationally. Note that $|\mathrm{uniq}(\mathcal{P})| = |\pi(\mathcal{P})|$, for $\pi(\mathcal{P})$ defined in Definition A.7.5. By Eq. (A.47), we have, for every n,

$$\mathbb{E}\, X^p \le \sum_{\mathcal{P}} N_{\mathcal{P}} C n^{-|\pi(\mathcal{P})|} \le C' \sum_{\mathcal{P}} n^{-|\pi(\mathcal{P})|}$$

summing over multisets $\mathcal{P}$ of size p taking value in $[n]$. where C is the constant in Definition A.8.3 and $C' = C \max_{\mathcal{P}} N_{\mathcal{P}}$. Then the same symmetry argument leading to Eq. (A.48) yields

$$\mathbb{E}\, X^p \le C' \sum_{\tau} \binom{n}{\pi(\tau)} n^{-|\tau|} = \sum_{\tau} O(1) = O(1),$$

where τ ranges over all partitions of integer p. □

As an example, if $x_\alpha = x(n)_\alpha, \alpha \in [n]$, are sampled independently and the distribution of x_α has kth moment bounded above by $\nu_k / n^{k/2}$, where ν_k are independent of n, then $\mathbb{E}\, x^{\mathcal{P}} \le \nu^{\pi(\mathcal{P})} n^{-p/2}$ (where $\nu^{\pi(\mathcal{P})} = \prod_{k \in \pi(\mathcal{P})} \nu_k$). If each x_α further has mean 0, then as discussed above Eq. (A.49), $\mathbb{E}\, x^{\mathcal{P}} = 0$ if $|\pi(\mathcal{P})| > p/2$. For $\mathcal{P}$ with $|\pi(\mathcal{P})| \le p/2$, we have $\mathbb{E}\, x^{\mathcal{P}} \le \nu^{\pi(\mathcal{P})} n^{-p/2} \le C n^{-|\pi(\mathcal{P})|/2}$ where $C = \max_\tau \nu^\tau$ taken over all partitions τ of integer p. Thus SMC (Definition A.8.3) is satisfied, and Lemma A.8.4 applies to yield

Proposition A.8.5. *Consider a sequence x of random vectors $(x(n) \in \mathbb{R}^n)_{n=1}^\infty$. Suppose $x_\alpha = x(n)_\alpha, \alpha \in [n]$, are sampled independently. Assume there are $\nu_k, k = 1, 2, \ldots$, independent of n such that $\mathbb{E}\,|x_\alpha|^k \le \nu_k n^{-k/2}$ for all n and k. Then x satisfies Small Moment Condition (Definition A.8.3), and for any $p \ge 1$,*

$$\mathbb{E} \left| \sum_{\alpha=1}^{n} x(n)_\alpha \right|^p = O(1) \quad \text{as } n \to \infty. \tag{A.54}$$

Here the constant in $O(1)$ depends on p and $\nu_1, \ldots, \nu_p$ (i.e., the first p moment bounds only).

A.9 A Priori Smoothness and Moment Controls

Our main result in this section is Lemma A.9.3, which bounds the moments of derivatives of vector entries in the program. We shall come to it after stating some definitions.

Oblivious Constants In this and following sections, we need to reason carefully about constants hidden in big-O expressions. In particular, we isolate the following notion of *oblivious constant*, which roughly means that the constant does not depend on the fine details of a program beyond some finite number of smoothness and moment bounds.

For the first time reader, exactly understanding this notion is not a priority. So it may help to skip ahead to Lemma A.9.3, keeping in mind just this intuitive understanding of *oblivious constants*, and come back only after absorbing key ideas of our proofs.

Definition A.9.1. Consider a program T in Setup A.9.2, nonlinearities $\psi^1, \ldots, \psi^l$ for some $l \ge 0$, and multisets $\mathcal{P}_1, \ldots, \mathcal{P}_r$ for some $r \ge 0$. In the context of a bound or a big-O expression, we say a constant C is *$(\mathcal{P}_1, \ldots, \mathcal{P}_r)$-oblivious wrt T and $\psi^1, \ldots, \psi^l$* if all of the following hold.

1. C does not depend on n
2. C depends on each $\mathcal{P}_i$ only through its size $|\mathcal{P}_i|$
3. For each combination of $|\mathcal{P}_1|, \ldots, |\mathcal{P}_r|$, there is an integer $K > 1$ such that

(a) For any sequence $\nu_2, \nu_3, \ldots \geq 0$ such that $\mathbb{E}\,|a|^k \leq \nu_k n^{-k/2}$ for all matrix entries a of T, our constant C can be taken to depend on the distributions of matrix entries in T only through $\nu_2, \ldots, \nu_K$

(b) For any profile $((C_0, p_0), (C_1, p_1), \ldots)$ (Definition A.7.12) satisfied by all nonlinearities ϕ^i of the program T as well as $\psi^1, \ldots, \psi^l$, our constant C can be taken to depend on $\{\phi^i\}_i$ and $\{\psi^j\}_j$ only through $(C_0, p_0), \ldots, (C_K, p_K)$

(c) For any sequence $R_1, R_2, \ldots$ such that $\mathbb{E}\,|c^i|^q \leq R_q$ for all initial scalars c^i (i.e., where $i \in [M_0]$), all integers $q \geq 1$, and all n, our constant C can be taken to depend on $\{c^i\}_{i=1}^{M_0}$ only through $R_1, \ldots, R_K$.

(d) Furthermore, C is a oblivious function of $\nu_2, \ldots, \nu_K$, $(C_0, p_0), \ldots, (C_K, p_K)$, and $R_1, \ldots, R_K$.

We will just say $(\mathcal{P}_1, \ldots, \mathcal{P}_r)$-*oblivious* if the program and $\psi^1, \ldots, \psi^l$ are clear from context. Finally, any of the sets $\mathcal{P}_i$ can be an integer p, which just stands for the multiset $\underbrace{\{1, \ldots, 1\}}_{p}$.[4]

Thus, if an expression $\Lambda = \Lambda(\mathcal{P}, T, \psi)$ is bounded by a $\mathcal{P}$-oblivious constant wrt a program T and additional nonlinearity ψ, then

$$\sup_{\mathcal{P}, T, \psi} \Lambda \leq F(p, \mathcal{C}, (\nu_j)_j, (R_j)_j)$$

for some function F, where 1) $\mathcal{P}$ ranges over all multisets of size p, 2) ψ and all nonlinearities of T range over functions satisfying the profile $\mathcal{C}$, 3) the distributions of all matrix entries of T range of those that satisfy the moment bounds given by $(\nu_j)_j$, and 4) all initial scalars of T range over those that satisfy the moment bounds given by $(R_j)_j$.

Smoothness and Moment Control We relax Setup 3.3.6 slightly for our main result in this section.

Setup A.9.2. *Consider Setup 3.3.6, but relax 3*) to allow the variances of matrix entries to differ from n^{-1} but still bounded above by $\nu_2 n^{-1}$ for some $\nu_2 > 0$ common to all matrix entries.*

Lemma A.9.3 (Expected Smoothness of Vectors in a Program)**.** *Consider a Tensor Program under Setup A.9.2. Then, for any polynomially smooth $\psi : \mathbb{R}^M \to \mathbb{R}$, any $p \geq 1$, and any multiset $\mathcal{P}$ taking values in the program's matrix entries $\{A^i_{\alpha\beta}\}_{\alpha,\beta,i}$,*

$$\sup_{\alpha \in [n]} \mathbb{E}\,\left|\partial^{\mathcal{P}} \psi(g^1_\alpha, \ldots, g^M_\alpha; c^1, \ldots, c^M)\right|^p = O(1) \quad \text{as } n \to \infty. \tag{A.55}$$

where constant in the big-O is $(p, \mathcal{P})$-oblivious wrt ψ and the program.

We are slightly deviating from the partial derivative notation in Appendix A.7.5 as $\mathcal{P}$ now directly specify the variables to take derivatives against, rather than their indices.

One can interpret this result as saying that: any higher order derivative (with respect to weights) of any entry of $\psi(\cdots)$ will typically not explode to ∞ with n. The fact that the hidden constant is $(p, \mathcal{P})$-oblivious will be important when we prove our main result Theorem 3.5.2.

[4] In this case of $\mathcal{P}_i$ being an integer, condition 2 is then trivially satisfied, so the important condition is condition 3.

Remark A.9.4. As a sanity check, we discuss some features of this result before moving on to the proof. First, notice that the sup is outside the expectation. Were it the other way around, then we typically would expect some (function of) $\log n$ factors on the RHS of the bound.

Second, notice that if each matrix A^i has iid entries (instead of the more general case covered here where entries can come from different distributions), then by symmetry, the above expectation for all α would be identical, so the supremum is extraneous. But in general, this supremum is not extraneous.

Third, we assume the less stringent Setup A.9.2 instead of Setup 3.3.6 not just because we can but also because we need this in our inductive proof: we will need to reason about programs where some matrix entries are shrunken to 0, a condition that Setup A.9.2 captures but Setup 3.3.6 does not.

Remark A.9.5. This will hold for programs with variable dimensions, with the addendum that the constants $B_{p,|\mathcal{P}|}$ also can depend on hidden width ratios.

By applying power mean inequality to Lemma A.9.3, we also easily get

Lemma A.9.6. *Consider the same setting as Lemma A.9.3. Then*

$$\mathbb{E}\left|\partial^{\mathcal{P}}\frac{1}{n}\sum_{\alpha\in[n]}\psi(g_\alpha^1,\ldots,g_\alpha^M;c^1,\ldots,c^M)\right|^p = O(1) \quad \text{as } n\to\infty. \tag{A.56}$$

for the same hidden constant as in Eq. (A.55)*.*

However, later we will see that this bound is unnecessarily loose when $\mathcal{P}$ is not empty (Lemma A.10.6).

A.9.1 Proof of Lemma A.9.3: Induction Setup

In everything below, by *constant* we always mean something independent of n that may or may not depend on other data.

Fix the number of initial vectors M_0, as well as the sequence of scaled moment bounds $\nu_2, \nu_3, \ldots$, the profile $\mathcal{C}$, and the initial scalar moment bounds $R_1, R_2, \ldots$ as discussed in Setup A.9.2. We will prove the following claims simultaneously *for all programs that have M_0 initial vectors/scalars and satisfy the above constraints (specified by $(\nu_k)_k, \mathcal{C}, (R_k)_k$).* We do so by induction on the vector index j:

Claim 1(j) For any integer $p \geq 0$, there is a sequence of constants $B_{\mathsf{j},p,0}, B_{\mathsf{j},p,1}, \ldots$ such that

$$\sup_{\alpha\in[n]} \mathbb{E}\left|\partial^{\mathcal{P}} g_\alpha^{\mathsf{j}}\right|^p \leq B_{\mathsf{j},p,|\mathcal{P}|}$$

for any multiset $\mathcal{P}$ of the program's matrix entries. Furthermore, $B_{\mathsf{j},p,|\mathcal{P}|}$ is $(\mathsf{j}, p, \mathcal{P})$-oblivious.

Claim 2(j) For any polynomially smooth $\psi : \mathbb{R}^{2\mathsf{j}} \to \mathbb{R}$, any integer $p \geq 0$, there is a sequence of constants $B^\psi_{\mathsf{j},p,0}, B^\psi_{\mathsf{j},p,1}, \ldots$ such that

$$\sup_{\alpha\in[n]} \mathbb{E}\left|\partial^{\mathcal{P}}\psi(g_\alpha^1,\ldots,g_\alpha^{\mathsf{j}};c^1,\ldots,c^{\mathsf{j}})\right|^p \leq B^\psi_{\mathsf{j},p,|\mathcal{P}|}$$

for any multiset $\mathcal{P}$ of the program's matrix entries. Furthermore, $B^\psi_{\mathsf{j},p,|\mathcal{P}|}$ is $(\mathsf{j}, p, \mathcal{P})$-oblivious wrt to ψ and the program.

Note that $|\cdot|^0$ always equal 1 by convention in both claims. Of course, A.9.1(M) would yield Lemma A.9.3.

Obviously, A.9.1(j) is a special case of A.9.1(j), but our induction proof will go like this

$$\begin{gathered}\text{A.9.1(j}-1\text{)} \implies \text{A.9.1(j)} \\ \text{A.9.1}(1,\dots,\text{j}) \text{ and A.9.1}(1,\dots,\text{j}-1) \implies \text{A.9.1(j)}\end{gathered}$$

Before we begin the induction proof, we first record several consequences of the claims above.

Proposition A.9.7. *Recall ϕ^i denotes the ith nonlinearity of the program. A.9.1($i-1$) implies that*

$$\mathbb{E}\,|\partial^{\mathcal{P}} c^i|^p \le B^{\phi^i}_{i-1,p,|\mathcal{P}|}$$

for any multiset $\mathcal{P}$ of the program's matrix entries.

Proof. Unwinding the definition of c^i (Eq. (3.4)), we have

$$\begin{aligned}\mathbb{E}\,|\partial^{\mathcal{P}} c^i|^p &= \mathbb{E}\left|\frac{1}{n}\sum_{\alpha=1}^n \partial^{\mathcal{P}}\phi^i(g^1_\alpha,\dots,g^{i-1}_\alpha;c^1,\dots,c^{i-1})\right|^p \\ &\le \frac{1}{n}\sum_{\alpha=1}^n \mathbb{E}\left|\partial^{\mathcal{P}}\phi^i(g^1_\alpha,\dots,g^{i-1}_\alpha;c^1,\dots,c^{i-1})\right|^p && \text{applying Lemma A.7.2} \\ &\le \sup_{\alpha\in[n]} \mathbb{E}\left|\partial^{\mathcal{P}}\phi^i(g^1_\alpha,\dots,g^{i-1}_\alpha;c^1,\dots,c^{i-1})\right|^p \le B^{\phi^i}_{i,p,|\mathcal{P}|} && \text{applying A.9.1}(i-1).\end{aligned}$$

□

Proposition A.9.8. *Consider any polynomially smooth $\psi:\mathbb{R}^{2\text{j}}\to\mathbb{R}$ and any integers $p\ge 1$, $k\ge 0$. Recall that $\nabla^k\psi:\mathbb{R}^{2\text{j}}\to\mathbb{R}^{(2\text{j})^k}$ is the function that computes the tensor of ψ's kth-order partial derivatives. Then A.9.1$(1,\dots,\text{j})$ and A.9.1$(1,\dots,\text{j}-1)$ together imply the following L^p norm bound: There is a constant C, (j,p,k)-oblivious wrt ψ and the program, such that*

$$\sup_{\alpha\in[n]} \mathbb{E}\left\|\nabla^k\psi(g^1_\alpha,\dots,g^{\text{j}}_\alpha;c^1,\dots,c^{\text{j}})\right\|_p^p \le C$$

Note the form of this bound is very intuitive, since $\nabla^k\psi$ is just a polynomially bounded function, but taking values in a multi- but constant-dimensional space $\mathbb{R}^{(2\text{j})^k}$ instead of $\mathbb{R}$. The proof is just routine manipulation using Lemma A.7.2 and applications of A.9.1 and A.9.1.

Proof. Let (C,q) be the the kth element of ψ's profile, i.e., such that for all input vectors $v\in\mathbb{R}^{2\text{j}}$ and for all $\mathbf{U}\in[2\text{j}]^k$,

$$|\partial_v^{\mathbf{U}}\psi(v)| \le C(1+\|v\|_q^q). \tag{A.57}$$

Let $u \stackrel{\text{def}}{=} (g^1_\alpha,\dots,g^{\text{j}}_\alpha;c^1,\dots,c^{\text{j}})\in\mathbb{R}^{2\text{j}}$. Then

$$\|\nabla_u^k\psi(u)\|_p^p \le (2\text{j})^k \sup_{\mathbf{U}} |\partial_u^{\mathbf{U}}\psi(u)|^p$$

where $\mathbf{U}$ ranges over all vectors in $[2\text{j}]^k$. Thus it suffices to bound $\sup_{\alpha\in[n]}\mathbb{E}\sup_{\mathbf{U}}|\partial_u^{\mathbf{U}}\psi(u)|^p$ by a constant that depends on ψ only through C and q.

Applying Lemma A.7.1,

$$\mathbb{E}\,|\partial_u^{\mathbf{U}}\psi(u)|^p \le \mathbb{E}\,[C(1+\|u\|_q^q)]^p \le C'\mathbb{E}\,(1+\|u\|_{pq}^{pq})$$

where C' is a constant depending only on C, q, p, j continuously. Therefore it remains to show that $\mathbb{E}\,\|u\|_{pq}^{pq}$ is bounded by a constant independent of α. But, unwinding the definition of u,

$$\mathbb{E}\,\|u\|_{pq}^{pq} = \mathbb{E}\,|g_\alpha^1|^{pq} + \cdots + |g_\alpha^{\mathsf{j}}|^{pq} + |c^1|^{pq} + \cdots + |c^{\mathsf{j}}|^{pq}$$

By A.9.1$(1,\ldots,\mathsf{j})$,

$$\sup_{\alpha\in[n]} \mathbb{E}\,|g_\alpha^i|^{pq} \le B_{i,pq,0}$$

For $i \le \mathsf{j}$, by A.9.1$(i-1)$ and Proposition A.9.7, we have

$$\mathbb{E}\,|c^i|^{pq} \le B_{i-1,pq,0}^{\phi^i}$$

So $\mathbb{E}\,\|u\|_{pq}^{pq}$ is indeed bounded by a constant independent of α and (j, p, k)-oblivious. □

Proposition A.9.9. *Consider any integer $p \ge 0$, polynomially smooth $\psi : \mathbb{R}^{2\mathsf{j}} \to \mathbb{R}$ and any multisets $\mathcal{P},\mathcal{U}$ of matrix entries. Let $z_\alpha \overset{def}{=} \partial^{\mathcal{P}}\psi(g_\alpha^1,\ldots,g_\alpha^{\mathsf{j}};c^1,\ldots,c^{\mathsf{j}})$. Assume A.9.1(j). Then for any multiset $\mathcal{N}$ taking values in $[n]$,*

$$\mathbb{E}\,|\partial^{\mathcal{U}} z^{\mathcal{N}}|^p \le C$$

for some constant C that is $(\mathsf{j}, p, \mathcal{N}, \mathcal{P}, \mathcal{U})$-oblivious wrt ψ and the program.

The statement of this bound is a bit more complicated than the previous ones, but again the content is intuitive. It says that some interleaved composition of 1) taking a constant number of partial derivatives and 2) taking a product over a constant number of "neuron indices" $\alpha \in [n]$ will still result in an $O(1)$ quantity. The proof is again routine manipulation using Lemma A.7.2 and standard inequalities after applying the higher order product rule bound in Lemma A.7.11.

Proof. Let $l \overset{\text{def}}{=} |\mathcal{N}|$ and $v \in \mathbb{R}^l$ be the vector $(z_\alpha)_{\alpha\in\mathcal{N}}$ for an arbtirary ordering of $\mathcal{N}$ (including multiplicity). Thus $z^{\mathcal{N}} = \prod_j v_j$. Fix a ordering $\mathbf{U}$ of $\mathcal{U}$. By Lemma A.7.11 and Lemma A.7.2, there is a constant G depending only on $|\mathcal{U}|$ and p such that

$$|\partial^{\mathcal{U}} z^{\mathcal{N}}|^p \le G\sum_{t=0}^{l-1} \sqrt{l^{l-t-1}\|v\|_{2t}^{2tp}} \sum_\tau \sqrt{\prod_{\mathbb{Q}} \|\partial^{\mathbb{Q}} v\|^{2p}}$$

where τ ranges over partitions of $\{1,\ldots,|\mathcal{U}|\}$ of size $l-t$ and $\mathbb{Q}$ ranges over $\mathbf{U}[\tau]$ as in Lemma A.7.9. Here $\|v\|_0^0$ is by convention defined to be l. By Jensen's (concave) inequality, we then have

$$\mathbb{E}\,|\partial^{\mathcal{U}} z^{\mathcal{N}}|^p \le G\sum_{t=0}^{l-1} \sqrt{l^{l-t-1}\mathbb{E}\,\|v\|_{2t}^{2tp}} \sum_\tau \sqrt{\mathbb{E}\prod_{\mathbb{Q}} \|\partial^{\mathbb{Q}} v\|^{2p}}$$

Since the sizes of the ranges of t and of τ both depend only on $l = |\mathcal{N}|$ and $|\mathcal{U}|$, it suffices to show that both $\mathbb{E}\,\|v\|_{2t}^{2tp}$ and $\mathbb{E}\prod_{\mathbb{Q}} \|\partial^{\mathbb{Q}} v\|^{2p}$ have bounds that are $(\mathsf{j}, p, \mathcal{N}, \mathcal{P}, \mathcal{U})$-oblivious wrt ψ and the program.

Bounding $\mathbb{E}\|v\|_{2t}^{2t}$ Again, by Lemma A.7.2, $\mathbb{E}\|v\|_{2t}^{2tp} = O(\mathbb{E}\|v\|_{2tp}^{2tp})$, so it suffices to bound the latter. Now for all $t \in \{0, \dots, l-1\}$,

$$\mathbb{E}\|v\|_{2tp}^{2tp} = \sum_{\alpha \in \mathcal{N}} \mathbb{E}|z_\alpha|^{2tp} \le l \sup_{\alpha \in [n]} \mathbb{E}|z_\alpha|^{2tp} \le l B^{\psi}_{\mathsf{j}, 2tp, |\mathcal{P}|}$$

by A.9.1(j). This obviously satisfies the desired property.

Bounding $\mathbb{E}\prod_{\mathbb{Q}}\|\partial^{\mathbb{Q}} v\|^2$ Let $s \stackrel{\text{def}}{=} l - t$, so that the product $\prod_{\mathbb{Q}}$ iterates over s elements. Then by Hölder's Inequality,

$$\mathbb{E}\prod_{\mathbb{Q}}\|\partial^{\mathbb{Q}} v\|^{2p} \le \sqrt[s]{\prod_{\mathbb{Q}} \mathbb{E}\|\partial^{\mathbb{Q}} v\|^{2sp}}$$

By Lemma A.7.1, for a constant R depending on only l and s, we have

$$\begin{aligned}
\mathbb{E}\|\partial^{\mathbb{Q}} v\|^{2sp} &\le R\mathbb{E}\|\partial^{\mathbb{Q}} v\|_{2sp}^{2sp} \le lR \sup_{\alpha \in [n]} \mathbb{E}|\partial^{\mathbb{Q}} z_\alpha|^{2sp} \\
&= lR \sup_{\alpha \in [n]} \mathbb{E}|\partial^{\mathbb{Q}} \partial^{\mathcal{P}} \psi(g_\alpha^1, \dots, g_\alpha^{\mathsf{j}}; c^1, \dots, c^{\mathsf{j}})|^{2sp} \\
&\le lRB^{\psi}_{\mathsf{j}, 2sp, |\mathbb{Q}|+|\mathcal{P}|}
\end{aligned}$$

by A.9.1(j). From this, it's clear that $\mathbb{E}\prod_{\mathbb{Q}}\|\partial^{\mathbb{Q}} v\|^2$ has the desired property as well. □

A.9.2 Base Case: A.9.1$(1, \dots, M_0)$ and A.9.1$(1, \dots, M_0)$

Here we consider the case of $\mathsf{j} = 1, \dots, M_0$.

When $|\mathcal{P}| > 0$, A.9.1(j) and A.9.1(j) are trivially true since there's no dependence on the matrices A^i yet.

Now assume $|\mathcal{P}| = 0$. Then A.9.1(j) follows from standard Gaussian moment expressions. For A.9.1(j), we note $x \stackrel{\text{def}}{=} \psi(g^1, \dots, g^{\mathsf{j}}; c^1, \dots, c^{\mathsf{j}})$ has

$$\begin{aligned}
|x_\alpha| &\le C(1 + |g_\alpha^1|^q + \cdots + |g_\alpha^{\mathsf{j}}|^q + |c^1|^q + \cdots + |c^{\mathsf{j}}|^q) \\
&\le C(1 + |g_\alpha^1|^q + \cdots + |g_\alpha^{\mathsf{j}}|^q + \mathsf{j}R_q)
\end{aligned}$$

where C, q come from a profile of ψ and R_q is the qth moment bound on all the initial scalars. Then again we can apply standard Gaussian moment expressions to derive A.9.1(j).

A.9.3 A.9.1$(1, \dots, \mathsf{j})$ and A.9.1$(1, \dots, \mathsf{j}-1)$ Imply A.9.1(j)

Here we assume A.9.1$(1, \dots, \mathsf{j})$ and A.9.1$(1, \dots, \mathsf{j}-1)$ for $\mathsf{j} \ge M_0 + 1$ and derive A.9.1(j).

Let $u \stackrel{\text{def}}{=} (g_\alpha^1, \dots, g_\alpha^{\mathsf{j}}; c^1, \dots, c^{\mathsf{j}}) \in \mathbb{R}^{2\mathsf{j}}$ and let $\mathbf{P}$ be any ordering of $\mathcal{P}$. By Lemma A.7.10, we have

$$|\partial^{\mathbf{P}} \psi(u)| \le \sum_{\tau} \|D^{|\tau|}\| \cdot \prod_{\mathbb{Q}} \|\partial^{\mathbb{Q}} u\|$$

where $D^k \stackrel{\text{def}}{=} \nabla_u^k \psi(u) \in \mathbb{R}^{(2\mathsf{j})^k}$, τ ranges over partitions of $\{1, \dots, |\mathcal{P}|\}$, and, for each τ, $\mathbb{Q}$ ranges over the elements (which are multisets) of the partition $\mathbf{P}[\tau] = \{\{P_i : i \in \mathcal{S}\} : \mathcal{S} \in \tau\}$ of $\mathbf{P}$.

Thus, applying Lemma A.7.2,

$$|\partial^{\mathbf{P}}\psi(u)|^p \le \left(\sum_\tau \|D^{|\tau|}\| \cdot \prod_{\mathbb{Q}} \|\partial^{\mathbb{Q}} u\|\right)^p \le G \sum_\tau \|D^{|\tau|}\|^p \cdot \prod_{\mathbb{Q}} \|\partial^{\mathbb{Q}} u\|^p$$

where G is the constant from Lemma A.7.2 that depends only on p and $|\mathcal{P}|$ (through the number of partitions of $\{1,\dots,|\mathcal{P}|\}$). Taking expectation and applying another Cauchy-Schwarz gives

$$\mathbb{E}\,|\partial^{\mathbf{P}}\psi(u)|^p \le G \sum_\tau \sqrt{\mathbb{E}\,\|D^{|\tau|}\|^{2p}} \cdot \sqrt{\mathbb{E}\prod_{\mathbb{Q}} \|\partial^{\mathbb{Q}} u\|^{2p}}$$

Since the number of partitions of $\{1,\dots,|\mathcal{P}|\}$ (which is the range of τ) depends only on $|\mathcal{P}|$, it suffices to prove that both $\mathbb{E}\,\|D^{|\tau|}\|^{2p}$ and $\mathbb{E}\prod_{\mathbb{Q}} \|\partial^{\mathbb{Q}} u\|^{2p}$ are bounded by constants that are $(\mathsf{j}, p, \mathcal{P})$-oblivious wrt ψ and the program and are independent of α.

Bounding $\mathbb{E}\,\|D^{|\tau|}\|^{2p}$ This follows directly from Proposition A.9.8 (which is a straightforward bound by replacing each partial derivative of ψ with its polynomial upper bound).

Bounding $\mathbb{E}\prod_{\mathbb{Q}} \|\partial^{\mathbb{Q}} u\|^{2p}$ First, note this quantity clearly does not depend on ψ.

Let $k = |\tau|$. Recall $\mathbb{Q}$ ranges over the k multisets $\{P_i : i \in \mathcal{S}\}$ as $\mathcal{S}$ ranges over elements of the partition τ. By Hölder's Inequality and Lemma A.7.1,

$$\mathbb{E}\prod_{\mathbb{Q}} \|\partial^{\mathbb{Q}} u\|^{2p} \le \prod_{\mathbb{Q}} \sqrt[k]{\mathbb{E}\,\|\partial^{\mathbb{Q}} u\|^{2pk}} \le R \prod_{\mathbb{Q}} \sqrt[k]{\mathbb{E}\,\|\partial^{\mathbb{Q}} u\|_{2pk}^{2pk}}$$

where R is a constant depending only on $2pk$ and j (coming from Lemma A.7.1). Since $|\mathbb{Q}|$ and $k = |\tau|$ are both bounded by $|\mathbf{P}|$, it suffices to show $\mathbb{E}\,\|\partial^{\mathbb{Q}} u\|^{2pk}$ is bounded by a constant independent of α and is $(\mathsf{j}, p, k, \mathbb{Q})$-oblivious wrt ψ and the program. Now,

$$\|\partial^{\mathbb{Q}} u\|_{2pk}^{2pk} = \sum_{i=1}^{\mathsf{j}} (\partial^{\mathbb{Q}} g_\alpha^i)^{2pk} + (\partial^{\mathbb{Q}} c^i)^{2pk} \le \sum_{i=1}^{\mathsf{j}} B_{i,2pk,|\mathbb{Q}|} + B^{\phi^i}_{i-1,2pk,|\mathbb{Q}|}$$

by A.9.1$(1,\dots,\mathsf{j})$ and Proposition A.9.7. This bound indeed is independent of α and has the required oblivious property.

A.9.4 A.9.1$(\mathsf{j}-1)$ Implies A.9.1(j)

Assume $\mathsf{j} \ge M_0 + 1$. Let $y_\beta \stackrel{\text{def}}{=} \phi^{\mathsf{j}}(g_\beta^1, \dots, g_\beta^{\mathsf{j}-1}; c^1, \dots, c^{\mathsf{j}-1})$. Recall $g_\alpha^{\mathsf{j}} = \sum_{\beta=1}^n W_{\alpha\beta}^{\mathsf{j}} y_\beta$, where W^{j} is one of the program's matrices A^i or their transposes.

Reduction via Product Rule Then by the product rule of differentiation,

$$\partial^{\mathcal{P}} g^{\mathsf{j}}_{\alpha} = \left(\sum_{\beta=1}^{n} W^{\mathsf{j}}_{\alpha\beta} \partial^{\mathcal{P}} y_{\beta} \right) + remainder$$

where *remainder* is a sum of at most $|\mathcal{P}|$ elements of the form $\partial^{\mathcal{P}'} y_\beta$ where $\mathcal{P}'$ is $\mathcal{P}$ with some element removed. Therefore, by Lemma A.7.2,

$$|\partial^{\mathcal{P}} g^{\mathsf{j}}_{\alpha}|^p \le R \left[\left| \sum_{\beta=1}^{n} W^{\mathsf{j}}_{\alpha\beta} \partial^{\mathcal{P}} y_{\beta} \right|^p + |remainder|^p \right]$$

where R depends only on p. We can easily bound $\mathbb{E}\,|remainder|^p$ by a $(\mathsf{j}, p, \mathcal{P})$-oblivious constant independent of α using A.9.1($\mathsf{j}-1$) and Lemma A.7.2. Thus, it suffices to bound $\left|\sum_{\beta=1}^{n} W^{\mathsf{j}}_{\alpha\beta} \partial^{\mathcal{P}} y_\beta\right|^p$ by a constant with the same property.

Plan: Show Small Moment Condition (SMC) Holds To do so, we will show the vector with entries $w_\beta z_\beta$ where $w_\beta \overset{\text{def}}{=} W^{\mathsf{j}}_{\alpha\beta}$ and $z_\beta \overset{\text{def}}{=} \partial^{\mathcal{P}} y_\beta$ satisfies the Small Moment Condition (Definition A.8.3) and apply Lemma A.8.4 to it. In particular, we will assume WLOG that p is an even integer and prove that for every multiset $\mathcal{N}$ of $[n]$ of size $|\mathcal{N}| = p$, we have

$$\mathbb{E}\, w^{\mathcal{N}} z^{\mathcal{N}} \le C n^{-|\mathrm{uniq}(\mathcal{N})|} \tag{A.58}$$

for a constant C that is $(\mathsf{j}, \mathcal{N}, \mathcal{P})$-oblivious, where $\mathrm{uniq}(\mathcal{N})$ is the set of unique elements of $\mathcal{N}$.

Taylor Expansion of $z^{\mathcal{N}}$ Fix $\mathcal{N}$. We now consider $z^{\mathcal{N}}$ as a function of w_β for unique elements $\beta \in \mathrm{uniq}(\mathcal{N})$ of $\mathcal{N}$ (keeping other weight entries fixed). Let $v = (w_\beta)_{\beta \in \mathrm{uniq}(\mathcal{N})} \in \mathbb{R}^{|\mathrm{uniq}(\mathcal{N})|}$ be the vector of such elements, so that we write $z^{\mathcal{N}} = z^{\mathcal{N}}(v)$ as function of v. By Lemma A.7.8, we Taylor expand $z^{\mathcal{N}}$ to the rth order, for some r to be determined later:

$$\begin{aligned} z^{\mathcal{N}} &= z^{\mathcal{N}}(0) + \langle \nabla z^{\mathcal{N}}(0), v\rangle + \cdots + \frac{1}{r!}\langle \nabla^r z^{\mathcal{N}}(0), v^{\otimes r}\rangle + R_{r+1} \\ &= R_{r+1} + \sum_{s=0}^{r} \frac{1}{s!}\langle \nabla^s z^{\mathcal{N}}(0), v^{\otimes s}\rangle \\ \text{where} \quad R_{r+1} &\overset{\text{def}}{=} \frac{1}{r!}\int_0^1 (1-t)^r \langle \nabla^{r+1} z^{\mathcal{N}}(tv), v^{\otimes (r+1)}\rangle dt \end{aligned}$$

While the tensors appearing in this expansion may seem at first like "large objects", note that in terms of n, the tensors have constant sizes, so their norms are entirely determined by how their entries scale with n. Intuitively, the derivative tensors $\nabla^s z^{\mathcal{N}}$ will have $O(1)$ entry sizes, by induction hypothesis, while $v^{\otimes s}$ has size $O(n^{-s/2})$. But before we keep following this logic of naive bounds, it pays to notice there are a lot of cancellation.

Cancellation Using Independence and Zero-Mean Because now $\nabla^s z^{\mathcal{N}}(0)$ no longer depends on and thus is independent (as a random variable) from v (and thus $w^{\mathcal{N}}$)[5], taking expectation we

[5] this was the main purpose of the Taylor expansion

now have

$$\mathbb{E}\, w^{\mathcal{N}} z^{\mathcal{N}} = \mathbb{E}\, w^{\mathcal{N}} R_{r+1} + \sum_{s=0}^{r} \frac{1}{s!} \langle \mathbb{E}\, \nabla^s z^{\mathcal{N}}(0), \mathbb{E}\, w^{\mathcal{N}} v^{\otimes s} \rangle$$

where $w^{\mathcal{N}} v^{\otimes s}$ is the scalar multiplication of the scalar $w^{\mathcal{N}}$ with the tensor $v^{\otimes s}$.

Now suppose

$$\mathcal{N} \text{ has exactly } k \text{ elements that appear singly (i.e., have multiplicity 1) in } \mathcal{N}.$$

Then $\mathbb{E}\, w^{\mathcal{N}} v^{\otimes s}$ will be 0 for all $s < k$: indeed, every entry of $w^{\mathcal{N}} v^{\otimes s}$ is a monomial $w^{\mathcal{N}'}$ that will have some w_β appearing by itself in the product (i.e., has degree 1), so that

$$\mathbb{E}\, w^{\mathcal{N}'} = \mathbb{E}\, w_\beta \mathbb{E}\, w^{\mathcal{N}' \setminus \{\beta\}} = 0$$

using the fact that w_β is zero-mean and independent from $w^{\mathcal{N}' \setminus \{\beta\}}$.

Therefore, we will take r (the order of the Taylor expansion) to be $k-1$, so that

$$\mathbb{E}\, w^{\mathcal{N}} z^{\mathcal{N}} = \mathbb{E}\, w^{\mathcal{N}} R_{k+1}.$$

Unwinding the definition of R_{k+1} and using Cauchy-Schwarz, we have

$$\mathbb{E}\, w^{\mathcal{N}} z^{\mathcal{N}} \le \frac{1}{(k-1)!} \int_0^1 (1-t)^{k-1} \sqrt{\mathbb{E}\, \|\nabla^k z^{\mathcal{N}}(tv)\|^2 \cdot \mathbb{E}\, \|w^{\mathcal{N}} v^{\otimes k}\|^2}\, dt \tag{A.59}$$

Constructing the SMC Constant Let $\mathcal{N}_1$ be the subset of elements of $\mathcal{N}$ with multiplicity 1 (so that $|\mathcal{N}_1| = k$) and let $\mathcal{N}' = \mathcal{N} \setminus \mathcal{N}_1$. We will show that

$$\mathbb{E}\, w^{\mathcal{N}} z^{\mathcal{N}} \le B_k n^{-|\mathrm{uniq}(\mathcal{N})|} \tag{A.60}$$

for some constant B_k that is $(\mathrm{j}, \mathcal{N}_1, \mathcal{N}', \mathcal{P})$-oblivious. In particular, this means B_k depends on $\mathcal{N}$ only through $|\mathcal{N}| = |\mathcal{N}_1| + |\mathcal{N}'| = p$ and $|\mathcal{N}_1| = k$. Then the constant C in Eq. (A.58) can be taken as $\max_k B_k$ where k ranges from 0 to p.

In light of Eq. (A.59), to prove Eq. (A.60), it thus suffices to show that

$$\mathbb{E}\, \|\nabla^k z^{\mathcal{N}}(tv)\|^2 = O(1)$$
$$\mathbb{E}\, \|w^{\mathcal{N}} v^{\otimes k}\|^2 = O(n^{-2|\mathrm{uniq}(\mathcal{N})|})$$

where the big-Os hide $(\mathrm{j}, \mathcal{N}_1, \mathcal{N}', \mathcal{P})$-oblivious constants that furthermore are independent of $t \in [0,1]$.

Bounding $\mathbb{E}\, \|w^{\mathcal{N}} v^{\otimes k}\|^2$ Each entry of the tensor $w^{\mathcal{N}} v^{\otimes k}$ is just a product of $|\mathcal{N}| + k = p + k$ matrix entries, whose expected square norm can be bounded by $\nu_{2(k+p)} n^{-(k+p)}$, where $\nu_{2(k+p)}$ is the scaled moment bound on the matrix entries we fixed at the beginning of this proof. There are $|\mathrm{uniq}(\mathcal{N})|^k \le p^k$ entries in this tensor, so

$$\mathbb{E}\, \|w^{\mathcal{N}} v^{\otimes k}\|^2 \le p^k \nu_{2(k+p)} n^{-(k+p)}$$

Now note that, by the definition of k, we have $k + p \ge 2|\mathrm{uniq}(\mathcal{N})|$. Thus

$$\mathbb{E}\, \|w^{\mathcal{N}} v^{\otimes k}\|^2 = O(n^{-2|\mathrm{uniq}(\mathcal{N})|})$$

where the constant in $O(-)$ is $(p, \mathcal{N}_1, \mathcal{N}', \mathcal{P})$-oblivious.

Bounding $\mathbb{E}\,\|\nabla^k z^{\mathcal{N}}(tv)\|^2$. Recall that all bounds in this proof (Appendix A.9) are oblivious wrt the program, so they only depend on the program through the bounds $(\nu_q)_q, (R_q)_q, \mathcal{C}$ fixed at the beginning of this proof. Notice that $\nabla^k z^{\mathcal{N}}(tv)$ is just $\nabla^k z^{\mathcal{N}}$ computed in the program where the matrix entries $\{W^{\mathrm{j}}_{\alpha\beta} : \beta \in \mathcal{N}\}$ (which are the entries of v) are scaled *down* and such a program satisfies the exact same data (in particular the moment bounds given by $(\nu_q)_q$).

Thus, by Proposition A.9.9, *every* entry of $\nabla^k z^{\mathcal{N}}(tv)$ has the same $(\mathrm{j}, \mathcal{N}_1, \mathcal{N}', \mathcal{P})$-oblivious bound C on its expected square norm, *uniformly* over $t \in [0,1]$. Because $\nabla^k z^{\mathcal{N}}(tv)$ has $|\mathrm{uniq}(\mathcal{N})|^k \le p^k$ entries,

$$\mathbb{E}\,\|\nabla^k z^{\mathcal{N}}(tv)\|^2 \le Cp^k$$

which is $(\mathrm{j}, \mathcal{N}_1, \mathcal{N}', \mathcal{P})$-oblivious and independent of $t \in [0,1]$, as desired.

A.10 Program Transformations

A.10.1 Backpropagation Program

Given a program T as in Eq. (3.4) and a polynomially smooth function $\psi : \mathbb{R}^{2M} \to \mathbb{R}$, we can create a new program T_ψ that extends T, which we call the *backpropagation program of T with respect to ψ*, or just *backprogram* for short. Intuitively, T_ψ will compute the gradients of

$$c \overset{\text{def}}{=} \frac{1}{n}\sum_{\alpha=1}^{n} \psi(g^1_\alpha, \dots, g^M_\alpha; c^1, \dots, c^M) \tag{A.61}$$

with respect to all vectors in the program. In the context of this work, the importance of the backprogram construction is to easily express the partial derivative $\delta = \frac{\partial c}{\partial A^j_{\alpha\beta}}$ (Proposition A.10.2), which easily shows that $\delta = O(n^{-1})$ *instead of* $O(1)$ *as suggested by Proposition A.9.7* (see Lemma A.10.6). This will be crucial in the proof of Theorem 3.5.2.

Explicitly, T_ψ is constructed as follows. It has the same matrices A^i as T. The first M vectors and scalars g^i and c^i for $i = 1, \dots, M$ are the same as in T. It additionally has new vectors and scalars constructed after them. We will first describe the *mathematical objects* that will be computed by these new vectors and scalars, before discussing how to represent them in the form of Eq. (3.4).

Notation In this context, we use Sans Serif font to represent these mathematical objects, to distinguish them from objects in the program itself. Recall that ϕ^i is the nonlinearity used in iteration i in the original program T (as well as in the new program T_ψ). For $M \ge j > i$, we will write $\mathsf{x}^j_{i;} \in \mathbb{R}^n$ and $\mathsf{x}^j_{;i} \in \mathbb{R}^n$ for the vectors with entries

$$(\mathsf{x}^j_{i;})_\alpha \overset{\text{def}}{=} \partial_{g^i}\phi^j(g^1, \dots, g^{j-1}; c^1, \dots, c^{j-1})$$
$$(\mathsf{x}^j_{;i})_\alpha \overset{\text{def}}{=} \partial_{c^i}\phi^j(g^1, \dots, g^{j-1}; c^1, \dots, c^{j-1}).$$

In other words, $\mathsf{x}^j_{i;}$ is the partial derivative $\partial_{g^i}\phi^j$ with respect to the argument g^i and likewise $\mathsf{x}^j_{;i}$ is the partial derivative $\partial_{c^i}\phi^j$ with respect to the argument c^i, both evaluated on the original set of inputs for ϕ^j. For convenience, we will also write $\mathsf{x}^{M+1}_{i;}$ and $\mathsf{x}^{M+1}_{;i}$ for the partial derivatives of ψ,

i.e.,

$$(\mathsf{x}_{i;}^{M+1})_\alpha \stackrel{\text{def}}{=} \partial_{g^i}\psi(g^1,\dots,g^M;c^1,\dots,c^M)$$
$$(\mathsf{x}_{;i}^{M+1})_\alpha \stackrel{\text{def}}{=} \partial_{c^i}\psi(g^1,\dots,g^M;c^1,\dots,c^M).$$

In addition, for a given vector $v \in \mathbb{R}^n$, we write $\langle v\rangle \stackrel{\text{def}}{=} \frac{1}{n}\sum_{\alpha=1}^n v_\alpha$ for the average of the entries of v.

Mathematical Idea We iteratively construct the vectors $\mathsf{dx}^i, \mathsf{dg}^i \in \mathbb{R}^n$ for decreasing $i = M, M-1, \dots, M_0+1$ as follows.

$$\mathsf{dx}^{M+1} \stackrel{\text{def}}{=} 1 \in \mathbb{R}^n$$
$$\mathsf{dg}^i \stackrel{\text{def}}{=} \sum_{k=i+1}^{M+1} \mathsf{x}_{i;}^k \odot \mathsf{dx}^k + \sum_{M+1\geq j>k>i} \langle \mathsf{dx}^j \odot \mathsf{x}_{;k}^j\rangle x_{i;}^k \tag{A.62}$$
$$\mathsf{dx}^i \stackrel{\text{def}}{=} W^{i\top}\mathsf{dg}^i \tag{A.63}$$

where $W^{i\top}$ in the last line is the transpose of the same matrix used in iteration i of the original program T, as in Eq. (3.4). For example, the first few dg^i and dx^i looks like the following.

$$\begin{aligned}
\mathsf{dx}^{M+1} &= 1\\
\mathsf{dg}^M &= \mathsf{x}_{M;}^{M+1}\\
\mathsf{dx}^M &= W^{M\top}\mathsf{dg}^M\\
\mathsf{dg}^{M-1} &= \mathsf{x}_{M-1;}^{M+1} + \mathsf{x}_{M-1;}^{M} \odot \mathsf{dx}^M + \langle \mathsf{x}_{;M}^{M+1}\rangle \mathsf{x}_{M-1;}^M\\
\mathsf{dx}^{M-1} &= W^{M-1\top}\mathsf{dg}^{M-1}
\end{aligned}$$

One can verify the following statement using the chain rule.

Proposition A.10.1. *For $i = M, M-1, \dots, M_0+1$, the vector dx^i (Eq. (A.63)) equals the scaled total gradient $n\nabla_{x^i}c$ of c (scaled up by n) against the vector x^i defined in Eq. (3.4), and likewise dg^i (Eq. (A.62)) equals $n\nabla_{g^i}c$.*

The vectors dx^i and dg^i make it easy to express the partial derivative $\frac{\partial c}{\partial A^i_{\alpha\beta}}$ as well:

Proposition A.10.2. *For any matrix A^j of program T, for ψ, c as in Eq. (A.61) and dg^i as in Eq. (A.62), we have*

$$\frac{\partial c}{\partial A^j_{\alpha\beta}} = \sum_{i:W^i=A^j} \frac{1}{n}(\mathsf{dg}^i)_\alpha (x^i)_\beta + \sum_{i:W^i=A^{j\top}} \frac{1}{n}(\mathsf{dg}^i)_\beta (x^i)_\alpha,$$

where in the first sum we iterate over indices i such that $W^i = A^j$ and in the second, i such that $W^i = A^{j\top}$, both in the context of the original program T.

Proof. Each A^j is used in the computation of c only through any W^i that equals A^j or its transpose. Thus

$$\frac{\partial c}{\partial A^j_{\alpha\beta}} = \sum_{i:W^i=A^j} \frac{\partial c}{\partial W^i_{\alpha\beta}} + \sum_{i:W^i=A^{j\top}} \frac{\partial c}{\partial W^i_{\beta\alpha}}$$

But, from Proposition A.10.1 and chain rule, one easily calculates

$$\frac{\partial c}{\partial W^i_{\alpha\beta}} = \frac{1}{n}(\mathsf{dg}^i)_\alpha (x^i)_\beta.$$

□

Program T_ψ Construction Here we will construct the vectors g^a for $a = M+1, M+2, \dots$ for the new program T_ψ. To avoid confusion with vectors in the original program, we use superscript index a, b, e for talking about the new vectors, while i, j, k are reserved for indices of the old program. Our goal is to express each dx^i as some g^a with dg^i being the corresponding x^a (c.f. Eq. (3.4)). To do so, as is apparent from Eq. (A.62), we need to express $\langle \mathsf{dx}^j \odot \mathsf{x}^j_{;k} \rangle$ for each $M+1 \geq j > k$ as scalars c^b as well.

So the strategy is to, in descending order of i, alternatingly express dx^i, then the scalars $\langle \mathsf{dx}^j \odot \mathsf{x}^j_{;k} \rangle$ for $M+1 \geq j > k > i-1$, then express dg^{i-1} through Eq. (A.62) and finally dx^{i-1} through Eq. (A.63), and repeat. At the end, the vectors of T_ψ contain $g^1, \dots, g^M$ (same as in T) and $\mathsf{dx}^{M+1} \dots, \mathsf{dx}^{M_0+1}$ (new vectors) and the scalars contain $c^1, \dots, c^M$ (same as in T) and $\{\langle \mathsf{dx}^j \odot \mathsf{x}^j_{;k} \rangle\}_{M+1 \geq j > k > M_0}$ (new scalars). There are also other "junk" vectors (resp. scalars) that we don't care about, which arise when we only want to express some scalar (resp. vectors). This shows

Proposition A.10.3. *If T has $M - M_0$ iterations, then T_ψ has at most $(M - M_0)^2$ iterations.*

Combining this with the fact that the nonlinearities of T_ψ are just compositions of polynomials with first order derivatives of those of T, we deduce the following

Proposition A.10.4. *Any $(\mathcal{P}_1, \dots, \mathcal{P}_r)$-oblivious constant wrt T_ψ and $\psi^1, \dots, \psi^l$ is also $(\mathcal{P}_1, \dots, \mathcal{P}_r)$-oblivious wrt T and $\psi^1, \dots, \psi^l$.*

It's clear from Eq. (A.62) that dg^i can be expressed as some nonlinearity applied to vectors of T_ψ. Beyond this fact, the exact construction of T_ψ is not important for our purposes, so we will not detail it further here.

Proposition A.10.5. *For any program T and c, ψ as in Eq. (A.61), each $\mathsf{dg}^i = n\nabla_{g^i} c$ can be expressed as some nonlinearity ϕ applied to the vectors and scalars of T_ψ. If all nonlinearities of T are polynomially smooth, then ϕ is polynomially smooth as well.*

Matrix Derivative Bound

Lemma A.10.6. *Consider a program in Setup A.9.2. Then for any $p \geq 1$, any nonempty multiset $\mathcal{P}$ taking values in the program's matrix entries $\{A^i_{\alpha\beta}\}_{\alpha,\beta,i}$, and any scalar c of the program,*

$$\mathbb{E}\left|\partial^{\mathcal{P}} c\right|^p = O(n^{-p}) \quad \text{as } n \to \infty. \tag{A.64}$$

Furthermore, the constant in the big-O is $(p, \mathcal{P})$-oblivious wrt the program.

Note importantly that $\mathcal{P}$ has to be nonempty for this to hold, since without taking derivatives, c can definitely be $\Theta(1)$ (for example, if its limit $\mathring{c}$ is nonzero). This lemma improves on Lemma A.9.6 drastically when $\mathcal{P}$ is nonempty. The reason that Lemma A.9.6 is so loose in such cases is that, in the sum over $\alpha \in [n]$ in Eq. (A.56), only a constant number of α really achieves the sup bound in Lemma A.9.3. So the naive way Lemma A.9.6 converts the sup bound to an average bound turned out to leave a lot of room.

Proof. Suppose c is the ith scalar. For $i \le M_0$, the derivative is 0, so consider the case where $i > M_0$. For brevity, write ψ for ϕ^i (the nonlinearity used to create c^i). Construct the backprogram T_ψ with respect to ψ. By Proposition A.10.4, it suffices to show the bound for a constant $(p, \mathcal{P})$-oblivious wrt T_ψ.

Since $\mathcal{P}$ is nonempty, there is an element a of $\mathcal{P}$. Write $\mathcal{P}' = \mathcal{P} \setminus \{a\}$. Then by Proposition A.10.2,

$$\partial^{\mathcal{P}} c = \partial^{\mathcal{P}'}(\partial_a c) = \frac{1}{n} \sum \partial^{\mathcal{P}'}[(\mathsf{dg}^j)_\alpha (x^j)_\beta],$$

where the sum is over some collection of (j, α, β) of size at most $M - M_0$ (an upper bound on how many times a has been used in the original program).

Then a routine combination of Lemma A.7.2, product rule (Lemma A.7.11), and Cauchy-Schwarz reduces our problem to bounding $\mathbb{E}\,|\partial^{\mathbb{Q}}(\mathsf{dg}^j)_\alpha|^{2p}$ and $\mathbb{E}\,|\partial^{\mathbb{Q}}(x^j)_\beta|^{2p}$ for all subsets $\mathbb{Q}$ of $\mathcal{P}'$ by a $(p, \mathbb{Q})$-oblivious constant independent of α and β. This is precisely provided by Lemma A.9.3 applied to the backprogram T_ψ. □

Lemma A.10.7. *Consider a program in Setup A.9.2. Let $\mathbb{Q}_1, \dots, \mathbb{Q}_r$ be nonempty and let $\mathcal{P}$ be potentially empty multisets of matrix entries. Then for any scalar c of the program,*

$$\mathbb{E}\left|\partial^{\mathcal{P}} \prod_{i=1}^{r} \partial^{\mathbb{Q}_i} c\right|^p = O(n^{-rp})$$

where the hidden constant is $(p, \mathbb{Q}_1, \dots, \mathbb{Q}_r, \mathcal{P})$-oblivious wrt the program.

Proof. The product rule (Lemma A.7.11), Lemma A.7.2, and Hölder's Inequality show that the LHS is, within a constant factor depending only on p and $|\mathcal{P}|$, bounded by a sum of $O(1)$ number (depending only on p) of terms, each of which is a product over r elements of the form

$$\mathbb{E}\,|\partial^{\mathbb{Q}} c|^{rp}$$

for multisets $\mathbb{Q}$ with size at most $|\mathcal{P}| + \max_i |\mathbb{Q}_i|$. Then the desired result follows from Lemma A.10.6. □

A.11 Proof of Theorem 3.5.2

We will prove the following more specific version of Theorem 3.5.2, which says more about the hidden constant. Recall the dot derivative notation from Section 3.5 as well as the notion of *oblivious constants* in Definition A.9.1.

Theorem A.11.1. *Consider a program in Setup 3.3.6 and its interpolation (Definition 3.5.1), and let c be a scalar in it. Then for any finite $p \ge 1$,*

$$\sup_t \mathbb{E}\,\dot{c}(t)^p = O(n^{-p/2})$$

where the hidden constant is p-oblivious wrt the program.

Proof. By the power-mean inequality, it suffices to show this for any even integer p. Let $a = a(t)$ be the vector of all matrix entries in the program, which is an interpolation of the Gaussian and

non-Gaussian matrix entries. Let N be its dimension (which is equal to n^2 times the number of matrices in the program). We will use κ to index a's entries. Denote its derivative against t by $\dot{a} \in \mathbb{R}^N$, as in Section 3.5. For brevity, we will suppress the argument (t) below.

Let $D \stackrel{\text{def}}{=} \nabla_a c \in \mathbb{R}^N$, so that for any multiset $\mathcal{P}$ taking values in $[N]$, $D^{\mathcal{P}} = \prod_{\kappa \in \mathcal{P}} \frac{\partial c}{\partial a_\kappa}$. By chain rule, we have

$$\dot{c} = \langle D, \dot{a} \rangle = \langle 1, D \odot \dot{a} \rangle.$$

From here on, the proof follows an outline similar to that of Appendix A.9.4, except in how the cancellation happens.

Proof Plan: Small Moment Condition. We will show that the N-dimensional vector $\sqrt{n} D \odot \dot{a}$ (which has entries $\sqrt{n} D_\kappa \dot{a}_\kappa$) satisfies the Small Moment Condition (Definition A.8.3): For any even $p \geq 0$ and any multiset $\mathcal{P}$ of size p taking values in $[N]$, we need to show there's a $\mathcal{P}$-oblivious constant C wrt the program that is also independent of t such that

$$n^{p/2} D^{\mathcal{P}} \dot{a}^{\mathcal{P}} \leq C N^{-|\text{uniq}(\mathcal{P})|}. \tag{A.65}$$

Then by Lemma A.8.4, we have

$$n^{p/2} \mathbb{E} \, |\dot{c}|^p = \mathbb{E} \, |\langle 1, \sqrt{n} D \odot \dot{a} \rangle|^p = O(1)$$

where the hidden constant is independent of $t \in [0, 1]$, which yields Theorem A.11.1.

Taylor Expansion. Now fix $\mathcal{P}$. Let $v \stackrel{\text{def}}{=} (a_\kappa)_{\kappa \in \text{uniq}(\mathcal{P})}$ be the vector of unique a_κ for $\kappa \in \mathcal{P}$ (for any ordering of uniq$(\mathcal{P})$). We think of $D^{\mathcal{P}}$ as a function $D^{\mathcal{P}}(v)$ of v. For an integer r to be specified later, we Taylor expand $D^{\mathcal{P}}$ in v around 0 to the rth order (Lemma A.7.8): with ∇^i denoting differentation wrt v,

$$D^{\mathcal{P}}(v) = D^{\mathcal{P}}(0) + \langle \nabla D^{\mathcal{P}}(0), v \rangle + \cdots + \frac{1}{r!} \langle \nabla^r D^{\mathcal{P}}(0), v^{\otimes r} \rangle + R_{r+1}$$
$$\text{where} \quad R_{r+1} \stackrel{\text{def}}{=} \frac{1}{r!} \int_0^1 (1 - \zeta)^r \langle \nabla^{r+1} D^{\mathcal{P}}(\zeta v), v^{\otimes (r+1)} \rangle d\zeta.$$

Cancellation Using Independence and Zero-Mean Now notice $\nabla^i D^{\mathcal{P}}(0)$ is independent from $v^{\otimes i}$ and $\dot{a}^{\mathcal{P}}$. Therefore, for $i = 0, \ldots, r$,

$$\mathbb{E} \, \langle \nabla^i D^{\mathcal{P}}(0), \dot{a}^{\mathcal{P}} v^{\otimes i} \rangle = \langle \mathbb{E} \, \nabla^i D^{\mathcal{P}}(0), \mathbb{E} \, \dot{a}^{\mathcal{P}} v^{\otimes i} \rangle.$$

Furthermore, notice $\mathbb{E} \, \dot{a}^{\mathcal{P}} v^{\otimes i} = 0$ for small values of i. Indeed, if

$$k \text{ is the number of elements in } \mathcal{P} \text{ appearing exactly once}$$

and $i < 2k$, then every entry of the expected tensor $\mathbb{E} \, \dot{a}^{\mathcal{P}} v^{\otimes i}$ has the form

$$(\mathbb{E} \, \dot{a}_\kappa) \cdot (other) \quad \text{or} \quad (\mathbb{E} \, \dot{a}_\kappa a_\kappa) \cdot (other)$$

for some a_κ where *other* is the expectation of some other monomial in a and $\dot{a}$ independent from a_κ and $\dot{a}_\kappa$. Both cases evaluate to 0 by Lemma 3.5.3. Thus, we now choose r (the degree of Taylor expansion) to be $2k-1$. Then

$$\mathbb{E}\,\dot{a}^{\mathcal{P}} D^{\mathcal{P}}(v) = \mathbb{E}\,\dot{a}^{\mathcal{P}} R_{2k}.$$

By Lemma A.7.2 and Cauchy-Schwarz, unwinding the definition of R_{2k}, we now have

$$\mathbb{E}\,\dot{a}^{\mathcal{P}} D^{\mathcal{P}}(v) \le \frac{1}{(2k-1)!}\int_0^1 (1-\zeta)^{2k-1}\sqrt{\mathbb{E}\,\|\nabla^{2k} D^{\mathcal{P}}(\zeta v)\|^2 \cdot \mathbb{E}\,\|\dot{a}^{\mathcal{P}} v^{\otimes 2k}\|^2}\,d\zeta \tag{A.66}$$

Let $\mathcal{P}_1$ be the subset of $\mathcal{P}$'s elements appearing in $\mathcal{P}$ exactly once, so that $|\mathcal{P}_1| = k$. Let $\mathcal{P}' = \mathcal{P} \setminus \mathcal{P}_1$. So it suffices to show

$$\sqrt{\mathbb{E}\,\|\dot{a}^{\mathcal{P}} v^{\otimes 2k}\|^2} = O(n^{-(2k+p)/2})$$
$$\sqrt{\mathbb{E}\,\|\nabla^{2k} D^{\mathcal{P}}(\zeta v)\|^2} = O(n^{-p})$$

where the hidden constants a) are $(\mathcal{P}_1, \mathcal{P}')$-oblivious and b) are independent of t (the interpolation variable) and ζ (the integration variable in the Taylor remainder R_{2k}). If these are shown, then, plugging into Eq. (A.66),

$$n^{p/2}\mathbb{E}\,\dot{a}^{\mathcal{P}} D^{\mathcal{P}}(v) = O(n^{-k-p}) = O(n^{-2|\mathrm{uniq}(\mathcal{P})|}) = O(N^{-|\mathrm{uniq}(\mathcal{P})|})$$

where the second equality follows because k is the number of elements of $\mathcal{P}$ with multiplicity 1 and $p = |\mathcal{P}|$, thus $k + p \ge 2|\mathrm{uniq}(\mathcal{P})|$. The hidden constant here will depend on $k = |\mathcal{P}_1|$, but we can take the max of such constants over all $k = 0, 1, \ldots, p$ to arrive at the desired $\mathcal{P}$-oblivious constant in Eq. (A.65).

Bounding $\dot{a}^{\mathcal{P}} v^{\otimes 2k}$. Each entry of $\dot{a}^{\mathcal{P}} v^{\otimes 2k}$ is a degree $2k + |\mathcal{P}| = 2k + p$ monomial in $\dot{a}$ and a. Thus, by Lemma 3.5.3, its expected square norm is bounded by $O(n^{-(2k+p)})$ uniformly over all entries, where the hidden constant depends only on the scaled moment bound ν_{2k+p} of the program's matrix entries. Because $\dot{a}^{\mathcal{P}} v^{\otimes k}$ has $|\mathrm{uniq}(\mathcal{P})|^{2k} \le p^{2k} = O(1)$ entries, we have

$$\mathbb{E}\,\|\dot{a}^{\mathcal{P}} v^{\otimes 2k}\|^2 = O(n^{-(2k+p)})$$

where the hidden constant is $(\mathcal{P}_1, \mathcal{P}')$-oblivious, as desired.

Bounding $\nabla^{2k} D^{\mathcal{P}}(\zeta v)$. We can think of $\nabla^{2k} D^{\mathcal{P}}(\zeta v)$ as $\nabla^{2k} D^{\mathcal{P}}$ computed on a program where the matrix entries in v are multiplied by a factor of $0 \le \zeta \le 1$. This program would then satisfy the same scaled moment bounds $(\nu_j)_{j=2}^{\infty}$ as the original program, uniformly over all t. Thus all of our oblivious bounds would apply to $\nabla^{2k} D^{\mathcal{P}}(\zeta v)$. In particular, Lemma A.10.7 tells us each entry of $\nabla^{2k} D^{\mathcal{P}}(\zeta v)$ has expected square norm bounded uniformly (over all entries) by $O(n^{-2|\mathcal{P}|}) = O(n^{-2p})$ with a $(\mathcal{P}_1, \mathcal{P}')$-oblivious constant independent of ζ and t. Finally, because there are $|\mathrm{uniq}(\mathcal{P})|^{2k} \le p^{2k} = O(1)$ entries, we get

$$\mathbb{E}\,\|\nabla^{2k} D^{\mathcal{P}}(\zeta v)\|^2 = O(n^{-2p})$$

$(\mathcal{P}_1, \mathcal{P}')$-obliviously and uniformly over ζ and t as desired.

□

A.12 L^p Convergence From Almost Sure Convergence

The following is a standard lemma.

Lemma A.12.1. *For any nonnegative random variable $X \in \mathbb{R}$ and deterministic $B \geq 0$, we have*

$$\mathbb{E}\, X\mathbb{I}(X > B) = B \Pr[X > B] + \int_B^\infty \Pr[X > t]dt$$

Proof. Integration by parts. □

Lemma A.12.2. *Suppose a sequence of random variables $(X_n)_{n=1}^\infty$ converges almost surely to 0. Then for any $p \in [1, \infty)$, X_n converges to 0 in L^p as well if for some $q > p$, there exists a constant C such that $\mathbb{E}\,|X_n|^q < C$ for all n.*

This lemma's proof is a standard truncation trick.

Proof. For any $B > 0$, we have

$$\mathbb{E}\,|X_n|^p = \mathbb{E}\,|X_n|^p \mathbb{I}(|X_n|^p \leq B) + \mathbb{E}\,|X_n|^p \mathbb{I}(|X_n|^p > B)$$

The random variable $|X_n|^p \mathbb{I}(|X_n|^p \leq B)$ converges almost surely to 0, so by dominated convergence, it also converges in mean:

$$\mathbb{E}\,|X_n|^p \mathbb{I}(X_n \leq B) \to 0.$$

This convergence happens for any fixed $B > 0$. So it suffices to show that $\sup_n \mathbb{E}\,|X_n|^p \mathbb{I}(|X_n|^p > B)$ becomes arbitrarily small as $B \to \infty$.

By Markov's Inequality, we have

$$\Pr[|X_n|^p > t] = \Pr[|X_n|^q > t^{q/p}] \leq \frac{\mathbb{E}\,|X_n|^q}{t^{q/p}}$$

By Lemma A.12.1,

$$\begin{aligned}\mathbb{E}\,|X_n|^p \mathbb{I}(|X_n|^p > B) &= B \Pr[|X_n|^p > B] + \int_B^\infty \Pr[|X_n|^p > t]dt \\ &\leq \frac{\mathbb{E}\,|X_n|^q}{B^{q/p-1}} + \int_B^\infty \frac{\mathbb{E}\,|X_n|^q}{t^{q/p}} dt.\end{aligned}$$

Since $\mathbb{E}\,|X_n|^q < C$ and $q/p > 1$ by assumption, this quantity goes to 0 as $B \to \infty$, as desired. □

Theorem A.12.3. *Consider a program in Setup A.9.2 and a scalar c in the program. If c converges almost surely to a limit $\mathring{c}$, then c converges in L^p to $\mathring{c}$ as well for every $p \in [1, \infty)$.*

Proof. This follows from Lemma A.12.2 and Lemma A.9.6. □

A.13 Empirical Validation

In the present section, we validate our main result, Theorem 3.3.7, by empirically checking some of its corollaries mentioned in Section 3.4.

A.13.1 On Non-Gaussian Biases, Input, and Output layers

Here we first note a very important subtlety concerning distribution universality. When expressing neural network computations (i.e. forward and backward passes) as a tensor program, we use matrices for initial hidden weights and vectors for biases and initial input and output weights. Indeed, since for input and output layers, only one dimension grows to infinity, they cannot be expressed as $n \times n$ matrices in the program for which both dimensions go to infinity. Instead, they are expressed as a set of vectors.

As noted in Section 3.4, we model non-Gaussian vector variables as images of Gaussian ones under elementwise nonlinear maps. This requires adding a nonlinearity to a program, which may alter the limit in Theorem 3.3.7.

This means that the NNGP and NTK kernels of Corollaries 3.4.3 and 3.4.4 are not necessarily distribution-invariant. However both corollaries are still true as they merely state that these kernels exist.

A.13.2 Setup

We perform our experiments with vanilla RNN, vanilla GRU network, and a simple Transformer. We build on the code acompanying Yang (2019b) and Yang (2020a).

As for experiments aimed to validate NNGP correspondence, Corollary 3.4.3, we compute correlation matrices on two sentences, "The brown fox jumps over the dog" and "The quick brown fox jumps over the lazy dog", embedded using GloVe embeddings Pennington et al. (2014). We compute the empirical kernel for different distributions and measure the relative Frobenius norm distance between this empirical kernel and the analytic kernel for Gaussian initialization. We compute mean and standard deviations for this relative Frobenius norm distance using 100 random initializations.

As for experiments aimed to validate convergence to a kernel method, Corollary 3.4.4, we compute empirical kernels on two "sentences", the first 5 pixels of the first CIFAR10 image and the first 5 pixels of the second CIFAR10 image. We compute it for different distributions and measure the relative Frobenius norm distance between this empirical kernel and the analytic neural tangent kernel. Same as for NNGP correspondence, we compute mean and standard deviations for this relative Frobenius norm distance using 100 random initializations.

Following Setup 3.3.6, we consider distributions with zero mean and variance $1/n$. The distributions we consider are Gaussian, Gaussian truncated at 2σ, uniform, Laplace, and the other one which we call *Cubecauchy*. The Cubecauchy distribution is a distribution of a cubic root of a Cauchy random variable. Such a random variable has finite mean and variance, but does not have any other moments. Since it does not have all moments, it does not follow Setup 3.3.6, which makes our Theorem 3.3.7 inapplicable. However, as we shall see shortly, the empirical kernels we consider still converge to the same limit suggesting that existence of all moments is not necessary.

A.13.3 Results

We start with empirically validating NNGP correspondence, Corollary 3.4.3. We first swap only hidden weights with their non-Gaussian counterparts. In this case, our Theorem 3.3.7 predicts that the limit should not depend on matrix entry distributions. As we see in Fig. A.1, this is indeed the case. Moreover, the empirical kernel that corresponds to the Cubecauchy distribution still converges to the same limit as for the other distributions, thus suggesting that coincidence of only the first

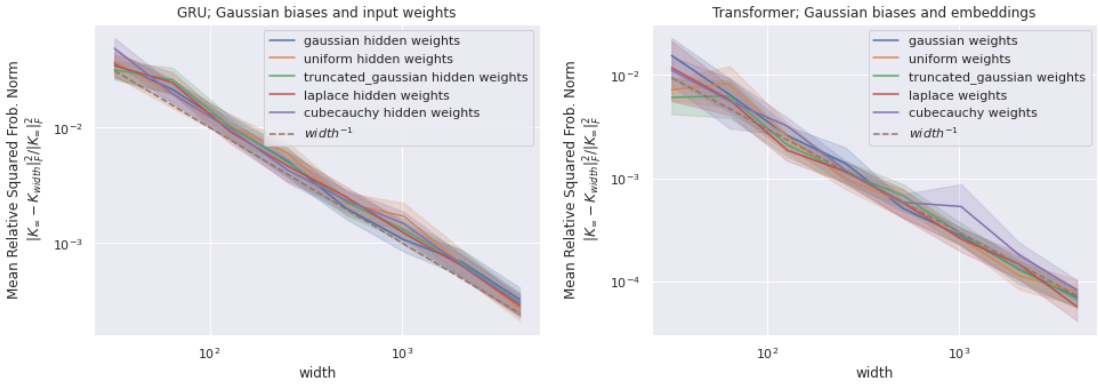


Figure A.1: Relative Frobenius norm distance between empirical NNGP kernels for different distributions and the analytic kernel for Gaussian initialization. All trainable parameters expressed with vectors (i.e. biases, embeddings, and input weights) are assumed to be Gaussian. Left: a simple GRU network, right: a simple transformer, see Appendix A.13.2. As we see, NNGP kernels for all distributions considered converge to the same analytic kernel.

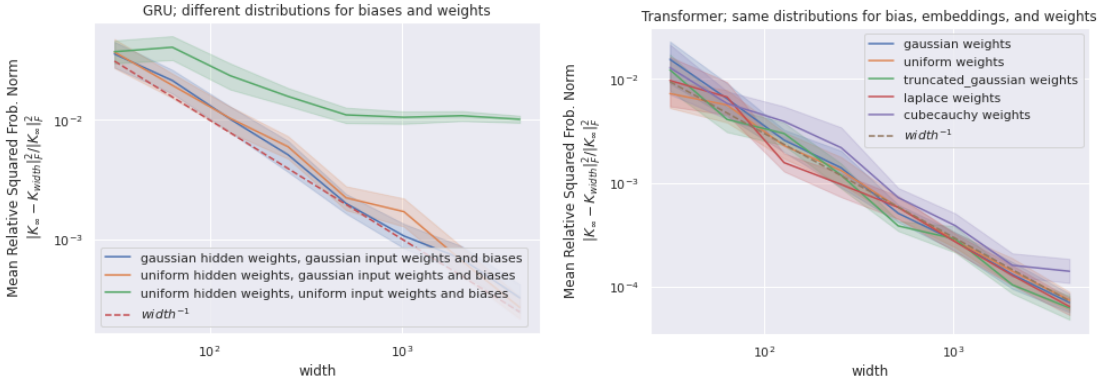


Figure A.2: Relative Frobenius norm distance between empirical NNGP kernels for different distributions and the analytic kernel for Gaussian initialization. We assume all trainable parameters to have the same distribution. Left: a simple GRU network, right: a simple transformer, see Appendix A.13.2. As we see, in some cases (left, green line), the limit NNGP kernel may depend on the parameter distribution since we had to modify the corresponding tensor program in order to model non-Gaussian vector variables, see Appendix A.13.1. Note that the limit kernel still exists and there is no contradiction with Corollary 3.4.3.

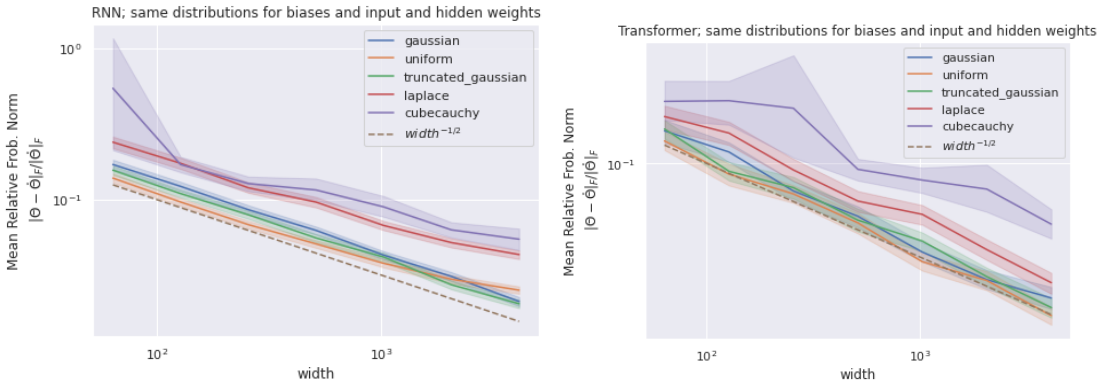


Figure A.3: Relative Frobenius norm distance between empirical neural tangent kernels for different distributions and the analytical neural tangent kernel for Gaussian initialization. We assume all trainable parameters to have the same distribution. Left: a simple recurrent neural network, right: a simple transformer, see Appendix A.13.2. As we see, the limit NTK does not depend on the parameter distribution, even when we use non-Gaussian vectors, see discussion in Appendix A.13.1.

two moments are necessary for the Master Theorem to hold.

What happens if we swap all other trainable parameters, i.e. biases, input weights, embeddings, with their non-Gaussian counterparts? As discussed above, the Master Theorem still holds, but it requires modifying the program itself which may alter the limit. As we observe in Fig. A.2, sometimes it is indeed the case (left plot), while sometimes the limit could be the same (right plot). We note that even when the limit is different from the one resulted from Gaussian weights, Fig. A.2 still demonstrates that the limit exists even for non-Gaussian ones, therefore validating Corollary 3.4.3.

Next, we validate neural tangent kernel convergence at initialization, checking a part of Corollary 3.4.4. As we see in Fig. A.3, even when all trainable parameters are initialized with non-Gaussians, the limit kernel still remains the same.

A.14 Empirical validation of the main theorem

In this section, we are going to check the statement of our main result, Theorem 3.3.7. Specifically, we are going to check that scalars converge and their limits depend only on first two moments of the matrix distribution but not on the distribution itself. In addition, we will check the following complementary statement: the limit does depend on the distribution of vectors.

A.14.1 Setup

Model. We consider a fully-connected network with one or two hidden layers and no biases. Note that there are no matrices in the case of one hidden layer and no initial scalars in both cases.

Dataset. We train the network on a subset of CIFAR10. We additionaly downsample the training images. This is done primarily for speeding up experiments: as we shall see further, we have to train

the network up to 10^4 times starting from different initializations in order to reliably observe the phenomena we are looking for.

Training details. We train the network with SGD with momentum equal to 0.9 and batchsize equal to $\min(\text{dataset\_size}, 100)$. We optimize cross-entropy loss on the train dataset. We do not apply any preprocessing to images apart from downsampling.

What we measure. We consider two scalars as (random) functions of width: train loss and Frobenius norm of output layer weights. Since a scalar is a random variable for any finite width, we compare its distribution for two initial weight distributions: a Gaussian and a non-Gaussian we consider. In order to compare the two, we measure a Wasserstein distance between their empirical distributions.

We vary the width from $2^5 = 32$ to $2^{13} = 8192$ in logarithmic scale. The choice of maximal width was dictated by our hardware restrictions: each experiment was done on a single NVidia A100 GPU.

A.14.2 Results

Below is a brief summary of our results:

- When all weights are initialized from the same distribution:
  1. As width increases, scalars converge to limits that do depend on the distribution;
  2. Scalars resulted from different initialization distributions are easier to distinguish later during training: we need smaller width to do it reliably;
  3. Scalars resulted from non-symmetric initialization distributions are easier to distinguish from ones resulted from a Gaussian, compared to scalars resulted from symmetric initialization distributions.
- When only matrix-like weights are initialized from the same distribution, while all other weights are initialized as Gaussians:
  1. When the initialization distribution for matrix-like weights has at least four moments, scalars converge as width increases and the limit does not depend on the distribution;
  2. A Wasserstein distance between a scalar resulted from a given distribution and the one resulted from a Gaussian follows a clear power-law when the given distribution is non-symmetric, while the power-law is less clear for symmetric distributions;
  3. When the initalization distribution for matrix-like weights has only two moments, it is not clear whether scalars converge or not, but if they do, the limit is different from the ones resulted from a Gaussian initialization.
- For both cases:
  1. In order to estimate the Wasserstein distance between scalars resulted from different initialization distributions, one may need around 10^4 random weight initializations;
  2. For datasets of realistic sizes (e.g. CIFAR10), scalars may require huge width ($\gg 2^{13}$) to converge.

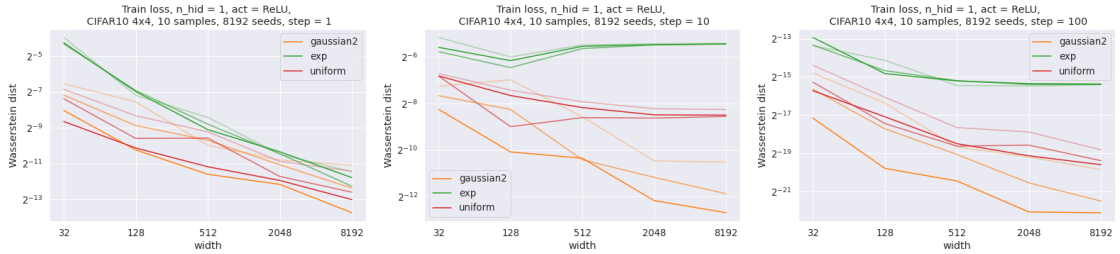


Figure A.4: A fully-connected network with 1 hidden layer trained on a subset of CIFAR10 downsampled to 4x4 image size. Train loss.

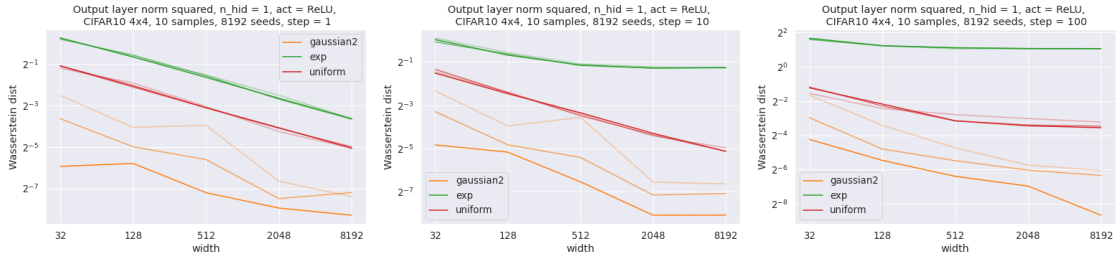


Figure A.5: A fully-connected network with 1 hidden layer trained on a subset of CIFAR10 downsampled to 4x4 image size. Frobenius norm of the readout layer.

One hidden layer

We start with the case of one hidden layer. As there are no matrices in this case, we expect the infinite-width limit to depend on the weight distribution.

We compare uniform and exponential initial weight distributions, both with zero mean and unit variance, with a Gaussian one. As we see in Figs. A.4 and A.5, the Wasserstein distance decays as a power-law early in the training. Nevertheless, both curves reach a non-zero asymptote for epoch=100. This indicates that the limit does depend on the distribution. Furthermore, the plato starts earlier later during training. This suggests that we simply do not consider large enough width to reach the plato in the early phase of training.

We conjecture that the this happens because the divergence between a Gaussian and a non-Gaussian accumulates at each training step. At the early phase, the divergence caused by the initial weight distribution difference is small compared to the divergence caused by finite width. However, as training progresses, the former increases faster than the latter.

Another phenomenon we observe here is that the exponential initial weight distribution results in larger divergence from the Gaussian one and an earlier plato start. We conjecture that the reason could lie in the difference in the third moment: it equals zero for both Gaussian and uniform distributions, while it does not for the exponential one.

Since we compare empirical distributions, we have to check that we use a sufficient number of random initializations to estimate the Wasserstein distance reliably. In order to do this, we plot three curves instead of one for each distribution, each indicating a different number of random initializations we used to estimate the distance: N, $N/2$, and $N/4$, starting from the solid one to

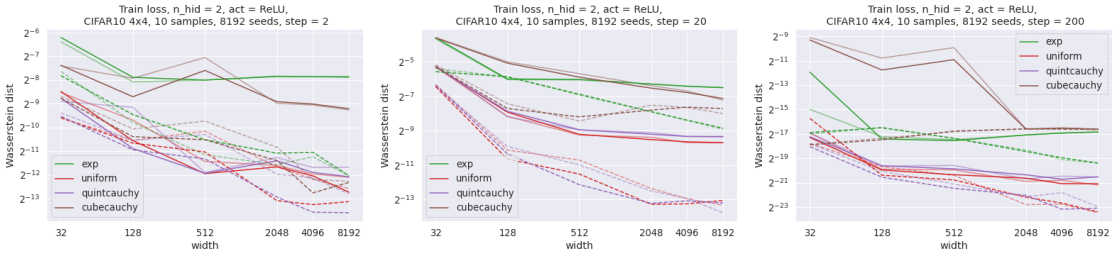


Figure A.6: A fully-connected network with 2 hidden layers trained on a subset of CIFAR10 downsampled to 4x4 image size. Train loss.

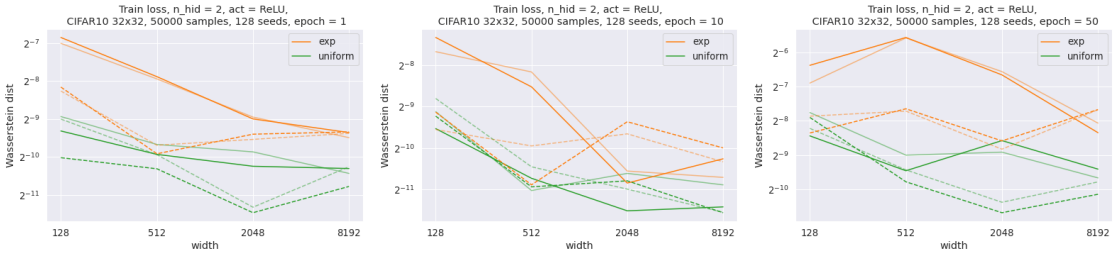


Figure A.7: A fully-connected network with 2 hidden layers trained on the full CIFAR10 without any downsampling. Train loss.

the most faint. As we see, for the exponential distribution, $N = 8192$ is enough to get a reliable estimate since N and $N/2$ give almost the same curve, however, it might be barely enough for the uniform one.

Comparing the Frobenius norm of the readout layer, Fig. A.5, and the train loss, Fig. A.4, we notice that the former requires fewer random initializations to converge. The reason may lie in the fact that train loss tends to zero as training progresses, thus making different distributions more similar, hence asking for more random intializations to track the difference.

Apart from exponential and uniform distributions, we also plot the Wasserstein distance between a Gaussian and another independently sampled Gaussian (orange curves). As we use only a finite number of random initializations to estimate the distance, these curves are not zero; nevertheless, we expect them to vanish as the number of random intializations grows. While they indeed decay with this number, they are still comparable with the curves for the uniform distribution when the number of random initializations used to compute the distance is small. This indicates that even as many as $N/4 = 1024$ random initializations might be not enough to reliably distinguish the train loss of a Gaussian intialization from that of the uniform one, especially when the width is small.

Two hidden layers

We consider a network with two hidden layers in this section. Weights of the second layer are represented with a matrix. In Fig. A.6, solid lines correspond to swapping all weights with a prescribed distribution, while dashed lines correspond to initializing only matrix-like weights from a prescribed distribution, while initializing all other weights as Gaussians.

We consider the following distributions: exponential, uniform, and two others that we call "cubeCauchy" and "quintCauchy". The former is a distribution of a cubic root of a Cauchy-distributed random variable, while the latter is the same but a quintic root. We normalize both distributions (together with the two others) to have zero mean and unit variance. The idea of considering these two is two check if the existence of all moments is necessary for Theorem 3.3.7 to hold: the quintCauchy distribution has four moments, while the cubeCauchy one has only two. According to the experiments, our theorem should work when four moments exist but does not hold when there are only two. This resembles a classical random matrix result of Theorem A.3.5 we used to prove Theorem A.3.2.

As for uniform and exponential distributions, we observe qualitatively the same picture as in the one hidden layer case when we make all weights non-Gaussian (solid curves): the exponential distribution results in an earlier plato, while the plato starts earlier later during the training for all distributions. The quintCauchy curves show qualitatively the same behavior as the uniform curves, while the cubeCauchy curves have much larger values. We hypothesize that the difference is due to the third moment: it equals zero for the uniform and quintCauchy distributions (since they are symmetric), while it does not for the exponential and cubeCauchy distributions.

When we swap only matrix-like weights (dashed curves), all curves decay as width grows except for the cubeCauchy one. This suggests that having four moments might be enough for Theorem 3.3.7 to work but having only two moments is not enough. Although not stated in our theorem, we expect the curves to decay as power-laws for large enough width. We do observe a clear power-law for the exponential distributions, while for others the power-law is less clear; the reason could be the lack of random initializations or width being too small.

Larger image dimensions and dataset sizes. For the experiments in Fig. A.6, we used a subset of only 10 images of CIFAR10 downsampled to 4x4; in this case, we could train the network up to convergence with enough number of random initializations in a reasonable time (around two weeks on a single NVidia A100 GPU). Taking the full CIFAR10 without any downsampling increases the training time both in terms of the number of steps required for the network to converge, and in terms of the time needed for a single gradient descent step. For this reason, we could do only 50 training epochs and 128 random initializations in a reasonable amount of time in this setup. As we see in Fig. A.7, the width we consider (up to 8192 neurons) is not enough for the Wasserstein distance to reach a plato or a power-law. As we noted above, the distance probably platos with smaller width later during the training, but we did not have enough computational resource to check this conjecture.

B Appendix: Tail bounds for Tensor Programs

B.1 Missing proofs

B.1.1 Corollary 4.2.4

Corollary 4.2.4.

$$\forall n \in \mathbb{N},\ \tau > 0 \qquad \mathcal{P}(c^1 - \mathring{c}^1 > \tau) \leq e^{-C_2^1 n\tau^2} + ne^{-C_0 n}, \tag{4.17}$$

and we have the same bound for $\mathcal{P}(c^1 - \mathring{c}^1 < -\tau)$.

Proof. Let us compute κ:

$$\begin{aligned}\mathcal{P}(x_\beta > t) &\leq \mathcal{P}\left(\lambda^2(\nu^1 + |g_\beta^1|)|g_\beta^1| > t\right)\\ &= \mathcal{P}\left(\lambda^2|g_\beta^1|^2 + \lambda^2\nu^1|g_\beta^1| - t > 0\right) = \mathcal{P}\left(|g_\beta^1| > \tfrac{-\lambda^2\nu^1 + \sqrt{\lambda^4(\nu^1)^2 + 4t\lambda^2}}{2\lambda^2}\right)\\ &= \mathcal{P}\left(|g_\beta^1| > \tfrac{\nu^1}{2}\left(\sqrt{1 + \tfrac{4t}{\lambda^2(\nu^1)^2}} - 1\right)\right) = \operatorname{erfc}\left(\tfrac{\nu^1}{2\sqrt{2}}\left(\sqrt{1 + \tfrac{4t}{\lambda^2(\nu^1)^2}} - 1\right)\right).\end{aligned} \tag{B.1}$$

In order to upper-bound the erfc with an exponent of the form $e^{-\kappa t}$, we use the following result we prove later in this section:

Lemma B.1.1. $\forall x, a \geq 0$ $\operatorname{erfc}\left(a(\sqrt{1+x} - 1)\right) \leq e^{-\frac{\theta(a)x}{4}}$, *where* $\theta(a) = \frac{4a^2}{a\sqrt{\pi}+2}$.

Applying this lemma gives

$$\mathcal{P}(x_\beta > t) = \operatorname{erfc}\left(\tfrac{\nu^1}{2\sqrt{2}}\left(\sqrt{1 + \tfrac{4t}{\lambda^2(\nu^1)^2}} - 1\right)\right) \leq e^{-\theta\left(\frac{\nu^1}{2\sqrt{2}}\right)\frac{t}{\lambda^2(\nu^1)^2}}, \tag{B.2}$$

which implies $\kappa = \frac{\theta(\nu^1/\sqrt{8})}{\lambda^2(\nu^1)^2} = \frac{1}{\lambda^2\left(4 + \nu^1\sqrt{\frac{\pi}{2}}\right)}$.

We then bound the moments:

$$\mathbb{E}\left[|x|\right] \leq \mathbb{E}\left[\lambda^2\left(\nu^1 + |g_\beta^1|\right)|g_\beta^1|\right] \leq \lambda^2\left(1 + \nu^1\sqrt{\frac{2}{\pi}}\right); \tag{B.3}$$

$$\mathbb{E}\left[x^2\right] \leq \mathbb{E}\left[\lambda^4\left((\nu^1)^2 + 2\nu^1|g^1_\beta| + |g^1_\beta|^2\right)|g^1_\beta|^2\right] \leq \lambda^4\left(3 + 4\sqrt{\frac{2}{\pi}}\nu^1 + (\nu^1)^2\right). \tag{B.4}$$

Finally, we bound quantities containing q:

$$\begin{aligned} q \leq \mathbb{E}\left[x^2\right] + \tfrac{16}{\kappa^2 e^{-\kappa\mathbb{E}|x|}} &\leq \lambda^4\left(\left(3 + 4\sqrt{\frac{2}{\pi}}\nu^1 + (\nu^1)^2\right) + 16\left(4 + \nu^1\sqrt{\frac{\pi}{2}}\right)^2 e^{\frac{1+\nu^1\sqrt{\frac{2}{\pi}}}{4+\nu^1\sqrt{\frac{\pi}{2}}}}\right) \\ &\leq \lambda^4\left(\left(3 + 256e^{\frac{2}{\pi}}\right) + \left(4\sqrt{\frac{2}{\pi}} + 64\sqrt{2\pi}e^{\frac{2}{\pi}}\right)\nu^1 + (1 + 8\pi e^{\frac{2}{\pi}})(\nu^1)^2\right); \end{aligned} \tag{B.5}$$

$$\frac{q\kappa^2}{2} \geq 8e^{-\kappa\mathbb{E}\left[x\right]} \geq 8e^{-\frac{1+\nu^1\sqrt{\frac{2}{\pi}}}{4+\nu^1\sqrt{\frac{\pi}{2}}}} \geq 8e^{-\frac{2}{\pi}}; \tag{B.6}$$

$$\frac{2\mathbb{E}\left[x\right]}{q\kappa} \leq \frac{1}{8}\kappa\mathbb{E}\left[x\right]e^{\kappa\mathbb{E}\left[x\right]} \leq \frac{1}{8}\frac{1 + \nu^1\sqrt{\frac{2}{\pi}}}{4 + \nu^1\sqrt{\frac{\pi}{2}}}e^{\frac{1+\nu^1\sqrt{\frac{2}{\pi}}}{4+\nu^1\sqrt{\frac{\pi}{2}}}} \leq \frac{e^{\frac{2}{\pi}}}{4\pi} < 1. \tag{B.7}$$

Plugging all of the above into Lemma 4.2.3, and noting that $\mathring{c}^1 = \mathbb{E}\left[c^1\right]$ by LLN, gives the result. □

We now prove Lemma B.1.1:

Proof. Since at $x = 0$ both sides equal 1, we have to check the same inequality for derivatives of logarithms:

$$\frac{d\left(\ln\operatorname{erfc}\left(a(\sqrt{1+x}-1)\right)\right)}{dx} = -\frac{ae^{-a^2(\sqrt{1+x}-1)^2}}{\sqrt{\pi}\sqrt{1+x}\operatorname{erfc}\left(a(\sqrt{1+x}-1)\right)} \leq -\frac{a}{\sqrt{\pi}\sqrt{1+x}} \leq -\frac{a}{\sqrt{\pi}\sqrt{1+c}} \tag{B.8}$$

for all $x \in [0, c]$ for some $c > 0$ to be determined later. This gives

$$\operatorname{erfc}\left(a(\sqrt{1+x}-1)\right) \leq e^{-\frac{ax}{\sqrt{(1+c)\pi}}} \qquad \forall x \in [0, c],\ a \geq 0. \tag{B.9}$$

On the other hand,

$$\sqrt{1+x} - 1 \geq \frac{\sqrt{1+c}-1}{\sqrt{c}}\sqrt{x} \qquad \forall x \geq c, \tag{B.10}$$

which implies

$$\operatorname{erfc}\left(a(\sqrt{1+x}-1)\right) \leq \operatorname{erfc}\left(a\frac{\sqrt{1+c}-1}{\sqrt{c}}\sqrt{x}\right) \leq e^{-a^2\frac{(\sqrt{1+c}-1)^2}{c}x} \qquad \forall x \geq c,\ a \geq 0. \tag{B.11}$$

The optimal c is given by equalizing the exponents for both cases:

$$\frac{a}{\sqrt{(1+c)\pi}} = a^2\frac{(\sqrt{1+c}-1)^2}{c} = a^2\frac{\sqrt{1+c}-1}{\sqrt{1+c}+1}. \tag{B.12}$$

Denote $\sqrt{1+c}$ as y. Then we get a quadratic equation for y:

$$a\sqrt{\pi}y = \frac{y+1}{y-1}; \tag{B.13}$$

$$a\sqrt{\pi}y^2 - (a\sqrt{\pi}+1)y - 1 = 0; \tag{B.14}$$

$$y = \frac{1 + a\sqrt{\pi} + \sqrt{(1+a\sqrt{\pi})^2 + 4a\sqrt{\pi}}}{2a\sqrt{\pi}}. \tag{B.15}$$

Therefore

$$\operatorname{erfc}\left(a(\sqrt{1+x}-1)\right) \le e^{-\frac{2a^2x}{1+a\sqrt{\pi}+\sqrt{1+6a\sqrt{\pi}+a^2\pi}}} \qquad \forall x, a \ge 0. \tag{B.16}$$

To simplify the further analysis we use

$$1 + b + \sqrt{1+6b+b^2} \le 2b + 4 \qquad \forall b \ge 0. \tag{B.17}$$

This gives the final bound:

$$\operatorname{erfc}\left(a(\sqrt{1+x}-1)\right) \le e^{-\frac{a^2x}{a\sqrt{\pi}+2}} \qquad \forall x, a \ge 0. \tag{B.18}$$

□

B.1.2 Lemma 4.2.5

Lemma 4.2.5. *Suppose $(x_\beta)_{\beta=1}^n$ are independent RVs, and $\forall \beta \in [n]$ $\mathcal{P}(x_\beta > t) \le e^{-\kappa_\beta t}$ for some $\kappa_\beta > 0$. Define $c = \frac{1}{n}\sum_\beta x_\beta$ and*

$$q_\beta = \mathbb{E}[x_\beta^2] + \frac{16}{\kappa_\beta^2 e^{\kappa_\beta \mathbb{E}[x_\beta]}}. \tag{4.18}$$

Then $\forall n > 2\max_\beta\left(\frac{\mathbb{E}[x_\beta]}{q_\beta\kappa_\beta}\right)$

$$\mathcal{P}(c - \mathbb{E}[c] > t) \le e^{-\frac{nt^2}{\frac{2}{n}\sum_\beta q_\beta}} + ne^{-\frac{n\min_\beta(q_\beta\kappa_\beta^2)}{2}}. \tag{4.19}$$

Proof. We closely follow the proof of Theorem 1 of Bakhshizadeh et al. (2023). If x is a RV, let $x^L = \min(x, L)$. Take some $(L_\beta)_{\beta=1}^n$ such that $L_\beta > \mathbb{E}[x_\beta]$ $\forall \beta \in [n]$. Applying Markov's inequality and Lemma 1 of Bakhshizadeh et al. (2023), we get

$$\begin{aligned}
\mathcal{P}(c - \mathbb{E}[c] > t) &\le \mathcal{P}\left(\sum_\beta \left(x_\beta^{L_\beta} - \mathbb{E}[x_\beta]\right) > nt\right) + \mathcal{P}(\exists\beta :\ x_\beta > L_\beta) \\
&\le \frac{\mathbb{E}\left[\prod_\beta e^{\lambda(x_\beta^{L_\beta} - \mathbb{E}[x_\beta])}\right]}{e^{\lambda nt}} + \sum_\beta \mathcal{P}(x_\beta > L_\beta) \\
&\le \frac{\prod_\beta e^{\frac{\lambda^2}{2}q_\beta}}{e^{\lambda nt}} + \sum_\beta e^{-\kappa_\beta L_\beta} \le e^{\frac{\lambda^2}{2}\sum_\beta q_\beta - \lambda nt} + \sum_\beta e^{-\kappa_\beta L_\beta}.
\end{aligned} \tag{B.19}$$

Take $\lambda = \frac{nt}{\sum_\beta q_\beta}$ and $L_\beta = \frac{n}{2}q_\beta\kappa_\beta$. Then $L_\beta > \mathbb{E}[x_\beta]$ is implied by $n > 2\frac{\mathbb{E}[x_\beta]}{q_\beta\kappa_\beta}$. □

B.1.3 Corollary 4.2.6

Corollary 4.2.6. *Let* $\mathring{c}^{l+1}(g^{1:l}) = \frac{1}{n}\sum_\beta \mathbb{E}_{z\sim\mathcal{N}(0,1)}\psi(g_\beta^{1:l}, \sqrt{v^l}z)$, $d^l = \frac{1}{n}\sum_\beta \left(\nu^{l+1} + 2\sum_{j=1}^l |g_\beta^j|\right)^2$. *Then*

$$\forall n \in \mathbb{N},\ \tau > 0 \qquad \mathcal{P}\left(c^{l+1} - \mathring{c}^{l+1}(g^{1:l}) > \tau \,\middle|\, g^{1:l}\right) \le e^{-\hat{C}_2^l n\tau^2} + ne^{-C_0 n}, \tag{4.20}$$

where

$$\hat{C}_2^l = \frac{1}{2\lambda^4\left((3+256e^{\frac{2}{\pi}})(v^l)^2 + \left(2+32\pi e^{\frac{2}{\pi}}\right)\sqrt{\frac{(2v^l)^3 d^l}{\pi}} + (1+8\pi e^{\frac{2}{\pi}})v^l d^l\right)}, \tag{4.21}$$

and we have the same bound for $\mathcal{P}(c^{l+1} - \mathring{c}^{l+1}(g^{1:l}) < -\tau)$.

In order to prove the corrolary, we will need the following lemma. Consider the following function acting on $\mathbb{R}_+$:

$$h^l(z) = \frac{\theta\left(\frac{z}{\sqrt{8v^l}}\right)\left(v^l + \sqrt{\frac{2v^l}{\pi}}z\right)}{z^2}. \tag{B.20}$$

The following is proven in Appendix B.1:

Lemma B.1.2. *The function* $h^l(z)$ *is monotonically increasing on* $\mathbb{R}_+$ *and bounded by* $\frac{2}{\pi}$.

Proof. We are going to show that $(h^l)'(z)$ is positive for positive z and vanishing as $z \to +\infty$. Let us compute the derivative:

$$\begin{aligned}(h^l)'(z) &= \frac{d}{dz}\left[\theta\left(\frac{z}{\sqrt{8v^l}}\right)\left(v^l z^{-2} + \sqrt{\frac{2v^l}{\pi}}z^{-1}\right)\right] \\ &= \frac{1}{\sqrt{8v^l}}\theta'\left(\frac{z}{\sqrt{8v^l}}\right)\left(v^l z^{-2} + \sqrt{\frac{2v^l}{\pi}}z^{-1}\right) - \theta\left(\frac{z}{\sqrt{8v^l}}\right)\left(2v^l z^{-3} + \sqrt{\frac{2v^l}{\pi}}z^{-2}\right).\end{aligned} \tag{B.21}$$

Recall

$$\theta(x) = \frac{4x^2}{x\sqrt{\pi}+2}. \tag{B.22}$$

This gives

$$\theta'(x) = \frac{8x(x\sqrt{\pi}+2) - 4\sqrt{\pi}x^2}{(x\sqrt{\pi}+2)^2} = \frac{4x(x\sqrt{\pi}+4)}{(x\sqrt{\pi}+2)^2}. \tag{B.23}$$

We see that $\lim_{z\to\infty}(h^l)'(z) = 0$. This derivative is also positively proportional to

$$\begin{aligned}(h^l)'(z) &\propto \frac{x\sqrt{\pi}+4}{\sqrt{8v^l}}\left(v^l z + \sqrt{\frac{2v^l}{\pi}}z^2\right) - x(x\sqrt{\pi}+2)\left(2v^l + \sqrt{\frac{2v^l}{\pi}}z\right)\Bigg|_{x=\frac{z}{\sqrt{8v^l}}} \\ &= \left(\sqrt{\pi}\frac{z}{8v^l} + \frac{4}{\sqrt{8v^l}}\right)\left(v^l z + \sqrt{\frac{2v^l}{\pi}}z^2\right) - \left(\sqrt{\pi}\frac{z^2}{8v^l} + \frac{2z}{\sqrt{8v^l}}\right)\left(2v^l + \sqrt{\frac{2v^l}{\pi}}z\right) \\ &= \left(\frac{1}{\sqrt{\pi}} - \frac{\sqrt{\pi}}{8}\right)z^2,\end{aligned} \tag{B.24}$$

which is positive $\forall z > 0$. This means that $h^l(z) < \lim_{z\to+\infty} h^l(z) = \frac{2}{\pi}$. □

We are now ready to prove Corollary 4.2.6:

Proof. We again use Lemma B.1.1 in order to compute κ_β:

$$\begin{aligned}
\mathcal{P}(x_\beta > t) &\leq \mathcal{P}\left(\lambda^2(\nu^{l+1} + 2\|g_\beta^{1:l}\|_1 + \sqrt{v^l}|z|)\sqrt{v^l}|z| > t\right) \\
&= \mathcal{P}\left(\lambda^2 v^l|z|^2 + \lambda^2(\nu^{l+1} + 2\|g_\beta^{1:l}\|_1)\sqrt{v^l}|z| - t > 0\right) \\
&= \mathcal{P}\left(|z| > \frac{-\lambda^2(\nu^{l+1}+2\|g_\beta^{1:l}\|_1)\sqrt{v^l} + \sqrt{\lambda^4(\nu^{l+1}+2\|g_\beta^{1:l}\|_1)^2 v^l + 4t\lambda^2 v^l}}{2\lambda^2 v^l}\right) \\
&= \mathcal{P}\left(|z| > \frac{\nu^{l+1}+2\|g_\beta^{1:l}\|_1}{2\sqrt{v^l}}\left(\sqrt{1 + \frac{4t}{\lambda^2(\nu^{l+1}+2\|g_\beta^{1:l}\|_1)^2}} - 1\right)\right) \\
&= \operatorname{erfc}\left(\frac{\nu^{l+1}+2\|g_\beta^{1:l}\|_1}{2\sqrt{2v^l}}\left(\sqrt{1 + \frac{4t}{\lambda^2(\nu^{l+1}+2\|g_\beta^{1:l}\|_1)^2}} - 1\right)\right) \\
&\leq e^{-\theta\left(\frac{\nu^{l+1}+2\|g_\beta^{1:l}\|_1}{2\sqrt{2v^l}}\right)\frac{t}{\lambda^2(\nu^{l+1}+2\|g_\beta^{1:l}\|_1)^2}}.
\end{aligned} \tag{B.25}$$

which gives $\kappa_\beta = \frac{1}{\lambda^2\eta_\beta^l}$, where $\eta_\beta^l = \frac{(\nu^{l+1}+2\|g_\beta^{1:l}\|_1)^2}{\theta\left(\frac{\nu^{l+1}+2\|g_\beta^{1:l}\|_1}{2\sqrt{2v^l}}\right)}$. We then bound the moments:

$$\mathbb{E}\left[|x_\beta|\right] \leq \mathbb{E}\left[\lambda^2\left(\nu^{l+1} + 2\|g_\beta^{1:l}\|_1 + \sqrt{v^l}|z|\right)\sqrt{v^l}|z|\right] \leq \lambda^2\left(v^l + \sqrt{\frac{2v^l}{\pi}}\left(\nu^{l+1} + 2\|g_\beta^{1:l}\|_1\right)\right); \tag{B.26}$$

$$\begin{aligned}
\mathbb{E}\left[x_\beta^2\right] &\leq \mathbb{E}\left[\lambda^4\left(\nu^{l+1} + 2\|g_\beta^{1:l}\|_1 + \sqrt{v^l}|z|\right)^2 v^l z^2\right] \\
&\leq \lambda^4\left(3(v^l)^2 + 2\sqrt{\frac{(2v^l)^3}{\pi}}\left(\nu^{l+1} + 2\|g_\beta^{1:l}\|_1\right) + v^l\left(\nu^{l+1} + 2\|g_\beta^{1:l}\|_1\right)^2\right).
\end{aligned} \tag{B.27}$$

Finally, we bound quantities containing q_β:

$$\begin{aligned}
q_\beta &\leq \mathbb{E}\left[x_\beta^2\right] + \frac{16}{\kappa_\beta^2 e^{-\kappa_\beta \mathbb{E}\left[|x_\beta|\right]}} \\
&\leq \lambda^4\Bigg(3(v^l)^2 + 2\sqrt{\frac{(2v^l)^3}{\pi}}\left(\nu^{l+1} + 2\|g_\beta^{1:l}\|_1\right) \\
&\quad + v^l\left(\nu^{l+1} + 2\|g_\beta^{1:l}\|_1\right)^2 + 16(\eta_\beta^l)^2 e^{\frac{v^l + \sqrt{\frac{2v^l}{\pi}}\left(\nu^{l+1}+2\|g_\beta^{1:l}\|_1\right)}{\eta_\beta^l}}\Bigg).
\end{aligned} \tag{B.28}$$

By Lemma B.1.2, $\frac{v^l + \sqrt{\frac{2v^l}{\pi}}\left(\nu^{l+1}+2\|g_\beta^{1:l}\|_1\right)}{\eta_\beta^l} = h(\nu^{l+1} + 2\|g_\beta^{1:l}\|_1) \leq \frac{2}{\pi}$, which implies

$$q_\beta \leq \lambda^4\left(3(v^l)^2 + 2\sqrt{\frac{(2v^l)^3}{\pi}}\left(\nu^{l+1} + 2\|g_\beta^{1:l}\|_1\right) + v^l\left(\nu^{l+1} + 2\|g_\beta^{1:l}\|_1\right)^2 + 16(\eta_\beta^l)^2 e^{\frac{2}{\pi}}\right). \tag{B.29}$$

There are some calculations we will need to compute the average of q_β:

$$f(x,y) := \frac{x^4}{\theta^2\left(\frac{|x|}{\sqrt{8|y|}}\right)} = y^2\left(\sqrt{\frac{\pi}{2|y|}}|x| + 4\right)^2 = |y|\left(\frac{\pi}{2}x^2 + 16|y| + 8\sqrt{\frac{\pi}{2}|y||x|}\right). \tag{B.30}$$

Therefore

$$\frac{1}{n}\sum_\beta (\eta_\beta^l)^2 = \frac{1}{n}\sum_\beta f(\nu^{l+1} + 2\|g_\beta^{1:l}\|_1, v^l) = \frac{\pi}{2}v^l d^l + 16(v^l)^2 + 8\sqrt{\frac{\pi}{2}(v^l)^3 d^l}, \tag{B.31}$$

where we used the bound $\frac{1}{n}\sum_\beta\left(\nu^{l+1} + 2\|g_\beta^{1:l}\|_1\right) \le \sqrt{\frac{1}{n}\sum_\beta\left(\nu^{l+1} + 2\|g_\beta^{1:l}\|_1\right)^2} = \sqrt{d^l}$. Hence

$$\frac{1}{n}\sum_\beta q_\beta \le \lambda^4\left((3 + 256e^{\frac{2}{\pi}})(v^l)^2 + \left(2 + 32\pi e^{\frac{2}{\pi}}\right)\sqrt{\frac{(2v^l)^3 d^l}{\pi}} + (1 + 8\pi e^{\frac{2}{\pi}})v^l d^l\right). \tag{B.32}$$

$$\frac{q_\beta \kappa_\beta^2}{2} \ge 8e^{-\kappa_\beta \mathbb{E}[x_\beta]} \ge 8e^{-\frac{1}{\eta_\beta^l}\left(v^l + \sqrt{\frac{2v^l}{\pi}}\left(\nu^{l+1} + 2\|g_\beta^{1:l}\|_1\right)\right)} = 8e^{-h^l\left(\nu^{l+1} + 2\|g_\beta^{1:l}\|_1\right)} \ge 8e^{-\frac{2}{\pi}}; \tag{B.33}$$

$$\frac{2\mathbb{E}[x]}{q\kappa} \le \frac{1}{8}\kappa_\beta \mathbb{E}[x_\beta]e^{\kappa_\beta \mathbb{E}[x_\beta]} \le \frac{h^l\left(\nu^{l+1} + 2\|g_\beta^{1:l}\|_1\right)}{8}e^{h^l\left(\nu^{l+1} + 2\|g_\beta^{1:l}\|_1\right)} = \frac{e^{\frac{2}{\pi}}}{4\pi} < 1. \tag{B.34}$$

Plugging all of the above into Lemma 4.2.5 gives the result. □

B.1.4 Lemma 4.2.7

Lemma 4.2.7. *$\forall\sigma > 0$ $\mathring{u}_\sigma^l \le \mathring{c}^{l+1} + D^l\lambda^2\sigma$, where $D^l = 1 + \sqrt{\frac{\mathring{d}^l}{2\pi\mathring{v}^l}}$.*

Proof. By pseudo-Lipschitzness,

$$\psi(z^{1:l}, \sqrt{v}z) - \psi(z^{1:l}, \sqrt{\mathring{v}^l}z) \le \lambda^2\left(\nu^{l+1} + 2\sum_{j=1}^l |z^j| + (\sqrt{v} + \sqrt{\mathring{v}^l})|z|\right)|\sqrt{v} - \sqrt{\mathring{v}^l}||z|. \tag{B.35}$$

Taking the expectation and the supremum over $v \in [\mathring{v}^l, \mathring{v}^l + \sigma]$, we get

$$\begin{aligned}
&\sup_{v\in[\mathring{v}^l, \mathring{v}^l+\sigma]} \mathbb{E}_z\left[\psi(z^{1:l}, \sqrt{v}z) - \psi(z^{1:l}, \sqrt{\mathring{v}^l}z)\right] \\
\le&\lambda^2\left(\sqrt{\tfrac{2}{\pi}}\left(\nu^{l+1} + 2\sum_{j=1}^l |z^j|\right)\left(\sqrt{\mathring{v}^l + \sigma} - \sqrt{\mathring{v}^l}\right) + \sigma\right) \\
\le&\lambda^2\left(1 + \tfrac{\nu^{l+1} + 2\sum|z^j|}{\sqrt{2\pi\mathring{v}^l}}\right)\sigma.
\end{aligned} \tag{B.36}$$

Also, by monotonicity of ψ, $\sup_{v\in[0,\mathring{v}^l]}\left[\psi(z^{1:l}, \sqrt{v}z) - \psi(z^{1:l}, \sqrt{\mathring{v}^l}z)\right] = 0$. Therefore $\psi_\sigma(z^{1:l}) - \psi_0(z^{1:l}) \le \lambda^2\left(1 + \frac{\nu^{l+1} + 2\sum|z^j|}{\sqrt{2\pi\mathring{v}^l}}\right)\sigma$. Evaluating this at $g_\beta^{1:l}$, taking average over β, and taking the limit gives $\mathring{u}_\sigma^l - \mathring{c}^{l+1} \le \lambda^2\left(1 + \sqrt{\frac{\mathring{d}^l}{2\pi\mathring{v}^l}}\right)\sigma$. □

C Appendix: Feature learning in L_2-regularized DNNs: Attraction/repulsion and sparsity

The appendix is structured as follows:

1. In Section C.1, we describe the Experimental setup.

2. In Section C.2, we prove Proposition 5.3.1 of the main underlying the first reformulation.

3. In Section C.3, we prove Proposition 5.3.3 for the second reformulation. We also give an example of a local minimum of the original loss which is not a local minimum in the second reformulation.

4. In Section C.4, we prove Proposition 5.4.3, 5.4.8, 5.4.9 and 5.4.10 of the main.

C.1 Experimental Setup

The experiments were done on fully-connected DNNs of depth $L = 3$ with varying widths.

We used the MNIST dataset LeCun et al. (1998) under the 'Creative Commons Attribution-Share Alike 3.0' license. For the MNIST examples we trained the networks on the multiclass cross-entropy loss with L_2-regularization.

We also used synthetic data sampled from a teacher network. The network has depth $L = 3$, widths $\mathbf{n} = (50, 10, 10, 10)$ with random Gaussian weights. The cost used was the Mean Squared Error (MSE).

For the experiments of Figure 5.1 of the main, the DNN was trained with full batch GD. For the experiments of Figure 5.2 we first trained with Adam Kingma and Ba (2015) and finished with full batch GD (GD seems to be better suited to consistently reach the bottom of the local minima, though Adam trains faster overall). For the right plot of Figure 5.2, three independent networks were trained for every width and the one with the smallest loss at the end of training was chosen (the plotted test error is that of the chosen network).

The goal of Figure 5.2 is to identify the start of the plateau, note however that we cannot guarantee that our training procedures actually approaches a global minimum. Interestingly it was easier to observe a plateau on MNIST rather than on the teacher network data, which is why we had to take the minimum over 3 trials in the teacher setting. This could be due to the change of loss (from cross entropy to the MSE) or due to the change of the data. Note that in Figure 5.2 (right), it is unclear whether the 'failed' trials , i.e. the small blue dots with a loss above the plateau even for

large widths, are stuck at local minima of the loss or if they could have reached the plateau if we had trained them longer.

The experiments each took between 1 and 4 hour on a single NVIDIA GeForce GTX 1080.

C.2 Equivalence for the first reformulation

Proposition C.2.1 (Proposition 5.3.1 of the main)**.** *The infimum of* $\mathcal{L}_\lambda(\mathbf{W}) = C(Z_L(X;\mathbf{W})) + \lambda \|\mathbf{W}\|^2$*, over the parameters* $\mathrm{W} \in \mathbb{R}^P$ *is equal to the infimum of*

$$\mathcal{L}^r_\lambda(Z_1,\dots,Z_L) = C(Z_L) + \lambda \sum_{\ell=1}^{L} \left\| Z_\ell \left(Z^\sigma_{\ell-1}\right)^+ \right\|^2_F$$

over the set $\mathcal{Z}$ *of hidden representations* $\mathbf{Z} = (Z_\ell)_{\ell=1,\dots,L}$ *such that* $Z_\ell \in \mathbb{R}^{n_\ell \times N}$, $\mathrm{Im} Z^T_{\ell+1} \subset \mathrm{Im}\left(Z^\sigma_\ell\right)^T$, *with the notations* $Z^\sigma_0 = \begin{pmatrix} X \\ \beta \mathbf{1}^T_N \end{pmatrix}$ *and* $Z^\sigma_\ell = \begin{pmatrix} \sigma(Z_\ell) \\ \beta \mathbf{1}^T_N \end{pmatrix}$.

Furthermore, if $\mathbf{W}$ *is a local minimizer of* $\mathcal{L}_\lambda$ *then* $(Z_1(X;\mathbf{W}),\dots,Z_L(X;\mathbf{W}))$ *is a local minimizer of* $\mathcal{L}^r_\lambda$*. Conversely, keeping the same notations, if* $(Z_\ell)_{\ell=1,\dots,L}$ *is a local minimizer of* $\mathcal{L}^r_\lambda$*, then* $\mathrm{W} = (Z_\ell (Z^\sigma_{\ell-1})^+)_{\ell=1,\dots,L}$ *is a local minimizer of* $\mathcal{L}_\lambda$*.*

Proof. We write Φ for the map which sends some weights $\mathbf{W}$ to the hidden representations $(Z_1(X;\mathbf{W}),\dots,Z_L(X;\mathbf{W}))$ and Ψ for the map which sends some hidden representations $\mathbf{Z} \in \mathcal{Z}$ to $\mathbf{W}$ with weight matrices $W_\ell = Z_\ell \left(Z^\sigma_{\ell-1}\right)^+$.

We clearly have $\Phi(\Psi(\mathbf{Z})) = \mathbf{Z}$ for any $\mathbf{Z} \in \mathcal{Z}$, however it is not true in general that $\Psi(\Phi(\mathbf{W}))$ for all $\mathbf{W}$ (actually this is true iff $\mathbf{W}$ lies in the image of Ψ).

Let $\mathcal{L}_\lambda(\mathbf{W}) = C(Y_{\mathbf{W}}) + \lambda \|\mathbf{W}\|^2$ and $\mathcal{L}^r_\lambda(\mathbf{Z}) = C(Z_L) + \lambda \sum_{\ell=1}^{L} \left\| Z_\ell \left(Z^\sigma_{\ell-1}\right)^+ \right\|^2_F$. One can show that $\mathcal{L}_\lambda(\Psi(\mathbf{Z})) = \mathcal{L}^r_\lambda(\mathbf{Z})$ for all $\mathbf{Z} \in \mathcal{Z}$ and $\mathcal{L}^r_\lambda(\Phi(\mathbf{W})) \leq \mathcal{L}_\lambda(\mathbf{W})$ for all $\mathbf{W}$ (actually $\mathcal{L}^r_\lambda(\Phi(\mathbf{W})) = \mathcal{L}_\lambda(\mathbf{W})$ if $\mathbf{W} \in \mathrm{Im}\Psi$ and $\mathcal{L}^r_\lambda(\Phi(\mathbf{W})) < \mathcal{L}_\lambda(\mathbf{W})$ otherwise). The first fact implies that $\inf_{\mathbf{W}} \mathcal{L}_\lambda(\mathbf{W}) \leq \inf_{\mathbf{Z}\in\mathcal{Z}} \mathcal{L}^r_\lambda(\mathbf{Z})$ while the second implies $\inf_{\mathbf{W}} \mathcal{L}_\lambda(\mathbf{W}) \geq \inf_{\mathbf{Z}\in\mathcal{Z}} \mathcal{L}^r_\lambda(\mathbf{Z})$, furthermore the maps Φ and Ψ must map global minimizers to global minimizers.

Local Minima: We now extend the correspondence to local minima and saddles:

We prove that if $\mathbf{Z}$ is a local minimum of $\mathbf{Z} \mapsto \mathcal{L}^r_\lambda(\mathbf{Z})$ then $\mathbf{W} = \Psi(\mathbf{Z})$ is a local minimum of $\mathbf{W} \mapsto \mathcal{L}_\lambda(\mathbf{W})$ through the contrapositive: if $\mathbf{W} = \Psi(Z)$ is not a local minimum of the loss $\mathbf{W} \mapsto \mathcal{L}_\lambda(\mathbf{W})$ (i.e. there is a sequence of weights $\mathbf{W}_1, \mathbf{W}_2, \dots$ which converges to $\mathbf{W}$ with $\mathcal{L}_\lambda(\mathbf{W}_i) < \mathcal{L}_\lambda(\mathbf{W})$ for all i) then $\mathbf{Z}$ is not a local minimum. We simply consider the sequence $\mathbf{Z}_i = \Phi(\mathbf{W}_i)$ which converges to $\mathbf{Z} = \Phi(\Psi(\mathbf{Z}))$ by the continuity of Φ. This sequence satisfies $\mathcal{L}^r_\lambda(\mathbf{Z}_i) \leq \mathcal{L}_\lambda(\mathbf{W}_i) < \mathcal{L}_\lambda(\mathbf{W}) = \mathcal{L}^r_\lambda(\mathbf{Z})$, proving that $\mathbf{Z}$ is not a local minimum.

Let us now prove if $\mathbf{W}$ is a local minimum of $\mathbf{W} \mapsto \mathcal{L}_\lambda(\mathbf{W})$ then $\mathbf{Z} = \Phi(\mathbf{W})$ is a local minimum of $\mathbf{Z} \mapsto \mathcal{L}^r_\lambda(\mathbf{Z})$, again using the contrapositive. Assume that there is a sequence $\mathbf{Z}_1, \mathbf{Z}_2, \dots$ which converges to $\mathbf{Z} = \Phi(\mathbf{W})$ with $\mathcal{L}^r_\lambda(\mathbf{Z}_i) < \mathcal{L}^r_\lambda(\mathbf{Z})$ for all i. We consider the sequence $\mathbf{W}_i = \Psi(\mathbf{Z}_i)$, however this sequence might not be convergent since Ψ is not continuous, however we know the sequence is bounded, since $\|\mathbf{W}_i\|^2 = \sum_{\ell=1}^{L} \left\| Z_{i,\ell} \left(Z^\sigma_{i,\ell-1}\right)^+ \right\|^2_F \leq \frac{1}{\lambda}\mathcal{L}^r_\lambda(\mathbf{Z}_i) < \frac{1}{\lambda}\mathcal{L}^r_\lambda(\mathbf{Z})$, this implies that there exists a subsequence $\mathbf{Z}_{k_i}$ such that $\mathbf{W}_{k_i} = \Psi\left(\mathbf{Z}_{k_i}\right)$ converges to some weights $\mathbf{W}'$. Note

that since $\Phi(\mathbf{W}) = \Phi(\mathbf{W}')$ the weight matrices must agree up to 'useless weights', i.e. for all ℓ

$$W_\ell = Z_\ell \left(Z^\sigma_{\ell-1}\right)^+ + \tilde{W}_\ell$$
$$W'_\ell = Z_\ell \left(Z^\sigma_{\ell-1}\right)^+ + \tilde{W}'_\ell.$$

If $\tilde{W}_\ell \neq 0$ then $\mathbf{W}$ is not a local minimum (since we could choose the weights $W^\epsilon_\ell = Z_\ell \left(Z^\sigma_{\ell-1}\right)^+ + (1-\epsilon)\tilde{W}_\ell$ for any $0 < \epsilon < 1$ to get a lower loss). We may therefore assume $\tilde{W}_\ell = 0$, but this implies that $\tilde{W}'_\ell = 0$ too since $\|\mathbf{W}\| = \|\mathbf{W}'\|$ and therefore $\mathbf{W}' = \mathbf{W}$ and therefore $\mathbf{W}$ is not a local minimum since the sequence $\mathbf{W}_{k_i}$ approaches $\mathbf{W}$ with a strictly lower loss. □

C.2.1 Optimization

It is possible to optimize the first reformulation directly, using projected gradient descent to guarantee that the constraints $\mathrm{Im} Z^T_{\ell+1} \subseteq \mathrm{Im}\left(Z^\sigma_\ell\right)^T$ remain satisfied. As we show now, this projection is unnecessary in the continuous case, which suggests that it might also be unnecessary in gradient descent with a small enough learning rate.

Assume there is a ℓ s.t. $\mathrm{Im} Z^T_{\ell+1} \nsubseteq \mathrm{Im}\left(Z^\sigma_\ell\right)^T$, i.e. there is a vector $v \in \mathbb{R}^N$ (with $\|v\| = 1$) such that $v \in \ker Z^\sigma_\ell$ but $\|Z_{\ell+1} v\| > 0$. Consider any $\tilde{\mathcal{Z}}$ such that $\left\|\tilde{\mathcal{Z}} - \mathcal{Z}\right\| \leq \epsilon$, then

$$\left\|\tilde{Z}_{\ell+1}\left(\tilde{Z}^\sigma_\ell\right)^+\right\|^2_F \geq \left\|\tilde{Z}_{\ell+1} v v^T \left(\tilde{Z}^\sigma_\ell\right)^+\right\|^2_F = \left\|\tilde{Z}_{\ell+1} v\right\|^2 \left\|v^T \left(\tilde{Z}^\sigma_\ell\right)^+\right\|^2 \geq \frac{\|Z_{\ell+1} v\|^2 - \epsilon}{\epsilon}.$$

This implies that the loss explodes in the vicinity of any point where the constraints are not satisfied. As a result, gradient flow on the cost $\mathcal{L}^r_\lambda$ starting from a value with a non-zero loss will never approach a non-acceptable point (where $\mathrm{Im} Z^T_{\ell+1} \nsubseteq \mathrm{Im}\left(Z^\sigma_\ell\right)^T$) since the loss is decreasing during gradient flow.

C.3 Equivalence for the second reformulation

Proposition C.3.1 (Proposition 5.3.3 of the main)**.** *For positively homogeneous non-linearities σ, the infimum of $\mathcal{L}_\lambda(\mathbf{W}) = C(Z_L(X;\mathbf{W})) + \lambda \|\mathbf{W}\|^2$, over the parameters $\mathrm{W} \in \mathbb{R}^P$ is equal to the infimum over $\mathcal{K}_\mathbf{n}(X)$ of*

$$\mathcal{L}^{\mathrm{k}}_\lambda(\mathbf{K}, Z_L) = C(Z_L) + \lambda \sum_{\ell=1}^{L} \mathrm{Tr}\left[K_\ell \left(K^\sigma_{\ell-1}\right)^+\right].$$

The set $\mathcal{K}_\mathbf{n}(X)$ is the set of covariances $\mathbf{K} = ((K_1, K^\sigma_1), \ldots, (K_{L-1}, K^\sigma_{L-1}))$ and outputs Z_L such that for all hidden layer $\ell = 1, \ldots, L-1$:

- *the pair (K_ℓ, K^σ_ℓ) belongs to the (translated) n_ℓ-conical hull*

$$S_{n_\ell,\beta} = \mathrm{cone}_{n_\ell}\left(\left\{\left(xx^T, \sigma(x)\sigma(x)^T\right) : x \in \mathbb{R}^N\right\}\right) + (0, \beta^2 \mathbf{1}_{N\times N}),$$

- $\mathrm{Im} K_\ell \subset \mathrm{Im} K^\sigma_{\ell-1}$*, with the notation $K^\sigma_0 = X^T X + \beta^2 \mathbf{1}_{N\times N}$ and for the outputs,* $\mathrm{Im} Z_L \subset \mathrm{Im} K^\sigma_{L-1}$.

Proof. Consider the map Ψ that maps parameters $\mathbf{W}$ to the the tuple $(\mathbf{K}, Z_L)$. We simply need to show that the image of Ψ is the set $\mathcal{K}_{\mathbf{n}}(X)$. The fact that $\mathrm{Im}\Psi \subset \mathcal{K}_{\mathbf{n}}(X)$ can easily be checked.

To prove $\mathrm{Im}\Psi \supset \mathcal{K}_{\mathbf{n}}(X)$ we need to construct a pre-image $\mathbf{W} \in \Psi^{-1}(\mathbf{K}, Z_L)$ from any tuple $(\mathbf{K}, Z_L)$ in $\mathcal{K}_{\mathbf{n}}(X)$. For every hidden layer ℓ, we have $(K_\ell, K_\ell^\sigma) \in S_{n_\ell,\beta}$. There are hence representations $Z_\ell \in \mathbb{R}^{n_\ell \times N}$ such that $K_\ell = Z_\ell^T Z_\ell$ and $K_\ell^\sigma = \sigma(Z_\ell)^T \sigma(Z_\ell) + \beta^2 \mathbf{1}_{N\times N}$, furthermore for all ℓ, we have $\mathrm{Im}Z_\ell^T = \mathrm{Im}K_\ell$ and $\mathrm{Im}\begin{pmatrix} \sigma(Z_\ell) \\ \beta \mathbf{1}_N^T \end{pmatrix} = \mathrm{Im}K_\ell^\sigma$, which implies that $\mathrm{Im}Z_\ell^T \subset \mathrm{Im}\left(Z_{\ell-1}^\sigma\right)^T$ and therefore that the tuple $(Z_1, \dots Z_L)$ is in the set $\mathcal{Z}_{\mathbf{n}}$ and we can choose the weight matrices $W_\ell = Z_\ell \left(Z_{\ell-1}^\sigma\right)^+$ to obtain a preimage $\mathbf{W} \in \Psi^{-1}(\mathbf{K}, Z_L)$. □

C.3.1 Non-correspondence of the local minima

Let us consider the map $\Gamma : \mathbf{Z} \mapsto (\mathbf{K}, Z_L)$ which maps each hidden representation Z_ℓ to the kernel pair $(Z_\ell^T Z_\ell, (Z_\ell^\sigma)^T Z_\ell^\sigma)$. The continuity of Γ implies that if $\Gamma(\mathbf{Z})$ is a local minimum then so is $\mathbf{Z}$. The converse is not true, instead we have:

Proposition C.3.2. *A kernel and outputs pair $(\mathbf{K}, Z_L)$ is a local minimum if all $\mathbf{Z} \in \Gamma^{-1}(\mathbf{K})$ are local minima.*

Proof. We will prove the contrapositive of this statement: if $(\mathbf{K}, Z_L)$ is a saddle (i.e. there is a sequence $(\mathbf{K}_1, Z_{L,1}), (\mathbf{K}_2, Z_{L,2}), \dots$ which converges to $(\mathbf{K}, Z_L)$ such that $\mathcal{L}_\lambda(\mathbf{K}_i, Z_{L,i}) < \mathcal{L}_\lambda(\mathbf{K}, Z_L)$), then there is a $\mathbf{Z} \in \Gamma^{-1}(\mathbf{K}, Z_L)$ which is a saddle.

First note that for any i, $\Gamma^{-1}(\mathbf{K}_i, Z_{L,i})$ is compact (it is closed and bounded since $\|Z_\ell\|_F^2 = \mathrm{Tr}[K_\ell] < \infty$). There is hence a sequence $\mathbf{Z}_1, \mathbf{Z}_2 \dots$ with $\mathbf{Z}_i \in \Gamma^{-1}(\mathbf{K}_i, Z_{L,i})$ which converges to some $\mathbf{Z}$. By the continuity of Γ, we have $\Gamma(\mathbf{Z}) = (\mathbf{K}, Z_L)$ and we have $\mathcal{L}_\lambda(\mathbf{Z}_i) = \mathcal{L}_\lambda(\mathbf{K}_i, Z_{L,i}) < \mathcal{L}_\lambda(\mathbf{K}, Z_L) = \mathcal{L}_\lambda(\mathbf{Z})$, hence proving that $\mathbf{Z}$ is a saddle as needed. □

Let us now give an example of a set of weights $\mathbf{W}$ of a depth $L = 2$ network which is a local minimum of $\mathcal{L}_\lambda$ but such that the corresponding covariances $(\mathbf{K}, Z_L)$ are not a local minimum of $\mathcal{L}_\lambda^k$:

Proposition C.3.3. *Consider a shallow ReLU network ($L = 2$) of widths $n_0 = 1, n_1 = 2, n_2 = 1$ with no bias $\beta = 0$. Consider the MSE error $\mathcal{L}_\lambda(\mathbf{W}) = \frac{1}{N}\|Y(X;\mathbf{W}) - Y\|_F^2$ for the size $N = 2$ dataset with inputs $X = \begin{pmatrix} 1 & -1 \end{pmatrix}$ and outputs $Y = \begin{pmatrix} 1 & 1 \end{pmatrix}$.*

For any $\lambda < 1$ and any choices of $a_1, a_2 > 0$ s.t. $a_1^2 + a_2^2 = 1 - \lambda$ the parameters

$$W_1 = \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}, W_2 = \begin{pmatrix} a_1 & a_2 \end{pmatrix}$$

are a local minimum of the loss $\mathcal{L}_\lambda(\mathbf{W})$ however, the corresponding covariances and outputs $(K_1, K_1^\sigma), Z_2$ are not a local minimum of the second reformulation $\mathcal{L}_\lambda^c((K_1, K_1^\sigma), Z_2)$.

Proof. Consider a depth $L = 2$ network with no bias ($\beta = 0$) and widths $\mathbf{n} = (1, 2, 1)$ with a training set of size $N = 2$, with inputs $X = (1, -1)$ and outputs $Y = (1, 1)$. Let us consider this loss in the region where all four weights are positive:

$$W_1 = \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}, W_2 = \begin{pmatrix} b_1 & b_2 \end{pmatrix}$$

with $a_1, a_2, b_1, b_2 \geq 0$. We then have the following activations

$$\begin{aligned} Z_1 &= \begin{pmatrix} a_1 & -a_1 \\ a_2 & -a_2 \end{pmatrix} \\ \sigma(Z_1) &= \begin{pmatrix} a_1 & 0 \\ a_2 & 0 \end{pmatrix} \\ Z_2 &= \begin{pmatrix} a_1 b_1 + a_2 b_2 & 0 \end{pmatrix}. \end{aligned}$$

The cost therefore takes the form

$$\mathcal{L}_\lambda(\mathbf{W}) = (1 - a_1 b_1 - a_2 b_2)^2 + 1 + \lambda \left(a_1^2 + a_2^2 + b_1^2 + b_2^2\right).$$

Let us now reformulate the loss in terms of the two positive values

$$\begin{aligned} c &= \left(\frac{a_1 + b_1}{2}\right)^2 + \left(\frac{a_2 + b_2}{2}\right)^2 \\ d &= \left(\frac{a_1 - b_1}{2}\right)^2 + \left(\frac{a_2 - b_2}{2}\right)^2. \end{aligned}$$

Since $2(c + d) = a_1^2 + a_2^2 + b_1^2 + b_2^2$ and $c - d = a_1 b_1 + a_2 b_2$, we can rewrite

$$\mathcal{L}_\lambda(\mathbf{W}) = (1 - c + d)^2 + 1 + 2\lambda(c + d).$$

The above is minimized (over the set of positive c, d) at $c = 1 - \lambda$ and $d = 0$, since it is the unique point of the quarterplane $\left\{ \begin{pmatrix} c \\ d \end{pmatrix} : c, d \geq 0 \right\}$ where the gradient

$$\nabla \mathcal{L}_\lambda(\mathbf{W}) = \begin{pmatrix} \partial_c \mathcal{L}_\lambda(\mathbf{W}) \\ \partial_d \mathcal{L}_\lambda(\mathbf{W}) \end{pmatrix} = \begin{pmatrix} 0 \\ 4\lambda \end{pmatrix}$$

points toward the inside of the quarterplane.

The set weights which optimal amongst the set of positive weights equals the set of positive weights such that $c = 1 - \lambda$ and $d = 0$. Such weights a_1, a_2, b_1, b_2 must satisfy $a_1 = b_1$ and $a_2 = b_2$ (since $d = 0$) and $a_1^2 + a_2^2 = 1 - \lambda$ (since $c = 1 - \lambda$). In other terms, the weights of the form

$$W_1 = \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}, W_2 = \begin{pmatrix} a_1 & a_2 \end{pmatrix}$$

for any choice of positive a_1, a_2 s.t. $a_1^2 + a_2^2 = 1 - \lambda$ (we have assumed that $\lambda < 1$). For any choice of a_1, a_2 that are both strictly positive, the above weights lie in the inside of the set of positive weights, which implies that these weights form a local minimum.

To prove that the corresponding covariances (K_1, K_1^σ) are not a local minimum of the reformulation, it is sufficient to find a pre-image of these covariances which is not a local minimum. We will show that the extrema of the segment of local minima that we identified are not local minima. Since all weights on the segment have the same covariances, it follows from Proposition C.3.1 that if one of those points is not a local minimum, the covariances cannot be a local minimum of the reformulation.

Let us consider one of the extrema:

$$W_1 = \begin{pmatrix} \sqrt{1-\lambda} \\ 0 \end{pmatrix}, W_2 = \begin{pmatrix} \sqrt{1-\lambda} & 0 \end{pmatrix}.$$

This extremum can be approached by the following weights as $\epsilon \searrow 0$

$$W_1^\epsilon = \begin{pmatrix} \sqrt{1-\lambda} \\ -\epsilon \end{pmatrix}, W_2^\epsilon = \begin{pmatrix} \sqrt{1-\lambda} & -\epsilon \end{pmatrix}.$$

We simply need to show that for small enough ϵ, we have $\mathcal{L}_\lambda(\mathbf{W}^\epsilon) < \mathcal{L}_\lambda(\mathbf{W})$. Let us first compute the activations

$$\begin{aligned} Z_1 &= \begin{pmatrix} \sqrt{1-\lambda} & -\sqrt{1-\lambda} \\ -\epsilon & \epsilon \end{pmatrix} \\ \sigma(Z_1) &= \begin{pmatrix} \sqrt{1-\lambda} & 0 \\ 0 & \epsilon \end{pmatrix} \\ Z_2 &= \begin{pmatrix} 1-\lambda & \epsilon^2 \end{pmatrix}. \end{aligned}$$

Therefore the cost $\mathcal{L}_\lambda(\mathbf{W}^\epsilon)$ takes the form

$$\mathcal{L}_\lambda(\mathbf{W}^\epsilon) = (1-\lambda-1)^2 + (\epsilon^2-1)^2 + 2\lambda\left((1-\lambda)+\epsilon^2\right).$$

Clearly for small enough $\epsilon > 0$, we have $\mathcal{L}_\lambda(\mathbf{W}^\epsilon) < \mathcal{L}_\lambda(\mathbf{W}) = \mathcal{L}_\lambda(\mathbf{W}^{\epsilon=0})$. □

C.4 Description of the Plateau

Proposition C.4.1 (Proposition 5.4.3 of the main)**.** *Let $(\mathbf{K}, Z_L) \in \mathcal{K}(X)$, then there are parameters $\mathbf{W}$ of a width $\mathbf{n}$ network with covariances and outputs $\mathbf{K}$ if and only if $n_\ell \geq \mathrm{Rank}_\sigma(K_\ell, K_\ell^\sigma)$ for all $\ell = 1, \ldots, L-1$.*

Proof. To prove that the constraints $n_\ell \geq \mathrm{Rank}_\sigma(K_\ell, K_\ell^\sigma)$ are sufficient, we construct the parameters $\mathbf{W}$ recursively from the first layer to the last. Since $n_1 \geq \mathrm{Rank}_\sigma(K_1, K_1^\sigma)$, there is a hidden representation $Z_1 \in \mathbb{R}^{n_\ell \times N}$ such that $K_\ell = Z_\ell^T Z_\ell$ and $K_\ell^\sigma = (Z_\ell^\sigma)^T Z_\ell^\sigma$ (there is a representation of dimension $\mathrm{Rank}_\sigma(K_1, K_1^\sigma) \times N$, but one can add some zero lines to it to obtain Z_1 without changing the resulting K_ℓ and K_ℓ^σ). Since $\mathrm{Im} Z_1 = \mathrm{Im} K_1 \subset \mathrm{Im} K_0^\sigma = \mathrm{Im} Z_0^\sigma$, we can choose the parameters of the first layer as $W_1 = Z_1 (Z_0^\sigma)^+$. All other weight matrices W_ℓ are then constructed in the same manner.

The fact that the constraints $n_\ell \geq \mathrm{Rank}_\sigma(K_\ell, K_\ell^\sigma)$ are necessary follows from the fact that for any network of width $\mathbf{n}$ with parameters $\mathbf{W}$ we have that $\mathrm{Rank}_\sigma(K_\ell(\mathbf{W}), K_\ell^\sigma(\mathbf{W})) \leq n_\ell$ since $K_\ell(\mathbf{W}) = (Z_\ell(\mathbf{W}))^T Z_\ell(\mathbf{W})$ and $K_\ell^\sigma(\mathbf{W}) = (Z_\ell^\sigma(\mathbf{W}))^T Z_\ell^\sigma(\mathbf{W})$. □

C.4.1 Tightness of the upper bound

Let us first prove the Proposition on the CP-rank of matrices resulting from graphs without cliques:

Proposition C.4.2 (Proposition 5.4.8 of the main)**.** *Given a graph G with N vertices and k edges, consider the $k \times N$ matrix E with entries $E_{ev} = 1$ if the vertex v is an endpoint of the edge e and $E_{ev} = 0$ otherwise. The matrix $A = E^T E$ is completely positive and if the graph G contains no cliques of 3 or more vertices then $\mathrm{Rank}_{\mathrm{cp}}(A) = k$.*

Proof. The fact that $A = E^T E$ implies $\text{Rank}_{\text{cp}}(A) \leq k$, we only need to show $\text{Rank}_{\text{cp}}(A) \geq k$. Let assume that there is another decomposition $E^T E = B^T B$ for some $m' \times N$ matrix B with positive entries, we will now show that $k' \geq k$.

First, we show that the absence of cliques of 3 or more vertices implies that each line B_e has at most 2 non-zero entries. The absence of cliques implies that for all sets $\Omega = \{v_1, \ldots, v_r\}$ of 3 or more vertices, there must be a pair of vertices $v, w \in \Omega$ which are not connected, i.e. $(E^T E)_{vw} = 0$. If one line B_e contains more than two non-zero entries, corresponding to the vertices $\Omega = \{v_1, \ldots, v_r\}$ then for all $v \neq w \in \Omega$, we have

$$\left(B^T B\right)_{vw} \geq \left(B_e B_e^T\right)_{vw} = 1.$$

Now if all lines B_e have at most two non-zero entries it implies that $B_e B_e^T$ has at most two non-zero off-diagonal entries. We know that $E^T E$ has $2k$ non-zero off-diagonal entries. Since

$$E^T E = \sum_{e=1}^{m'} B_e B_e^T$$

it follows that $k' \geq k$, otherwise we could not recover all the off-diagonal entries. □

We may now prove the tightness of the upper bound on the σ-rank of the hidden representation in shallow ReLU networks without bias:

Proposition C.4.3 (Proposition 5.4.9 of the main)**.** *Consider a width-*n *shallow network ($L = 2$) with ReLU activation, no bias $\beta = 0$, $n_0 = N$, $n_1 \geq N(N+1)$, input dataset $X_N = I_N$, and any output dataset Y_N such that $\left(Y_N^T Y_N\right)^{\frac{1}{2}}$ is a completely positive matrix with CP-rank k.*

At any global minimum of $R_{\text{n}}(X_N, Y_N)$, we have $\text{Rank}_\sigma(K_1, K_1^\sigma) = k$. Furthermore for λ small enough, at any global minimum of $\mathcal{L}_{\lambda,\text{n}}^{\text{MSE}}(\mathbf{W}) = \frac{1}{N}\|Y(X_N; \mathbf{W}) - Y_N\|_F^2 + \lambda \|\mathbf{W}\|^2$, we have $\text{Rank}_\sigma(K_1, K_1^\sigma) \geq k$.

Proof. The proof is in two steps, we first show that the minimizer $\mathbf{K}$ of the representation cost has rank k, and then use this to show that for small enough λs the rank must be at least k.

Representation Cost: We first show that at a minimizer (K_1, K_1^σ) of the cost $\text{Tr}[K_1] + \text{Tr}\left[YY^T (K_1^\sigma)^+\right]$, we have $K_1 = K_1^\sigma$. This follows from the fact that if $K_1 \neq K_1^\sigma$, then the pair (K_1^σ, K_1^σ) has a strictly lower cost than the pair (K_1, K_1^σ): for any Z_1 such that $K_1 = Z_1^T Z_1$ and $K_1^\sigma = \sigma(Z_1)^T \sigma(Z_1)$, we have that $\text{Tr}[K_\ell] = \|Z_1\|_F^2 \geq \|\sigma(Z_1)\|_F^2 = \text{Tr}[K_\ell^\sigma]$ and the inequality is strict if $Z_1 \neq \sigma(Z_1)$ (which happens iff $K_1 \neq K_1^\sigma$).

The optimization of the previous cost over pairs (K_1, K_1^σ) in S is therefore equivalent to the optimization of the cost $K \mapsto \text{Tr}[K] + \text{Tr}\left[Y^T Y K^+\right]$ over completely positive matrices K such that $\text{Im}Y \subset \text{Im}K$. If we remove the complete positiveness constraint on K, then the unique minimizer of the above is $K = \left(Y^T Y\right)^{\frac{1}{2}}$. Now since $\left(Y^T Y\right)^{\frac{1}{2}}$ is completely positive, it is also the unique minimizer over complete positive matrices.

We therefore have $\text{Rank}_\sigma(K_1, K_1^\sigma) = \text{Rank}_{cp}\left(\left(Y^T Y\right)^{\frac{1}{2}}\right) = k$.

Regularized Loss: Let us consider the regularized loss

$$\frac{1}{N}\|Z_2 - Y\|_F^2 + \lambda \text{Tr}[K_1] + \lambda \text{Tr}\left[Z_2^T Z_2 (K_1^\sigma)^+\right].$$

The minimizer $\mathbf{K}(\lambda) = (K_1(\lambda), K_1^\sigma(\lambda), Z_2(\lambda))$ converges as $\lambda \searrow 0$ to the pair (K_1, K_1^σ, Y) where $K_1 = K_1^\sigma$ is the minimizer of the representation cost.

Let us now assume that there is no λ_0 such that for all $\lambda < \lambda_0$, any minimizer $\mathbf{K}$ of the loss $\mathcal{L}_\lambda$ satisfies $\text{Rank}_\sigma(K_1, K_1^\sigma) \geq k$. This would imply that there is a sequence $\lambda_1, \lambda_2, \dots$ of ridges with $\lim_{n\to\infty} \lambda_n = 0$ and corresponding minimizers $\mathbf{K}_1, \mathbf{K}_2, \dots$ (where $\mathbf{K}_n$ is a minimizer of the loss $\mathcal{L}_{\lambda_n}$) such that $\text{Rank}_\sigma\left(K_{n,1}, K_{n,1}^\sigma\right) < k$. Now by Proposition C.4.1 for all n there are parameters $\mathbf{W}_n$ of shallow ReLU network with $n_1 = k-1$ neurons in the hidden layer with covariances equal $\mathbf{K}_n$. The sequence $\mathbf{W}_1, \mathbf{W}_2, \dots$ is uniformly bounded in norm by the representation cost $R(X_N, Y_N)$, there is therefore a converging subsequence $\mathbf{W}_{n_1}, \mathbf{W}_{n_2}, \dots$ which converges to some parameters $\mathbf{W}$. The covariances and outputs (K_1, K_1^σ, Y) at these limiting parameters $\mathbf{W}$ must minimize the representation cost, i.e. $K_1 = K_1^\sigma = \left(Y^T Y\right)^{\frac{1}{2}}$, but this yields a contradiction, since $\text{Rank}_\sigma(K_1, K_1^\sigma) = k$ but $\mathbf{W}$ are parameters of network with $n_1 = k-1$ neurons in the hidden layer, which would imply $\text{Rank}_\sigma(K_1, K_1^\sigma) \leq k-1$. □

To show the tightness (up to constant factor) of the upper bound, one can simply apply this proposition to the special case $Y_N = E^T E$, where E is the edge-vertex incidence matrix of the complete bipartite graph, in which case $k = \frac{N^2}{4}$.

We could also consider an output dataset $Y_N \in \mathbb{R}^{n_L \times N}$ whose lines are one-hot vectors, corresponding to a classification task. If we reorder the training set by class, the covariance $Y_N^T Y_N$ is a block diagonal matrix, with all ones blocks corresponding to each class. The square root $\left(Y_N^T Y_N\right)^{\frac{1}{2}}$ is also block-diagonal but the block of a class i has value $\frac{1}{m_i}$ where m_i is the number of datapoints in the class i. The matrix $\left(Y_N^T Y_N\right)^{\frac{1}{2}}$ is completely positive and has rank k equal to the number of classes. This implies a much earlier plateau, which could explain why in real-world classification tasks, we observe a very early plateau.

Remark C.4.4. The representation cost for $Y = E^T E$ is $2\|E\|_F^2 = 4\frac{N^2}{4} = N^2$. We can obtain an almost optimal representation with $n_1 = N$ neurons by taking the weights $W_1 = \sqrt{\frac{N}{2}} I$ and $W_2 = \sqrt{\frac{2}{N}} E^T E$, with norm $\left\|\sqrt{\frac{N}{2}} I\right\|_F^2 + \left\|\sqrt{\frac{2}{N}} E^T E\right\|_F^2 = \frac{N^2}{2} + \frac{2}{N}(N\frac{N^2}{4} + 2\frac{N^2}{4}) = \frac{N^2}{2} + \frac{N^2}{2} + N = N^2 + N$.

C.4.2 One Dimensional Shallow Network

We now prove an upper bound on the start of the plateau for shallow networks with one-dimensional inputs and outputs:

Proposition C.4.5 (Proposition 5.4.10 of the main)**.** *Consider shallow networks ($L = 2$) with scalar inputs and outputs ($n_0 = n_2 = 1$), a ReLU nonlinearity, and a dataset $X, Y \in \mathbb{R}^{1\times N}$. Both the representation cost $R_{\mathbf{n}}(X, Y)$ and global minimum $\min_{\mathbf{W}} \mathcal{L}_{\lambda,\mathbf{n}}(\mathbf{W})$ for any $\lambda > 0$ are constant as long as $n_1 \geq 4N$.*

Proof. We show that if there is a network with depth $L = 2$ and $n_1 > 4N$ hidden neurons, we can construct a network with strictly less neurons with the same outputs on the dataset and a smaller parameter norm.

The network function can be written in the form

$$f_{\mathbf{W}}(x) = b + \sum_{k=1}^{n_1} a_k \sigma\left(c_k x + d_k\right).$$

Appendix C. Appendix: Feature learning in L_2-regularized DNNs: Attraction/repulsion and sparsity

We may assume that for all neuron i, we have $a_k^2 = c_k^2 + d_k^2$ since if this is not the case, one can multiply a_k by a scalar and divide c_k and d_k by the same scalar to satisfy this constraint while reducing the norm of the parameters.

For each neuron i, we define the cusp of the neuron the value $-\frac{d_k}{c_k}$, which is the point where the neuron goes from dead to active.

If there are neurons that are inactive on the whole training set, they can simply be removed without changing the outputs and reducing the norm.

If there are more $4N$ neurons, we either have:

1. There are more than 4 neurons whose cusp lies between two inputs x_i and x_{i+1} (w.l.o.g. we assume $x_1 < \cdots < x_N$).

2. There are more than 2 neurons whose cusp lies to the left or right of the data.

We will now show how in the case 1, one can remove a neuron while keeping the same outputs on the training data and reducing the norm of the parameters. The second case is analogous.

If there are five or more neurons with a cusp between x_i and x_{i+1}, then two of those neurons k, m must have the same signs $\text{sign} a_k = \text{sign} a_m$ and $\text{sign} c_k = \text{sign} c_m$ (w.l.o.g. we assume they are all positive). We will replace these two neurons by a single neuron $\tilde{a}\sigma(\tilde{c}x + \tilde{d})$ where $\tilde{a}, \tilde{c}, \tilde{d}$ are chosen as the unique positive values ($\tilde{d}$ may be negative) to satisfy

$$\begin{aligned}
\tilde{a}\tilde{c} &= a_k c_k + a_m c_m \\
\tilde{a}\tilde{d} &= a_k d_k + a_m d_m \\
\tilde{a}^2 &= \tilde{c}^2 + \tilde{d}^2.
\end{aligned}$$

First note this new neurons contributes $\tilde{a}^2 + \tilde{c}^2 + \tilde{d}^2 = 2\tilde{a}^2$ to the norm of the parameters which is less than the two previous neurons $2a_k^2 + 2a_m^2$, since

$$\begin{aligned}
\tilde{a}^2 &= \sqrt{\tilde{a}^2\left(\tilde{c}^2 + \tilde{d}^2\right)} \\
&= \sqrt{\left(a_k d_k + a_m d_m\right)^2 + \left(a_k c_k + a_m c_m\right)^2} \\
&= (a_k + a_m)\sqrt{\left(\frac{a_k}{a_k + a_m} d_k + \frac{a_m}{a_k + a_m} d_m\right)^2 + \left(\frac{a_k}{a_k + a_m} c_k + \frac{a_m}{a_k + a_m} c_m\right)^2} \\
&\leq (a_k + a_m)\left(\frac{a_k}{a_k + a_m}\sqrt{d_k^2 + c_k^2} + \frac{a_m}{a_k + a_m}\sqrt{d_m^2 + c_m^2}\right) \\
&= a_k^2 + a_m^2
\end{aligned}$$

where the inequality follows from the convexity of the norm function $(c, d) \mapsto \sqrt{c^2 + d^2}$.

For any x with $x \leq x_i$ or $x \geq x_{i+1}$, one can check that

$$\tilde{a}\sigma(\tilde{c}x + \tilde{d}) = a_k\sigma(c_k x + d_k) + a_m\sigma(c_m x + d_m),$$

which implies that replacement has not changed the values of the network on the training set. □

D Appendix: A Generalization bound for nearly-linear networks

D.1 Discussion

D.1.1 Assumptions

Whitened data. Overall, the assumption on whitened data is not necessary for a result similar to Theorem 6.5.2 to hold. We assume the data to be whitened for two reasons. First, it legitimates the choice of training time $t^*_\alpha(\beta)$ since it is based on the analysis of Saxe et al. (2013), which assumed the data to be whitened. If we dropped it, we could still evaluate the bound of Theorem 6.5.2 at $t = t^*_\alpha(\beta)$, but it would be less clear whether the trainining risk $\hat{R}_\gamma$ becomes already small by this time.

Second, we had to bound $\|\tilde{X}\|$ throughout the proof of Theorem 6.5.2 and $\|\tilde{X}^+\|$ in the proof of Lemma 6.7.3. For whitened data, these are simply $\sqrt{m}$ and $1/\sqrt{m}$, which is a clear dependence on m, making the final bound look cleaner. Otherwise, they would be random variables whose dependence on m would be not obvious.

Third, if we considered training on the original dataset (Y, X) instead of the whitened one, $(Y, \tilde{X})$, we would have to know $YX^\top$ and $XX^\top$ in order to determine $\theta^0(t)$ for a given t and initialization $\theta^0(0)$. These two matrices have $d + \frac{d(d+1)}{2}$ parameters, compared to just d for $Y\tilde{X}^\top$. This way, Υ_κ would grow as d instead of $\sqrt{d}$.

Gradient flow. We expect our technique to follow through smoothly for finite-step gradient descent. Introducing momentum also seems doable. However, generalizing it to other training procedures, e.g. the ones which use normalized gradients, might pose problems since it is not clear how to reasonably upper-bound the norm of the elementwise ratio of two matrices.

Assumption 6.5.1. We use this assumption to prove the second part of Lemma 6.7.4. If we dropped it, the bound would be $\|\Xi^\tau \tilde{X}^\top\|_F \leq \|\Xi^\tau\|_F \|\tilde{X}^\top\| \leq 1 + \rho\beta^L$ instead of $s + \rho\beta^L$. This would result in larger exponents for the definition u in Theorem 6.5.2.

As an argument in favor of this assumption, we demonstrate it empirically first (Fig. 6.7), and second, we prove it for the linear case, see Appendix D.5.

D.1.2 Proof

We expect the bounds on weight norms $u(t)$ to be quite loose since we use Lemma 6.7.4 to bound the loss. This lemma bounds the loss with its value at the initialization, while the loss should necessarily decrease. If we could account for the loss decrease, the resulting $u(t)$ would increase with a lower exponent, or even stay bounded as $\bar{u}(t)$, which corresponds to a linear model, does. This way, we would not have to assume ϵ to vanish as β vanishes in order to keep the bound non-diverging for small β at the training time $t^*_\alpha(\beta)$. Also, the general bound of Theorem 6.5.2 would diverge with training time t much slower. We leave it for future work.

As we see from our estimates, Υ_κ becomes the main bottleneck of our bound for small ϵ. The bound we used for Υ_κ is very naive; we believe that better bounds are possible. Improving the bound for Υ_κ will increase the maximal ϵ for which our bound is non-vacuous.

D.1.3 Other architectures

As becomes apparent from the proof in Appendix D.3.1, the proxy-model for $\kappa = 2$, Eq. (6.25), deviates from the original model $f^\epsilon_{\theta^\epsilon}$ as $O(\epsilon^2)$ for any map $(\epsilon, \theta, x) \to f^\epsilon_\theta(x)$ as long as the following holds:

1. $f^0_\theta(x)$ is linear in x for any θ;
2. $\frac{\partial^2 f^\epsilon_\theta(x)}{\partial \epsilon \partial \theta}$ is continuous as a function of (ϵ, θ, x);
3. the result of learning $\theta^\epsilon(t)$ is differentiable in ϵ for any t.

This is directly applicable to convolutional networks with no other nonlinearities except for ReLU's; in partiular, without max-pooling layers. One may introduce max-poolings by interpolating between average-poolings (which are linear) for $\epsilon = 0$ and max-poolings for $\epsilon = 1$. This is not applicable to Transformers (Vaswani et al., 2017) since attention layers are inherently nonlinear: queries and keys have to be multiplied.

Compared to the fully-connected case of the present work, our bound might become even tighter for convolutional nets since d becomes the number of color channels (up to 3) instead of the whole image size in pixels. However, the corresponding proxy-models might be over-simplistic: the linear net they will deviate from is just a global average-pooling followed by a linear $\mathbb{R}^d \to \mathbb{R}$ map. We leave exploring the convolutional net case for future work.

D.2 How far could we get with our approach?

Broadly speaking, the approach we apply in this work could be described as follows. Suppose we have a model f learned on a dataset (X, Y), and another "proxy" model g which we construct having an access to the same dataset (X, Y). We bound the distribution risk of the "original" model using the following relation:

$$R(f) - \hat{R}_\gamma(f) \le R^C_\gamma(f) - \hat{R}^C_\gamma(f) \le R^C_\gamma(g) - \hat{R}^C_\gamma(g) + \frac{\mathbb{E}_{x\sim\mathcal{D}}|g(x) - f(x)| + \frac{1}{m}\|g(X) - f(X)\|_1}{\gamma}. \tag{D.1}$$

In words, we say that performance of f is worse than that of g at most by some deviation term.

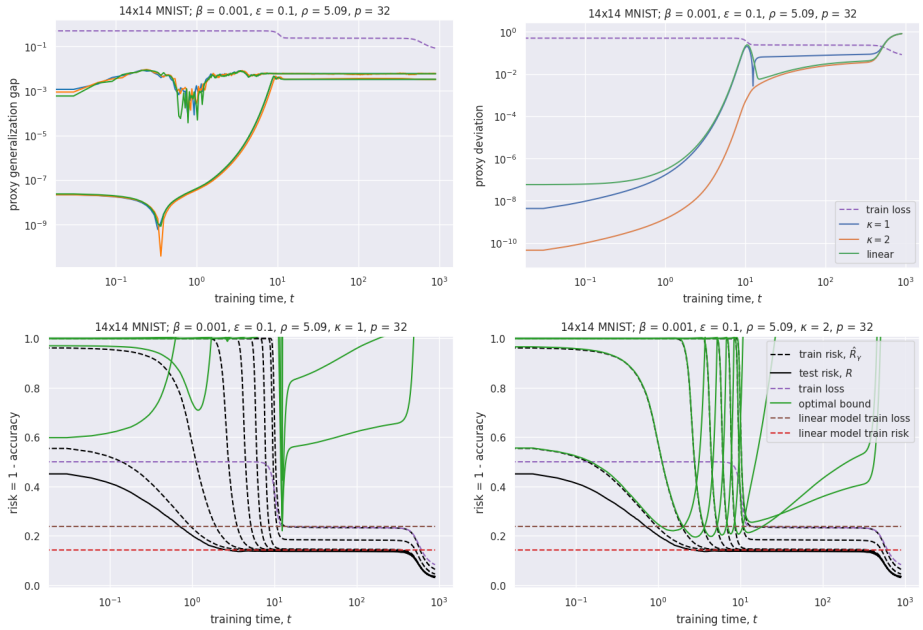


Figure D.1: We examine the optimistic bound of Eq. (D.1) for the proxy-models proposed in our paper: the $\kappa = 1$ one, $f^{\epsilon}_{\theta^0}$, the $\kappa = 2$ one of Eq. (6.25), and also a linear proxy, $f^{0}_{\theta^0}$. The bottom row demonstrates the full bound (green lines), while the top one depicts the two components of the bound, namely, the proxy model generalization gap $R^C_\gamma(g) - \hat{R}^C_\gamma(g)$ and the proxy model deviation $\mathbb{E}\,|f(x) - g(x)| + \hat{\mathbb{E}}\,|f(x) - g(x)|$, separately. Different lines of the same color (e.g. solid green and dashed black lines on the bottom row) correspond to different values of γ. Proxy generalization gap stays low during the whole training (top left figure), while the train risk and the model deviation over gamma contribute significantly (bottom row, two groups of lines correspond to the minimal and the maximal γ we considered). The optimistic bound for the 1st-order proxy (bottom left) gets non-vacuous only at the moment when GF escapes the origin and reaches the linear model loss. The bound for the 2nd-order proxy (bottom right) becomes non-vacuous soon after the original model becomes non-vacuous (but still stays near the origin), and stays non-vacuous until the model starts exploiting its nonlinearity to reduce the loss below the optimal linear model loss level (the last drop of purple and black lines).

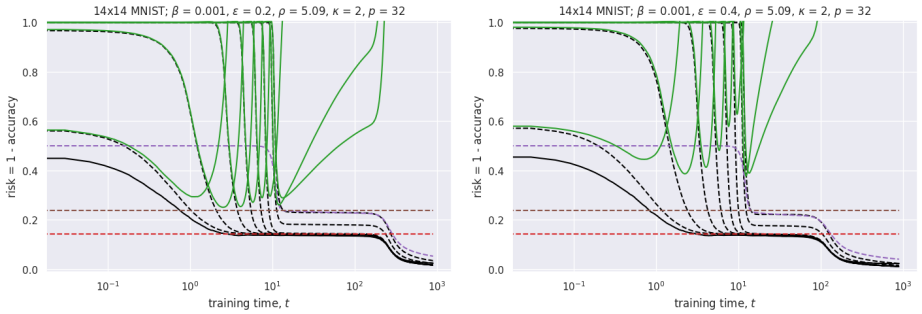


Figure D.2: The optimistic bound of Eq. (D.1) based on our $\kappa = 2$ proxy stays non-vacuous up to $\epsilon = 0.4$ until the gradient flow starts "exploiting" the nonlinearity (the last drop of purple and black lines).

The bound ends up to be good whenever (a) the generalization gap of g could be well-bounded, and (b) g does not deviate from f much. That is why we considered proxy-models based on linear learned weights: their generalization gap could be easily bounded analytically, and they do not deviate much from corresponding leaky ReLU nets as long as ReLU negative slopes are close to one.

The biggest conceptual disadvantage of this approach is that, given both f and g learn the training dataset, we have no chance proving that f performs better than g, we could only prove that f performs *not much worse* than g. Do the proxy-models we use in the present paper perform well, and how much do they deviate from original models? Our main theoretical result, Theorem 6.5.2, bounds the proxy-model generalization gap and the deviation from above. These bounds are arguably not optimal. It is therefore instructive to examine how well the bound would perform, if we could estimate Eq. (D.1) exactly.

D.2.1 Empirical validation

Setup. We work under the same setup as in Section 6.6[1], but instead of evaluating the bound of Theorem 6.5.2, we actually train a linear model with exactly the same procedure as for the original model, in order to get trained linear weights θ^0. We then evaluate the proxy-models considered in the present work: (1) the one for $\kappa = 1$, $f^{\epsilon}_{\theta^0}$, (2) the one for $\kappa = 2$, see Eq. (6.25), and also (3) the linear network, $f^0_{\theta^0}$. We then evaluate the rhs of Eq. (D.1) using a test part of the MNIST dataset. For this "optimistic" bound, we consider much larger values of epsilon: $\epsilon \geq 0.1$, i.e. our model much less "nearly-linear" now than before.

Figures. We present the results on Figs. D.1 and D.2. Dashed lines correspond to the train set, while solid ones correspond to the test set. Black lines are risks of the actual trained model $f^{\epsilon}_{\theta^{\epsilon}(t)}$: $R(t)$ and $\hat{R}_{\gamma}(t)$, respectively. Green lines are our "optimistic" bound Eq. (D.1) evaluated at different values of γ. Purple lines denote MSE loss of the actual trained model.

We also put three baselines on the plots. The dotted black line is the classification risk (and the

[1] We also downsample MNIST images to 14x14 instead of 7x7. The reason why we do it is that on one hand, we wanted to test our bounds on more realistic scenarios, while on the other, X does not appear to be full-rank for the original 28x28 MNIST.

MSE loss) of a zero model $f \equiv 0$. The brown dashed line is the MSE (train) loss of the optimal linear model, $f : x \to Y\tilde{X}^+x$. Finally, the red dashed line is the (train) classification risk of the optimal linear model.

Training phases. As we observe on risk plots (Fig. D.2 and the bottom row of Fig. D.1), the training process could be divided into three phases. During the first phase, the risk decreases until it reaches the risk of the optimal linear model, while the loss stays at the level of $f \equiv 0$. This indicates that while the weights stay very close to the origin, the network outputs already align with the outputs of the optimal linear model. During the second phase, both the loss and the risk stay at the level of the optimal linear model. As for the following phase, both the risk and the loss drop below the optimal linear model level. Therefore from this point on, the network starts to "exploit" its nonlinearity in order to reduce the train loss.

Observations:

- The generalization gap stays negligible for all models and γ's considered (Fig. D.1, top left);
- The proxy-model for $\kappa = 2$ approximates the original model best among all three proxies considered (Fig. D.1, top right);
- While the linear approximation deviates from the original model more than the one for $\kappa = 2$ during the first phase, their deviations are similar during the subsequent phases;
- At the same time, the $\kappa = 1$ approximation deviates more than that of $\kappa = 2$ during the second phase;
- The transition between the first and the second phases results in a nonmonotonic behavior of the deviation from the original model for $\kappa = 1$ and linear proxy-models;
- The resulting optimistic bound for $\kappa = 2$ (green lines of Fig. D.1, bottom right) stays non-vacuous during the first two phases for $\epsilon = 0.1$, while this is not the case for $\kappa = 1$ (green lines of Fig. D.1, bottom left);
- The optimistic bound for $\kappa = 2$ stays non-vacuous up to $\epsilon = 0.4$ (green lines of Fig. D.2).

It is tempting to assume that the weights θ^ϵ follow the same trajectory as the weights of the linear model, θ^0, during the first two phases. However, if it was the case, the $\kappa = 1$ proxy-model, $f^\epsilon_{\theta^0}$, would coincide with the original one, $f^\epsilon_{\theta^\epsilon}$, during this period. Then their quality would be the same; however, Fig. D.1, top right, demonstrates the opposite.

D.3 Missing calculations in Section 6.7

D.3.1 Proof of Lemma 6.7.3

We have:

$$\frac{1}{\epsilon^2}\left\|f^\epsilon_{\theta^\epsilon}(x) - \tilde{f}^\epsilon_{\theta^0,\theta^\epsilon}(x)\right\| = \frac{1}{\epsilon^2}\left\|\int_0^\epsilon \left(\frac{\partial f^\epsilon_{\theta^\tau}(x)}{\partial \tau} - \frac{\partial f^\epsilon_{\theta^\tau}(\tilde{X})\tilde{X}^+x}{\partial \tau}\right) d\tau\right\| \le \frac{1}{\epsilon}\sup_{\tau\in[0,\epsilon]}\left\|\frac{\partial f^\epsilon_{\theta^\tau}(x)}{\partial \tau} - \frac{\partial f^\epsilon_{\theta^\tau}(\tilde{X})}{\partial \tau}\tilde{X}^+x\right\|. \tag{D.2}$$

Appendix D. Appendix: A Generalization bound for nearly-linear networks

Since $f^0_{\theta^\tau}$ is a linear network and $\operatorname{rk}\tilde{X} = d$, we have $f^0_{\theta^\tau}(x) = f^0_{\theta^\tau}(\tilde{X})\tilde{X}^+x$ and $\frac{\partial f^0_{\theta^\tau}(x)}{\partial\tau} = \frac{\partial f^0_{\theta^\tau}(\tilde{X})\tilde{X}^+x}{\partial\tau}$. This implies

$$\begin{aligned}\frac{1}{\epsilon}\left\|\frac{\partial f^\epsilon_{\theta^\tau}(x)}{\partial\tau} - \frac{\partial f^\epsilon_{\theta^\tau}(\tilde{X})}{\partial\tau}\tilde{X}^+x\right\| &= \frac{1}{\epsilon}\left\|\int_0^\epsilon\left(\frac{\partial^2 f^\rho_{\theta^\tau}(x)}{\partial\rho\partial\tau} - \frac{\partial^2 f^\rho_{\theta^\tau}(\tilde{X})\tilde{X}^+x}{\partial\rho\partial\tau}\right)d\rho\right\| \\ &\le \sup_{\rho\in[0,\epsilon]}\left\|\frac{\partial^2 f^\rho_{\theta^\tau}(x)}{\partial\rho\partial\tau} - \frac{\partial^2 f^\rho_{\theta^\tau}(\tilde{X})}{\partial\rho\partial\tau}\tilde{X}^+x\right\| \\ &\le \sup_{\rho\in[0,\epsilon]}\left\|\frac{\partial^2 f^\rho_{\theta^\tau}(x)}{\partial\rho\partial\tau}\right\| + \sup_{\rho\in[0,\epsilon]}\left\|\frac{\partial^2 f^\rho_{\theta^\tau}(\tilde{X})}{\partial\rho\partial\tau}\right\|\|\tilde{X}^+x\|.\end{aligned} \tag{D.3}$$

Since we use LeakyReLU,

$$\left\|\frac{\partial^2 f^\rho_{\theta^\tau}(x)}{\partial\rho\partial\tau}\right\| \le (L-1)\sum_{l=1}^L\left\|\frac{\partial W_l^\tau}{\partial\tau}\right\|\prod_{k\neq l}\|W_k^\tau\|\|x\|; \tag{D.4}$$

$$\left\|\frac{\partial^2 f^\rho_{\theta^\tau}(\tilde{X})}{\partial\rho\partial\tau}\right\| \le (L-1)\left\|\frac{\partial W_1^\tau\tilde{X}}{\partial\tau}\right\|_F\prod_{k\in[2,L]}\|W_k^\tau\| + (L-1)\|W_1^\tau\tilde{X}\|_F\sum_{l=2}^L\left\|\frac{\partial W_l^\tau}{\partial\tau}\right\|\prod_{k\in[2,L]\setminus\{l\}}\|W_k^\tau\|. \tag{D.5}$$

Finally, since $\tilde{X}\tilde{X}^\top = mI_d$, we have $\|\tilde{X}^+\| = \frac{1}{\sqrt{m}}$. Combining everything together, we arrive into

$$\begin{aligned}\frac{\left\|f^\epsilon_{\theta^\epsilon}(x) - \tilde{f}^\epsilon_{\theta^0,\theta^\epsilon}(x)\right\|}{(L-1)\epsilon^2\|x\|} &\le \sup_{\tau\in[0,\epsilon]}\left[\sum_{l=1}^L\left\|\frac{\partial W_l^\tau}{\partial\tau}\right\|\prod_{k\neq l}\|W_k^\tau\|\right] \\ &+ \sup_{\tau\in[0,\epsilon]}\left[\frac{1}{\sqrt{m}}\|W_1^\tau\tilde{X}\|_F\sum_{l=2}^L\left\|\frac{\partial W_l^\tau}{\partial\tau}\right\|\prod_{k\in[2,L]\setminus\{l\}}\|W_k^\tau\|\right] \\ &+ \sup_{\tau\in[0,\epsilon]}\left[\frac{1}{\sqrt{m}}\left\|\frac{\partial W_1^\tau\tilde{X}}{\partial\tau}\right\|_F\prod_{k\in[2,L]}\|W_k^\tau\|\right].\end{aligned} \tag{D.6}$$

Plugging the bounds from Lemma 6.7.1 then gives

$$\left\|f^\epsilon_{\theta^\epsilon}(x) - \tilde{f}^\epsilon_{\theta^0,\theta^\epsilon}(x)\right\| \le (L-1)(L+1+(L-1)\rho)u^{L-1}v\|x\|\epsilon^2. \tag{D.7}$$

By a similar reasoning,

$$\begin{aligned}\frac{\left\|f^\epsilon_{\theta^\epsilon}(\tilde{X}) - \tilde{f}^\epsilon_{\theta^0,\theta^\epsilon}(\tilde{X})\right\|}{(L-1)\epsilon^2} &\le 2\sup_{\tau\in[0,\epsilon]}\left[\|W_1^\tau\tilde{X}\|_F\sum_{l=2}^L\left\|\frac{\partial W_l^\tau}{\partial\tau}\right\|\prod_{k\in[2,L]\setminus\{l\}}\|W_k^\tau\|\right] \\ &+ 2\sup_{\tau\in[0,\epsilon]}\left[\left\|\frac{\partial W_1^\tau\tilde{X}}{\partial\tau}\right\|_F\prod_{k\in[2,L]}\|W_k^\tau\|\right].\end{aligned} \tag{D.8}$$

Plugging the same bounds,

$$\left\|f^\epsilon_{\theta^\epsilon}(\tilde{X}) - \tilde{f}^\epsilon_{\theta^0,\theta^\epsilon}(\tilde{X})\right\| \le 2(L-1)(1+(L-1)\rho)u^{L-1}v\epsilon^2\sqrt{m}. \tag{D.9}$$

D.3.2 Proof of Lemma 6.7.4

Recall the definition of Ξ^τ:

$$\Xi^\tau(t) = \frac{1}{m}\left(Y - W_L^\tau(t)\left[D_{L-1}^\tau(t) \odot W_{L-1}^\tau(t)\left[\dots W_2^\tau(t)\left[D_1^\tau(t) \odot W_1^\tau(t)\tilde{X}\right]\right]\right]\right). \tag{D.10}$$

Since $m\|\Xi^\tau\|_F^2$ is the loss function we optimize, it does not increase under gradient flow. Hence, since multiplying by D_l^τ elementwise does not increase Frobenius norm,

$$m\|\Xi^\tau(t)\|_F \le m\|\Xi^\tau(0)\|_F \le \|Y\|_F + \|W_1^\tau(0)\tilde{X}\|_F \prod_{l=2}^{L}\|W_l^\tau(0)\|. \tag{D.11}$$

We have $\|W_1^\tau(0)\tilde{X}\|_F = \rho\|W_1^\tau(0)\tilde{X}\| \le \|W_1^\tau(0)\|\|\tilde{X}\|$. Since all y from $\operatorname{supp}\mathcal{D}$ are ± 1 and $\tilde{X}\tilde{X}^\top = mI_d$, we get $\|\Xi^\tau(t)\|_F = \frac{1}{\sqrt{m}}(1+\rho\beta^L)$.

Due to Assumption 6.5.1, $m\|\Xi^\tau\tilde{X}^\top\|_F^2$ does not increase under gradient flow:

$$m\|\Xi^\tau(t)\tilde{X}^\top\|_F \le m\|\Xi^\tau(0)\tilde{X}^\top\|_F \le \|Y\tilde{X}^\top\|_F + \|\tilde{X}\|\|W_1^\tau(0)\tilde{X}\|_F \prod_{l=2}^{L}\|W_l^\tau(0)\|. \tag{D.12}$$

Since $Y\tilde{X}^+ = s$, we have $Y\tilde{X}^\top = ms$. Therefore $\|\Xi^\tau(t)\tilde{X}^\top\|_F = s + \rho\beta^L$.

Finally, since $\Xi^\tau \in \mathbb{R}^{1\times m}$, $\|\Xi^\tau\| = \|\Xi^\tau\|_F$ and $\|\Xi^\tau\tilde{X}^\top\| = \|\Xi^\tau\tilde{X}^\top\|_F$.

D.3.3 Bounding derivatives in the proof of Lemma 6.7.1

Let us start with W_1^τ:

$$\frac{d^2W_1^\tau}{dtd\tau} = \left[D^\tau \odot \left(\frac{dW_2^{\tau,\top}}{d\tau}\Xi^\tau + W_2^{\tau,\top}\frac{d\Xi^\tau}{d\tau}\right)\right]\tilde{X}^\top - \left[\bar{\Delta} \odot W_2^{\tau,\top}\Xi^\tau\right]\tilde{X}^\top; \tag{D.13}$$

$$\left\|\frac{d^2W_1^\tau}{dtd\tau}\right\| \le \left\|\frac{dW_2^\tau}{d\tau}\right\|\left((1-\tau)\|\Xi^\tau\tilde{X}^\top\| + \tau\|\Xi^\tau\|_F\|\tilde{X}\|\right) + \|W_2^\tau\|\left(\left\|\frac{d\Xi^\tau}{d\tau}\right\|_F + \|\Xi^\tau\|_F\right)\|\tilde{X}\|. \tag{D.14}$$

We need a bound for $\left\|\frac{d\Xi^\tau}{d\tau}\right\|$:

$$m\frac{d\Xi^\tau}{d\tau} = W_2^\tau\left[\bar{\Delta} \odot W_1^\tau\tilde{X}\right] - W_2^\tau\left[D^\tau \odot \frac{dW_1^\tau}{d\tau}\tilde{X}\right] - \frac{dW_2^\tau}{d\tau}\left[D^\tau \odot W_1^\tau\tilde{X}\right]; \tag{D.15}$$

$$\begin{aligned} m\left\|\frac{d\Xi^\tau}{d\tau}\right\| &\le \|W_2^\tau\|\|W_1^\tau\tilde{X}\|_F + \|W_2^\tau\|\left\|\frac{dW_1^\tau}{d\tau}\tilde{X}\right\|_F + \left\|\frac{dW_2^\tau}{d\tau}\right\|\|W_1^\tau\tilde{X}\|_F \\ &\le u^2\rho\sqrt{m} + u\left\|\frac{dW_1^\tau}{d\tau}\tilde{X}\right\|_F + u\rho\sqrt{m}\left\|\frac{dW_2^\tau}{d\tau}\right\|. \end{aligned} \tag{D.16}$$

This results in

$$\frac{d}{dt}\left\|\frac{dW_1^\tau}{d\tau}\right\| \le \left\|\frac{d^2W_1^\tau}{dtd\tau}\right\| \le u\left(1 + \rho\beta^2 + \rho u^2 + u\frac{1}{\sqrt{m}}\left\|\frac{dW_1^\tau}{d\tau}\tilde{X}\right\|_F\right) + \left\|\frac{dW_2^\tau}{d\tau}\right\|(s + (1-s)\tau + \rho\beta^2 + \rho u^2). \tag{D.17}$$

Similarly, we have an evolution of $W_1^\tau \tilde{X}$:

$$\begin{aligned}\frac{d}{dt}\left\|\frac{dW_1^\tau}{d\tau}\tilde{X}\right\|_F &\le \left\|\frac{dW_2^\tau}{d\tau}\right\|\left((1-\tau)\|\Xi^\tau\tilde{X}^\top\|_F\|\tilde{X}\| + \tau\|\Xi^\tau\|_F\|\tilde{X}^\top\tilde{X}\|\right) + \|W_2^\tau\|\left(\left\|\frac{d\Xi^\tau}{d\tau}\right\|_F + \|\Xi^\tau\|_F\right)\|\tilde{X}^\top\tilde{X}\| \\ &\le u\left(1+\rho\beta^2+\rho u^2 + u\frac{1}{\sqrt{m}}\left\|\frac{dW_1^\tau}{d\tau}\tilde{X}\right\|_F\right)\sqrt{m} + \left\|\frac{dW_2^\tau}{d\tau}\right\|(s+(1-s)\tau+\rho\beta^2+\rho u^2)\sqrt{m}.\end{aligned} \tag{D.18}$$

Finally, consider W_2^τ:

$$\frac{d^2W_2^\tau}{dtd\tau} = \frac{d\Xi^\tau}{d\tau}\left[D^\tau \odot W_1^\tau\tilde{X}\right]^\top + \Xi^\tau\left[D^\tau\odot\frac{dW_1^\tau}{d\tau}\tilde{X} - \bar{\Delta}\odot W_1^\tau\tilde{X}\right]^\top; \tag{D.19}$$

$$\left\|\frac{d^2W_2^\tau}{dtd\tau}\right\| \le \left\|\frac{dW_1^\tau}{d\tau}\right\|\left((1-\tau)\|\Xi^\tau\tilde{X}^\top\| + \tau\|\Xi^\tau\|_F\|\tilde{X}\|\right) + \left(\|\Xi^\tau\| + \left\|\frac{d\Xi^\tau}{d\tau}\right\|\right)\|W_1^\tau\tilde{X}\|_F; \tag{D.20}$$

$$\begin{aligned}\frac{d}{dt}\left\|\frac{dW_2^\tau}{d\tau}\right\| &\le \left\|\frac{d^2W_2^\tau}{dtd\tau}\right\| \\ &\le u\left(1+\rho\beta^2+\rho u^2+u\frac{1}{\sqrt{m}}\left\|\frac{dW_1^\tau}{d\tau}\tilde{X}\right\|_F\right) + \left\|\frac{dW_1^\tau}{d\tau}\right\|(s+(1-s)\tau+\rho\beta^2)+\rho u^2\left\|\frac{dW_2^\tau}{d\tau}\right\|.\end{aligned} \tag{D.21}$$

We see that $\left\|\frac{dW_1^\tau}{d\tau}\right\|$, $\frac{1}{\sqrt{m}}\left\|\frac{dW_1^\tau}{d\tau}\tilde{X}\right\|_F$, and $\left\|\frac{dW_2^\tau}{d\tau}\right\|$ are all bounded by the same v which satisfies

$$\frac{dv(t)}{dt} = v(t)(\bar{s}+(1+\rho)u^2(t)) + u(t)(\bar{1}+\rho u^2(t)), \qquad v(0)=0, \tag{D.22}$$

where $\bar{s} = s+(1-s)\tau+\rho\beta^2$ and $\bar{1} = 1+\rho\beta^2$.

D.3.4 Solving the ODE for $v(t)$

Recall $u(t) = \bar{\beta}e^{\bar{s}t}$. We solve the homogeneous equation to get

$$v(t) = C(t)e^{\bar{s}t+\frac{1+\rho}{2\bar{s}}\bar{\beta}^2e^{2\bar{s}t}} = C(t)e^{(L-1)\ln u(t)+\frac{1+(L-1)\rho}{\bar{s}L}u^L(t)} = C(t)\bar{\beta}^{-1}u(t)e^{\frac{1+\rho}{2\bar{s}}u^2(t)}, \tag{D.23}$$

where $C(t)$ satisfies

$$\frac{dC(t)}{dt}u(t)e^{\frac{1+\rho}{2\bar{s}}u^2(t)} = \bar{\beta}u(t)\left[\bar{1}+\rho u^2(t)\right]. \tag{D.24}$$

Recall for $L=2$, $\hat{s} = \frac{2}{1+\rho}\bar{s}$. Then

$$\begin{aligned}C(t) &= \bar{\beta}\int e^{-\frac{u^2(t)}{\hat{s}}}\left[\bar{1}+\rho u^2(t)\right]dt \\ &= \frac{\bar{\beta}}{2}\left[\frac{\bar{1}}{\bar{s}}\left(\mathrm{Ei}\left(-\frac{u^2(t)}{\hat{s}}\right) - \mathrm{Ei}\left(-\frac{\bar{\beta}^2}{\hat{s}}\right)\right) - \rho\frac{\hat{s}}{\bar{s}}\left(e^{-\frac{u^2(t)}{\hat{s}}} - e^{-\frac{\bar{\beta}^2}{\hat{s}}}\right)\right].\end{aligned} \tag{D.25}$$

This gives the final solution:

$$v(t) = \frac{1}{2}u(t)[w(u(t)) - w(\beta)]e^{\frac{u^2(t)}{\hat{s}}}, \tag{D.26}$$

where we took

$$w(u) = \frac{\bar{1}}{\bar{s}}\,\mathrm{Ei}\left(-\frac{u^2(t)}{\hat{s}}\right) - \frac{2\rho}{1+\rho}e^{-\frac{u^2(t)}{\hat{s}}}. \tag{D.27}$$

D.4 Deep networks

D.4.1 Proof of Lemma 6.7.1 for $L \geq 3$

Bounding weight norms. For $l \in [2, L]$,

$$\frac{dW_l^\tau}{dt} = \left[D_l^\tau \odot W_{l+1}^{\tau,\top} \dots \left[D_{L-1}^\tau \odot W_L^{\tau,\top} \Xi^\tau\right]\right] \left[D_{l-1}^\tau \odot W_{l-1}^\tau \dots \left[D_1^\tau \odot W_1^\tau \tilde{X}\right]\right]^\top; \tag{D.28}$$

$$\left\|\frac{dW_l^\tau}{dt}\right\| \leq \left((1-\tau)\|\Xi^\tau \tilde{X}^\top\|\|W_1^\tau\| + \tau(L-1)\|\Xi^\tau\|_F \|W_1^\tau \tilde{X}\|_F\right) \prod_{k\in[2,L]\setminus\{l\}} \|W_k^\tau\|. \tag{D.29}$$

For $l = 1$,

$$\frac{dW_1^\tau}{dt} = \left[D_1^\tau \odot W_2^{\tau,\top} \dots \left[D_{L-1}^\tau \odot W_L^{\tau,\top} \Xi^\tau\right]\right] \tilde{X}^\top; \tag{D.30}$$

$$\left\|\frac{dW_1^\tau}{dt}\right\| \leq \left((1-\tau)\|\Xi^\tau \tilde{X}^\top\| + \tau(L-1)\|\Xi^\tau\|_F \|\tilde{X}\|\right) \prod_{k\in[2,L]} \|W_k^\tau\|; \tag{D.31}$$

$$\left\|\frac{dW_1^\tau \tilde{X}}{dt}\right\|_F \leq \left((1-\tau)\|\Xi^\tau \tilde{X}^\top\|_F \|\tilde{X}\| + \tau(L-1)\|\Xi^\tau\|_F \|\tilde{X}^\top \tilde{X}\|\right) \prod_{k\in[2,L]} \|W_k^\tau\|. \tag{D.32}$$

Using Lemma 6.7.4, we get $(1-\tau)\|\Xi^\tau \tilde{X}^\top\|\|W_1^\tau\| + \tau(L-1)\|\Xi^\tau\|\|W_1^\tau \tilde{X}\|_F \leq g_1(t)$ and $\|W_l^\tau\| \leq g_2(t)$ $\forall l \in [2:L]$, where

$$\frac{dg_1(t)}{dt} = \bar{s}^2 g_2^{L-1}(t), \quad \frac{dg_2(t)}{dt} = g_1(t) g_2^{L-2}(t), \qquad g_1(0) = \beta((1-\tau)(s+\beta^L) + \tau(L-1)(1+\beta^L)\rho), \quad g_2(0) = \beta. \tag{D.33}$$

We have the following first integral:

$$\frac{d}{dt}\left(g_1^2(t) - \bar{s}^2 g_2^2(t)\right) = 0, \qquad g_1^2(0) - \bar{s}^2 g_2^2(0) = \beta^2\left(((1-\tau)(s+\beta^L) + \tau(L-1)(1+\beta^L)\rho)^2 - \bar{s}^2\right). \tag{D.34}$$

Therefore

$$\frac{dg_2(t)}{dt} = g_2^{L-2}(t)\sqrt{\bar{s}^2 g_2^2(t) - \bar{s}^2 g_2^2(0) + g_1^2(0)}, \qquad g_2(0) = \beta. \tag{D.35}$$

Suppose $\rho > 1$. Then $g_1^2(0) - \bar{s}^2 g_2^2(0) > 0$ and the solution is given in the following implicit form:

$$t = \frac{g_2^{3-L}(t) {}_2F_1\left(\frac{1}{2}, \frac{3-L}{2}, \frac{5-L}{2}, -\frac{\bar{s}^2 g_2^2(t)}{g_1^2(0) - \bar{s}^2\beta^2}\right) - \beta_2^{3-L} F_1\left(\frac{1}{2}, \frac{3-L}{2}, \frac{5-L}{2}, -\frac{\bar{s}^2\beta^2}{g_1^2(0) - \bar{s}^2\beta^2}\right)}{(3-L)\sqrt{g_1^2(0) - \bar{s}^2\beta^2}}. \tag{D.36}$$

The above expression cannot be made explicit for general L (but we could get explicit expression for $L \in \{2, 3, 4\}$). As an alternative, we consider a looser bound $u(t)$:

$$\frac{du(t)}{dt} = \bar{s}_1 u^{L-1}(t), \qquad u(0) = \beta\frac{\bar{s}_\rho}{\bar{s}_1}, \tag{D.37}$$

where $\bar{s}_\rho := (1-\tau)(s+\beta^L) + \tau(L-1)(1+\beta^L)\rho$; note that $\bar{s}_1 = \bar{s}$. We have $u(t) \geq g_2(t)$ and $\bar{s}u(t) \geq g_1(t)\ \forall t \geq 0$. This ODE solves as

$$u(t) = \left(\bar{s}(2-L)t + \bar{\beta}^{2-L}\right)^{\frac{1}{2-L}}, \tag{D.38}$$

where $\bar{\beta} = \beta\frac{\bar{s}_\rho}{\bar{s}_1}$. Note that the solution exists only for $t < \frac{\bar{\beta}^{2-L}}{(L-2)\bar{s}}$.

For the input layer weights, we get

$$\frac{d\|W_1^\tau(t)\|}{dt} \leq \bar{s}u^{L-1}(t). \tag{D.39}$$

The solution is given by

$$\|W_1^\tau(t)\| \leq \beta + \bar{s}\int_0^t \left(\bar{s}(2-L)t + \bar{\beta}^{2-L}\right)^{\frac{L-1}{2-L}} dt = \beta - \bar{\beta} + \left(\bar{s}(2-L)t + \bar{\beta}^{2-L}\right)^{\frac{1}{2-L}} = \beta - \bar{\beta} + u(t). \tag{D.40}$$

Similarly, we have

$$\frac{1}{\sqrt{m}}\|W_1^\tau(t)\tilde{X}\|_F \leq \beta\rho + \bar{s}\int_0^t \left(\bar{s}(2-L)t + \bar{\beta}^{2-L}\right)^{\frac{L-1}{2-L}} dt = \beta\rho - \bar{\beta} + u(t). \tag{D.41}$$

For brevity, we define

$$b = \beta - \bar{\beta} = \beta\left(1 - \frac{\bar{s}_\rho}{\bar{s}_1}\right) = -\beta\tau(L-1)(\rho-1)\frac{1+\beta^L}{\bar{s}}, \tag{D.42}$$

and

$$b_\rho = \beta\rho - \bar{\beta} = \beta\left(\rho - \frac{\bar{s}_\rho}{\bar{s}_1}\right) = \beta(1-\tau)(\rho-1)\frac{s+\beta^L}{\bar{s}}. \tag{D.43}$$

As a simpler option, we could just say $\|W_1^\tau(t)\| \leq u(t)$ and $\frac{1}{\sqrt{m}}\|W_1^\tau(t)\tilde{X}\| \leq \rho u(t)$.

Bounding norms of weight derivatives. Recall the definition of Ξ^τ:

$$\Xi^\tau = \frac{1}{m}\left(Y - W_L^\tau\left[D_{L-1}^\tau \odot W_{L-1}^\tau \ldots \left[D_1^\tau \odot W_1^\tau\tilde{X}\right]\right]\right); \tag{D.44}$$

$$\begin{aligned} m\left\|\frac{d\Xi^\tau}{d\tau}\right\| &\leq (L-1)\|W_1^\tau\tilde{X}\|_F\prod_{l=2}^L\|W_l^\tau\| + \left\|\frac{dW_1^\tau\tilde{X}}{d\tau}\right\|\prod_{l=2}^L\|W_l^\tau\| + \|W_1^\tau\tilde{X}\|_F\sum_{k=2}^L\left\|\frac{dW_k^\tau}{d\tau}\right\|\prod_{l\in[2:L]\setminus\{k\}}\|W_l^\tau\| \\ &\leq u^{L-1}\left[(L-1)\sqrt{\rho m}u + \left\|\frac{dW_1^\tau\tilde{X}}{d\tau}\right\|_F + \sqrt{\rho m}\sum_{k=2}^L\left\|\frac{dW_k^\tau}{d\tau}\right\|\right]. \end{aligned} \tag{D.45}$$

For $l \in [2, L]$,

$$\begin{aligned}
\left\|\frac{d^2 W_l^\tau}{dtd\tau}\right\| &\le \left(\left\|\frac{d\Xi^\tau}{d\tau}\right\|_F + (L-1)\|\Xi^\tau\|_F\right)\|W_1^\tau \tilde{X}\|_F \prod_{k\in[2,L]\setminus\{l\}}\|W_k^\tau\| \\
&+ \sum_{j\in[2:L]\setminus\{l\}}\left((1-\tau)\|\Xi^\tau \tilde{X}^\top\|\|W_1^\tau\| + \tau(L-1)\|\Xi^\tau\|_F\|W_1^\tau\tilde{X}\|_F\right)\left\|\frac{dW_j^\tau}{d\tau}\right\|\prod_{k\in[2,L]\setminus\{l,j\}}\|W_k^\tau\| \\
&+ \left((1-\tau)\|\Xi^\tau\tilde{X}^\top\|\left\|\frac{dW_1^\tau}{d\tau}\right\| + \tau(L-1)\|\Xi^\tau\|_F\left\|\frac{dW_1^\tau\tilde{X}}{d\tau}\right\|_F\right)\prod_{k\in[2,L]\setminus\{l\}}\|W_k^\tau\| \\
&\le \left(u^{L-1}\left[(L-1)\rho u + \frac{1}{\sqrt{m}}\left\|\frac{dW_1^\tau\tilde{X}}{d\tau}\right\|_F + \rho\sum_{k=2}^{L}\left\|\frac{dW_k^\tau}{d\tau}\right\|\right] + \bar{1}(L-1)\right)\rho u^{L-1} \\
&+ \bar{s}u^{L-2}\sum_{j\in[2:L]\setminus\{l\}}\left\|\frac{dW_j^\tau}{d\tau}\right\| + \left((1-\tau)(s+\beta^L)\left\|\frac{dW_1^\tau}{d\tau}\right\| + \tau\bar{1}(L-1)\frac{1}{\sqrt{m}}\left\|\frac{dW_1^\tau\tilde{X}}{d\tau}\right\|_F\right)u^{L-2}.
\end{aligned} \tag{D.46}$$

For $l = 1$,

$$\begin{aligned}
\left\|\frac{d^2 W_1^\tau}{dtd\tau}\right\| &\le \left(\left\|\frac{d\Xi^\tau}{d\tau}\right\|_F + (L-1)\|\Xi^\tau\|_F\right)\|\tilde{X}\|\prod_{k\in[2,L]}\|W_k^\tau\| \\
&+ \sum_{j=2}^{L}\left((1-\tau)\|\Xi^\tau\tilde{X}^\top\| + \tau(L-1)\|\Xi^\tau\|_F\|\tilde{X}\|\right)\left\|\frac{dW_j^\tau}{d\tau}\right\|\prod_{k\in[2,L]\setminus\{j\}}\|W_k^\tau\| \\
&\le \left(u^{L-1}\left[(L-1)\rho u + \frac{1}{\sqrt{m}}\left\|\frac{dW_1^\tau\tilde{X}}{d\tau}\right\|_F + \rho\sum_{k=2}^{L}\left\|\frac{dW_k^\tau}{d\tau}\right\|\right] + \bar{1}(L-1)\right)u^{L-1} + \bar{s}u^{L-2}\sum_{j=2}^{L}\left\|\frac{dW_j^\tau}{d\tau}\right\|.
\end{aligned} \tag{D.47}$$

$\left\|\frac{dW_l^\tau}{d\tau}\right\|$ $\forall l \in [L]$, as well as $\frac{1}{\sqrt{m}}\left\|\frac{dW_1^\tau\tilde{X}}{d\tau}\right\|_F$, are all bounded by $v(t)$ which satisfies

$$\begin{aligned}
\frac{dv}{dt} &= \left[(L-1)\rho u^L + (1+(L-1)\rho)u^{L-1}v + (L-1)\bar{1}\right]u^{L-1} + \bar{s}(L-1)u^{L-2}v \\
&= v\left[(1+(L-1)\rho)u^{2L-2} + \bar{s}(L-1)u^{L-2}\right] + (L-1)u^{L-1}\left[\bar{1} + \rho u^L\right], \qquad v(0) = 0,
\end{aligned} \tag{D.48}$$

where $\bar{1} = 1 + \beta^L$.

Solving the ODE for v(t). Recall

$$u(t) = \left(\bar{s}(2-L)t + \bar{\beta}^{2-L}\right)^{\frac{1}{2-L}}. \tag{D.49}$$

We solve the homogeneous equation to get

$$\begin{aligned}
v(t) &= C(t)e^{\frac{L-1}{2-L}\ln\left(\bar{s}(2-L)t+\bar{\beta}^{2-L}\right)+\frac{1+(L-1)\rho}{\bar{s}L}\left(\bar{s}(2-L)t+\bar{\beta}^{2-L}\right)^{\frac{L}{2-L}}} \\
&= C(t)e^{(L-1)\ln u(t)+\frac{1+(L-1)\rho}{\bar{s}L}u^L(t)} = C(t)u^{L-1}(t)e^{\frac{1+(L-1)\rho}{\bar{s}L}u^L(t)},
\end{aligned} \tag{D.50}$$

where $C(t)$ satisfies

$$\frac{dC(t)}{dt}u^{L-1}(t)e^{\frac{1+(L-1)\rho}{\bar{s}L}u^L(t)} = (L-1)u^{L-1}(t)\left[\bar{1}+\rho u^L(t)\right]. \tag{D.51}$$

Let us introduce $\hat{s} = \frac{L}{1+(L-1)\rho}\bar{s}$. Then

$$\begin{aligned} C(t) &= (L-1)\int e^{-\frac{u^L(t)}{\hat{s}}}\left[\bar{1}+\rho u^L(t)\right]\,dt \\ &= \frac{L-1}{L}\hat{s}^{\frac{2-L}{L}}\left[\frac{\bar{1}}{\bar{s}}\left(\Gamma\left(\frac{2-L}{L},\frac{\beta^L}{\hat{s}}\right)-\Gamma\left(\frac{2-L}{L},\frac{u^L(t)}{\hat{s}}\right)\right)+\rho\frac{\hat{s}}{\bar{s}}\left(\Gamma\left(\frac{2}{L},\frac{\beta^L}{\hat{s}}\right)-\Gamma\left(\frac{2}{L},\frac{u^L(t)}{\hat{s}}\right)\right)\right]. \end{aligned} \tag{D.52}$$

This gives the final solution:

$$v(t) = \frac{L-1}{L}\hat{s}^{\frac{2-L}{L}}u^{L-1}(t)[w(u(t))-w(\beta)]e^{\frac{u^L(t)}{\hat{s}}}, \tag{D.53}$$

where we took

$$w(u) = -\frac{\bar{1}}{\bar{s}}\Gamma\left(\frac{2-L}{L},\frac{u^L}{\hat{s}}\right)-\frac{L\rho}{1+(L-1)\rho}\Gamma\left(\frac{2}{L},\frac{u^L}{\hat{s}}\right). \tag{D.54}$$

D.4.2 Evaluating the solution at the learning time

For a properly initialized linear network, $\forall l\in[L]\ \|W_l^0(t)\| = \bar{u}(t)$, where $u(t)$ satisfies (Saxe et al., 2013)

$$\frac{d\bar{u}(t)}{dt} = \bar{u}^{L-1}(t)(s-\bar{u}^L(t)), \qquad \bar{u}(0)=\beta, \tag{D.55}$$

which gives the solution in implicit form[2]:

$$t_\alpha^*(\beta) = \int\frac{d\bar{u}}{\bar{u}^{L-1}(s-\bar{u}^L)} = \frac{\bar{u}^{2-L}(t)}{s(2-L)}\,{}_2F_1\left(1,\frac{2}{L}-1,\frac{2}{L};\frac{\bar{u}^L(t)}{s}\right)-\frac{\beta^{2-L}}{s(2-L)}\,{}_2F_1\left(1,\frac{2}{L}-1,\frac{2}{L};\frac{\beta^L}{s}\right). \tag{D.56}$$

Suppose we are going to learn a fixed fraction of the data, i.e. take $\bar{u}(t) = (\alpha s)^{1/L}$ for $\alpha\in(0,1)$. Then

$$t_\alpha^*(\beta) = \frac{\beta^{2-L}\,{}_2F_1\left(1,\frac{2}{L}-1,\frac{2}{L};\frac{\beta^L}{s}\right)-(\alpha s)^{\frac{2}{L}-1}\,{}_2F_1\left(1,\frac{2}{L}-1,\frac{2}{L};\alpha\right)}{s(L-2)}. \tag{D.57}$$

Since $t_\alpha^*(\beta)$ is the time sufficient to learn a network for $\epsilon=0$, we suppose it also suffices to learn a nonlinear network. So, we are going to evaluate our bound at $t=t_\alpha^*$. Since we need $\hat{R}_\gamma(t_\alpha^*)<1$ for the bound to be non-vacuous, we should take γ small relative to α. We consider $\gamma=\alpha^\nu/q$ for $\nu,q\geq 1$.

Since the linear network learning time $t_\alpha^*(\beta)$ is correct for almost all initialization only when β vanishes, we are going to work in the limit of $\beta\to 0$. Since we need $\alpha\in(\beta^L/s,1)$, otherwise the linear training time is negative, we take $\alpha=\frac{r}{s}\beta^\lambda$ for $\lambda\in(0,L]$ and $r>1$.

[2] ${}_2F_1$ is a hypergeometric function defined as a series ${}_2F_1(a,b,c,z) = 1+\sum_{k=1}^\infty\frac{(a)_k(b)_k}{(c)_k}\frac{z^k}{k!}$, where $(q)_k = q(q+1)\dots(q+k-1)$.

Consider first $\lambda < L$:

$$t_\alpha^*(\beta) = \frac{\beta^{2-L}}{s(L-2)} + O(\beta^{2\lambda/L}). \tag{D.58}$$

Let us evaluate u at this time:

$$u(t_\alpha^*(\beta)) = \left(\bar{\beta}^{2-L} - \frac{\bar{s}}{s}\beta^{2-L} + O(\beta^{2\lambda/L})\right)^{\frac{1}{2-L}} = \beta\left(\frac{\bar{s}_\rho^{2-L}}{\bar{s}_1^{2-L}} - \frac{\bar{s}}{s}\right)^{\frac{1}{2-L}}\left(1 + O\left(\beta^{L-2+2\lambda/L}\right)\right). \tag{D.59}$$

Apparently, this expression does not make sense for $\rho = 1$ and even close to it, so we switch to $\lambda = L$, which we expect to be the right exponent:

$$t_\alpha^*(\beta) = \beta^{2-L}\frac{1 - r^{\frac{2}{L}-1}}{s(L-2)} + O(\beta^2). \tag{D.60}$$

Let us evaluate u at this time:

$$u(t_\alpha^*(\beta)) = \left(\bar{\beta}^{2-L} - \frac{\bar{s}}{s}\left(1 - r^{\frac{2}{L}-1}\right)\beta^{2-L} + O(\beta^2)\right)^{\frac{1}{2-L}} = \beta\left(\frac{\bar{s}_\rho^{2-L}}{\bar{s}_1^{2-L}} - \frac{\bar{s}}{s}\left(1 - r^{\frac{2}{L}-1}\right)\right)^{\frac{1}{2-L}}\left(1 + O\left(\beta^L\right)\right). \tag{D.61}$$

This expression makes sense whenever $\frac{\bar{s}_\rho^{2-L}}{\bar{s}_1^{2-L}} - \frac{\bar{s}}{s}\left(1 - r^{\frac{2}{L}-1}\right) > 0$, i.e. when r is close enough to 1.

Let us evaluate w at the training time now. Since $\Gamma(a, x) = \Gamma(a) - \frac{x^a}{a} + O(x^{a+1})$, we get

$$\begin{aligned} w(u(t_\alpha^*(\beta))) - w(\beta) &= \frac{\bar{1}}{\bar{s}}\frac{L}{2-L}\hat{s}^{\frac{L-2}{L}}\left(u^{2-L}(t_\alpha^*(\beta)) - \beta^{2-L}\right) + O(\beta^2). \\ &= \frac{\bar{1}}{\bar{s}}\frac{L}{2-L}\hat{s}^{\frac{L-2}{L}}\beta^{2-L}\left(\frac{\bar{s}_\rho^{2-L}}{\bar{s}_1^{2-L}} - \frac{\bar{s}}{s}\left(1 - r^{\frac{2}{L}-1}\right) - 1\right) + O(\beta^2). \end{aligned} \tag{D.62}$$

Then the quantity of interest becomes

$$\begin{aligned} \frac{Lu^{L-1}(t_\alpha^*(\beta))v(t_\alpha^*(\beta))}{\gamma} &= \frac{L-1}{\gamma}u^{2L-2}(t_\alpha^*(\beta))\frac{\bar{1}}{\bar{s}}\frac{L}{2-L}\beta^{2-L}\left(\frac{\bar{s}_\rho^{2-L}}{\bar{s}_1^{2-L}} - \frac{\bar{s}}{s}\left(1 - r^{\frac{2}{L}-1}\right) - 1\right)(1 + O(\beta^L)) \\ = \frac{qs^\nu}{r^\nu}\frac{\bar{1}}{\bar{s}}\frac{L(L-1)}{2-L}\beta^{L(1-\nu)}&\left(\frac{\bar{s}_\rho^{2-L}}{\bar{s}_1^{2-L}} - \frac{\bar{s}}{s}\left(1 - r^{\frac{2}{L}-1}\right)\right)^{\frac{2L-2}{2-L}}\left(\frac{\bar{s}_\rho^{2-L}}{\bar{s}_1^{2-L}} - \frac{\bar{s}}{s}\left(1 - r^{\frac{2}{L}-1}\right) - 1\right)(1 + O(\beta^L)). \end{aligned} \tag{D.63}$$

This expression does not diverge as $\beta \to 0$ when $\nu = 1$. We will also have a finite $\lim_{L\to\infty}\lim_{\beta\to 0}$ whenever $\epsilon \propto (L-1)^{-1}$.

D.5 Proving Assumption 6.5.1 for linear nets

We have for $l = 1$,

$$\nabla_1 := \frac{1}{2m}\frac{\partial\left\|Y - f_{\theta^\epsilon}^\epsilon(\tilde{X})\right\|_F^2}{\partial W_1^\epsilon} = -\left[D_1^\epsilon \odot W_2^{\epsilon,\top} \dots \left[D_{L-1}^\epsilon \odot W_L^{\epsilon,\top}\Xi^\epsilon\right]\right]\tilde{X}^\top. \tag{D.64}$$

For $l \in [2, L]$,

$$\nabla_l := \frac{1}{2m} \frac{\partial \left\| Y - f^\epsilon_{\theta^\epsilon}(\tilde{X}) \right\|_F^2}{\partial W^\epsilon_l} = - \left[D^\epsilon_l \odot W^{\epsilon,\top}_{l+1} \dots \left[D^\epsilon_{L-1} \odot W^{\epsilon,\top}_L \Xi^\epsilon \right] \right] \left[D^\epsilon_{l-1} \odot W^\epsilon_{l-1} \dots \left[D^\epsilon_1 \odot W^\epsilon_1 \tilde{X} \right] \right]^\top. \tag{D.65}$$

We also have

$$\nabla^X_1 := \frac{1}{2m} \frac{\partial \left\| \left(Y - f^\epsilon_{\theta^\epsilon}(\tilde{X}) \right) \tilde{X}^\top \right\|_F^2}{\partial W^\epsilon_1} = - \left[D^\epsilon_1 \odot W^{\epsilon,\top}_2 \dots \left[D^\epsilon_{L-1} \odot W^{\epsilon,\top}_L \Xi^\epsilon \tilde{X}^\top \tilde{X} \right] \right] \tilde{X}^\top. \tag{D.66}$$

$$\begin{aligned} \nabla^X_l &:= \frac{1}{2m} \frac{\partial \left\| \left(Y - f^\epsilon_{\theta^\epsilon}(\tilde{X}) \right) \tilde{X}^\top \right\|_F^2}{\partial W^\epsilon_l} \\ &= - \left[D^\epsilon_l \odot W^{\epsilon,\top}_{l+1} \dots \left[D^\epsilon_{L-1} \odot W^{\epsilon,\top}_L \Xi^\epsilon \tilde{X}^\top \tilde{X} \right] \right] \left[D^\epsilon_{l-1} \odot W^\epsilon_{l-1} \dots \left[D^\epsilon_1 \odot W^\epsilon_1 \tilde{X} \right] \right]^\top. \end{aligned} \tag{D.67}$$

The statement of Assumption 6.5.1 follows from

Conjecture D.5.1. $\forall \epsilon \in [0, 1]$ $\forall t \geq 0$ $\sum_{l=1}^L \operatorname{tr}\left[\nabla^X_l \nabla_l^\top \right] \geq 0$.

Indeed, the above conjecture states that loss gradients wrt weights and "projected" loss gradients wrt weights are positively aligned, so, whenever loss does not increase, neither does projected loss. Since we use gradient flow, loss is guaranteed to not increase. Below, we prove the conjecture for linear nets.

Proof of Conjecture D.5.1 for $\epsilon = 0$. Since ϵ is zero, we omit the corresponding sup-index:

$$\begin{aligned} \operatorname{tr}\left[\nabla^X_1 \nabla_1^\top \right] &= \operatorname{tr}\left[\left[W_2^\top \dots W_L^\top \Xi \tilde{X}^\top \tilde{X} \right] \tilde{X}^\top \tilde{X} \left[W_2^\top \dots W_L^\top \Xi \right]^\top \right] \\ &= \operatorname{tr}\left[\left[W_2^\top \dots W_L^\top \Xi \tilde{X}^\top \right] \left[W_2^\top \dots W_L^\top \Xi \tilde{X}^\top \right]^\top \right] = \operatorname{tr}\left[W_L \dots W_2 W_2^\top \dots W_L^\top \right] \operatorname{tr}\left[\Xi X^\top X \Xi^\top \right] \end{aligned} \tag{D.68}$$

by the circular property of trace, and the fact that Ξ is a matrix with a single row. Since $d_{out} = 1$, both traces are just squared Euclidean norms of vectors, hence they are non-negative: $\operatorname{tr}\left[\nabla^X_1 \nabla_1^\top \right] \geq 0$.

Let us do the same for the other layers:

$$\begin{aligned} \operatorname{tr}\left[\nabla^X_l \nabla_l^\top \right] &= \operatorname{tr}\left[\left[W_{l+1}^\top \dots W_L^\top \Xi \tilde{X}^\top \tilde{X} \right] \left[W_{l-1} \dots W_1 \tilde{X} \right]^\top \left[W_{l-1} \dots W_1 \tilde{X} \right] \left[W_{l+1}^\top \dots W_L^\top \Xi \right]^\top \right] \\ &= \operatorname{tr}\left[\left[W_{l+1}^\top \dots W_L^\top \Xi \tilde{X}^\top W_1^\top \dots W_{l-1}^\top \right] \left[W_{l+1}^\top \dots W_L^\top \Xi \tilde{X}^\top W_1^\top \dots W_{l-1}^\top \right]^\top \right] \\ &= \operatorname{tr}\left[W_L \dots W_{l+1} W_{l+1}^\top \dots W_L^\top \right] \operatorname{tr}\left[\Xi X^\top W_1^\top \dots W_{l-1}^\top W_{l-1} \dots W_1 X \Xi^\top \right] \geq 0 \end{aligned} \tag{D.69}$$

for the same reasons as before. □

Clearly, Conjecture D.5.1 should also hold for small enough ϵ whenever it holds for $\epsilon = 0$. However, the bound for the maximal ϵ for which we were able to guarantee the conjecture statement, vanishes with time t, as our weight bounds are too loose. For this reason, we do not include it here. See Section 6.6 for empirical validation of Assumption 6.5.1.

Bibliography

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. (2023). GPT-4 technical report. *arXiv preprint arXiv:2303.08774*.

Allen-Zhu, Z., Li, Y., and Song, Z. (2019). A convergence theory for deep learning via over-parameterization. In *International conference on machine learning*, pages 242–252. PMLR.

Araújo, D., Oliveira, R. I., and Yukimura, D. (2019). A mean-field limit for certain deep neural networks. *arXiv preprint arXiv:1906.00193*.

Arora, S., Du, S., Hu, W., Li, Z., and Wang, R. (2019a). Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In *International Conference on Machine Learning*, pages 322–332.

Arora, S., Du, S. S., Hu, W., Li, Z., Salakhutdinov, R. R., and Wang, R. (2019b). On exact computation with an infinitely wide neural net. In *Advances in Neural Information Processing Systems*, pages 8141–8150.

Arora, S., Du, S. S., Li, Z., Salakhutdinov, R., Wang, R., and Yu, D. (2020). Harnessing the power of infinitely wide deep nets on small-data tasks. In *International Conference on Learning Representations*.

Ba, J. L., Kiros, J. R., and E., H. G. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

Bach, F. (2017). Breaking the curse of dimensionality with convex neural networks. *Journal of Machine Learning Research*, 18(19):1–53.

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Bakhshizadeh, M., Maleki, A., and de la Pena, V. H. (2023). Sharp concentration results for heavy-tailed distributions. *Information and Inference: A Journal of the IMA*, 12(3):iaad011.

Bartlett, P. L., Foster, D. J., and Telgarsky, M. J. (2017). Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems*, pages 6240–6249.

Baymurzina, D., Golikov, E., and Burtsev, M. (2022). A review of neural architecture search. *Neurocomputing*, 474:82–93.

Bengio, Y., Lee, D.-H., Bornschein, J., Mesnard, T., and Lin, Z. (2015). Towards biologically plausible deep learning. *arXiv preprint arXiv:1502.04156*.

Bietti, A. and Mairal, J. (2019). On the inductive bias of Neural Tangent Kernels. *Advances in Neural Information Processing Systems*, 32.

Biggs, F. and Guedj, B. (2022). Non-vacuous generalisation bounds for shallow neural networks. In *International Conference on Machine Learning*, pages 1963–1981. PMLR.

Boltzmann, L. (1872). *Weitere studien über das wärmegleichgewicht unter gasmolekülen*, volume 166. Aus der kk Hot-und Staatsdruckerei.

Bombari, S., Amani, M. H., and Mondelli, M. (2022). Memorization and optimization in deep neural networks with minimum over-parameterization. *Advances in Neural Information Processing Systems*, 35:7628–7640.

Bordelon, B., Atanasov, A., and Pehlevan, C. (2024). A dynamical model of neural scaling laws. In *Forty-first International Conference on Machine Learning*.

Bordelon, B., Atanasov, A., and Pehlevan, C. (2025). How feature learning can improve neural scaling laws. In *The Thirteenth International Conference on Learning Representations*.

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. (2018). JAX: composable transformations of Python+NumPy programs.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Chen, L. and Xu, S. (2021). Deep Neural Tangent Kernel and Laplace kernel have the same RKHS. In *International Conference on Learning Representations*.

Chen, S., He, H., and Su, W. (2020). Label-aware Neural Tangent Kernel: Toward better generalization and local elasticity. *Advances in Neural Information Processing Systems*, 33:15847–15858.

Chen, W., Gong, X., and Wang, Z. (2021). Neural architecture search on imagenet in four GPU hours: A theoretically inspired perspective. In *International Conference on Learning Representations*.

Chen, W.-K. and Lam, W.-K. (2021). Universality of approximate message passing algorithms. *Electronic Journal of Probability*, 26:1–44.

Chizat, L. and Bach, F. (2018). On the global convergence of gradient descent for over-parameterized models using optimal transport. *Advances in neural information processing systems*, 31.

Chizat, L. and Bach, F. (2020). Implicit bias of gradient descent for wide two-layer neural networks trained with the logistic loss. In *Conference on learning theory*, pages 1305–1338. PMLR.

Chizat, L., Colombo, M., Fernández-Real, X., and Figalli, A. (2024). Infinite-width limit of deep linear neural networks. *Communications on Pure and Applied Mathematics*, 77(10):3958–4007.

Chizat, L., Oyallon, E., and Bach, F. (2019). On lazy training in supervised differentiable programming. In *Conference on Neural Information Processing Systems (NeurIPS)*.

Choromanska, A., Henaff, M., Mathieu, M., Arous, G. B., and LeCun, Y. (2015). The loss surfaces of multilayer networks. In *Artificial intelligence and statistics*, pages 192–204. PMLR.

Cohen, J. M., Kaur, S., Li, Y., Kolter, J. Z., and Talwalkar, A. (2021). Gradient descent on neural networks typically occurs at the edge of stability. *arXiv preprint arXiv:2103.00065*.

Dai, Z., Karzand, M., and Srebro, N. (2021). Representation costs of linear neural networks: Analysis and design. *Advances in Neural Information Processing Systems*, 34:26884–26896.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dong, X. and Yang, Y. (2020). NAS-Bench-201: Extending the scope of reproducible neural architecture search. In *International Conference on Learning Representations*.

Drew, J. H., Johnson, C. R., and Loewy, R. (1994). Completely positive matrices associated with M-matrices. *Linear and Multilinear Algebra*, 37(4):303–310.

Du, S., Lee, J., Li, H., Wang, L., and Zhai, X. (2019a). Gradient descent finds global minima of deep neural networks. In *International conference on machine learning*, pages 1675–1685. PMLR.

Du, S. S., Zhai, X., Poczos, B., and Singh, A. (2019b). Gradient descent provably optimizes over-parameterized neural networks. In *International Conference on Learning Representations*.

Dudeja, R., M. Lu, Y., and Sen, S. (2023). Universality of approximate message passing with semirandom matrices. *The Annals of Probability*, 51(5):1616–1683.

Dudley, R. M. (2018). *Real analysis and probability*. Chapman and Hall/CRC.

Dupuis, B. and Jacot, A. (2021). DNN-based topology optimisation: Spatial invariance and Neural Tangent Kernel. *Advances in Neural Information Processing Systems*, 34:27659–27669.

Dyer, E. and Gur-Ari, G. (2020). Asymptotics of wide networks from Feynman diagrams. In *International Conference on Learning Representations*.

Dziugaite, G. K. and Roy, D. M. (2017). Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. *arXiv preprint arXiv:1703.11008*.

Fort, S., Dziugaite, G. K., Paul, M., Kharaghani, S., Roy, D. M., and Ganguli, S. (2020). Deep learning versus kernel learning: an empirical study of loss landscape geometry and the time evolution of the Neural Tangent Kernel. *Advances in Neural Information Processing Systems*, 33:5850–5861.

Fukushima, K. (1969). Visual feature extraction by a multilayered network of analog threshold elements. *IEEE Transactions on Systems Science and Cybernetics*, 5(4):322–333.

Fukushima, K. (1988). Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural networks*, 1(2):119–130.

Galanti, T., Galanti, L., and Ben-Shaul, I. (2023). Comparative generalization bounds for deep neural networks. *Transactions on Machine Learning Research*.

Garriga-Alonso, A., Rasmussen, C. E., and Aitchison, L. (2019). Deep convolutional networks as shallow Gaussian processes. In *International Conference on Learning Representations*.

Geifman, A., Yadav, A., Kasten, Y., Galun, M., Jacobs, D., and Ronen, B. (2020). On the similarity between the Laplace and Neural Tangent Kernels. *Advances in Neural Information Processing Systems*, 33:1451–1461.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256.

Golikov, E. (2020a). Dynamically stable infinite-width limits of neural classifiers. *arXiv preprint arXiv:2006.06574*.

Golikov, E. (2020b). Notes on deep learning theory. *arXiv preprint arXiv:2012.05760*.

Golikov, E. (2020c). Towards a general theory of infinite-width limits of neural classifiers. In *International Conference on Machine Learning*, pages 3617–3626. PMLR.

Golikov, E. (2025). A generalization bound for nearly-linear networks. *Transactions on Machine Learning Research*.

Golikov, E., Pokonechnyy, E., and Korviakov, V. (2022). Neural Tangent Kernel: A survey. *arXiv preprint arXiv:2208.13614*.

Golikov, E. and Yang, G. (2022). Non-Gaussian Tensor Programs. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K., editors, *Advances in Neural Information Processing Systems*.

Gunasekar, S., Lee, J., Soudry, D., and Srebro, N. (2018a). Characterizing implicit bias in terms of optimization geometry. In *International Conference on Machine Learning*, pages 1832–1841. PMLR.

Gunasekar, S., Lee, J. D., Soudry, D., and Srebro, N. (2018b). Implicit bias of gradient descent on linear convolutional networks. *Advances in neural information processing systems*, 31.

Hanin, B. (2023). Random neural networks in the infinite width limit as Gaussian processes. *The Annals of Applied Probability*, 33(6A):4798–4819.

Hanin, B. and Nica, M. (2020a). Finite depth and width corrections to the Neural Tangent Kernel. In *International Conference on Learning Representations*.

Hanin, B. and Nica, M. (2020b). Products of many large random matrices and gradients in deep neural networks. *Communications in Mathematical Physics*, 376(1):287–322.

Hardy, M. (2006). Combinatorics of partial derivatives. *arXiv preprint math/0601149*.

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

He, K., Zhang, X., Ren, S., and Sun, J. (2016a). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

He, K., Zhang, X., Ren, S., and Sun, J. (2016b). Identity mappings in deep residual networks. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pages 630–645. Springer.

Hinton, G. E., Krizhevsky, A., and Sutskever, I. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25(1106-1114):1.

Hron, J., Bahri, Y., Sohl-Dickstein, J., and Novak, R. (2020). Infinite attention: NNGP and NTK for deep attention networks. In *International Conference on Machine Learning*, pages 4376–4386. PMLR.

Hu, W., Xiao, L., and Pennington, J. (2020). Provable benefit of orthogonal initialization in optimizing deep linear networks. In *International Conference on Learning Representations*.

Huang, J. and Yau, H.-T. (2020). Dynamics of deep neural networks and neural tangent hierarchy. In *International conference on machine learning*, pages 4542–4551. PMLR.

Hubel, D. H., Wiesel, T. N., et al. (1959). Receptive fields of single neurones in the cat's striate cortex. *J physiol*, 148(3):574–591.

Illing, B., Gerstner, W., and Brea, J. (2019). Biologically plausible deep learning – but how far can we go with shallow networks? *Neural Networks*, 118:90–101.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Ivakhnenko, A. G., Lapa, V. G., et al. (1966). Cybernetic predicting devices. *(No Title)*.

Jacot, A., Gabriel, F., and Hongler, C. (2018). Neural Tangent Kernel: Convergence and generalization in neural networks. In *Advances in neural information processing systems*, pages 8571–8580.

Jacot, A., Ged, F., Gabriel, F., Şimşek, B., and Hongler, C. (2021a). Deep linear networks dynamics: Low-rank biases induced by initialization scale and l_2 regularization. *arXiv preprint arXiv:2106.15933*.

Jacot, A., Ged, F., Şimşek, B., Hongler, C., and Gabriel, F. (2021b). Saddle-to-saddle dynamics in deep linear networks: Small initialization training, symmetry, and sparsity. *arXiv preprint arXiv:2106.15933*.

Jacot, A., Golikov, E., Hongler, C., and Gabriel, F. (2022). Feature learning in l_2-regularized DNNs: Attraction/repulsion and sparsity. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K., editors, *Advances in Neural Information Processing Systems*.

Kawaguchi, K., Deng, Z., Ji, X., and Huang, J. (2023). How does information bottleneck help deep learning? In *International Conference on Machine Learning*, pages 16049–16096. PMLR.

Kimeldorf, G. S. and Wahba, G. (1970). A correspondence between Bayesian estimation on stochastic processes and smoothing by splines. *The Annals of Mathematical Statistics*, 41(2):495–502.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *ICLR (Poster)*.

Kumar, T., Bordelon, B., Gershman, S. J., and Pehlevan, C. (2024). Grokking as the transition from lazy to rich training dynamics. In *The Twelfth International Conference on Learning Representations*.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Lee, J., Sohl-dickstein, J., Pennington, J., Novak, R., Schoenholz, S., and Bahri, Y. (2018). Deep neural networks as Gaussian processes. In *International Conference on Learning Representations*.

Lee, J., Xiao, L., Schoenholz, S., Bahri, Y., Novak, R., Sohl-Dickstein, J., and Pennington, J. (2019). Wide neural networks of any depth evolve as linear models under gradient descent. In *Advances in neural information processing systems*, pages 8572–8583.

Lee, J. D., Simchowitz, M., Jordan, M. I., and Recht, B. (2016). Gradient descent only converges to minimizers. In *Conference on learning theory*, pages 1246–1257.

Lewkowycz, A., Bahri, Y., Dyer, E., Sohl-Dickstein, J., and Gur-Ari, G. (2020). The large learning rate phase of deep learning: the catapult mechanism. *arXiv preprint arXiv:2003.02218*.

Li, M., Nica, M., and Roy, D. (2022). The neural covariance SDE: Shaped infinite depth-and-width networks at initialization. *Advances in Neural Information Processing Systems*, 35:10795–10808.

Li, Z., Luo, Y., and Lyu, K. (2021). Towards resolving the implicit bias of gradient descent for matrix factorization: Greedy low-rank learning. In *International Conference on Learning Representations*.

Marchenko, V. A. and Pastur, L. A. (1967). Распределение собственных значений в некоторых ансамблях случайных матриц. *Математический сборник*, 72(4):507–536.

Martens, J., Ballard, A., Desjardins, G., Swirszcz, G., Dalibard, V., Sohl-Dickstein, J., and Schoenholz, S. S. (2021). Rapid training of deep neural networks without skip connections or normalization layers using deep kernel shaping. *arXiv preprint arXiv:2110.01765*.

Matthews, A. G. d. G., Hron, J., Rowland, M., Turner, R. E., and Ghahramani, Z. (2018). Gaussian process behaviour in wide deep neural networks. In *International Conference on Learning Representations*.

McAllester, D. A. (1999). Some PAC-bayesian theorems. *Machine Learning*, 37(3):355–363.

Meanti, G., Carratino, L., Rosasco, L., and Rudi, A. (2020). Kernel methods through the roof: handling billions of points efficiently. *Advances in Neural Information Processing Systems*, 33:14410–14422.

Mei, S., Montanari, A., and Nguyen, P.-M. (2018). A mean field view of the landscape of two-layer neural networks. *Proceedings of the National Academy of Sciences*, 115(33):E7665–E7671.

Mertikopoulos, P., Hallak, N., Kavis, A., and Cevher, V. (2020). On the almost sure convergence of stochastic gradient descent in non-convex problems. *Advances in Neural Information Processing Systems*, 33:1117–1128.

Neal, R. M. (1995). *Bayesian Learning for Neural Networks*. PhD thesis, University of Toronto.

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

Nguyen, P.-M. (2019a). Mean field limit of the learning dynamics of multilayer neural networks. *arXiv preprint arXiv:1902.02880*.

Nguyen, P.-M. and Pham, H. T. (2023). A rigorous framework for the mean field limit of multilayer neural networks. *Mathematical Statistics and Learning*, 6(3):201–357.

Nguyen, Q. (2019b). On connected sublevel sets in deep learning. In *International Conference on Machine Learning*, pages 4790–4799.

Nguyen, Q. (2021). A note on connectivity of sublevel sets in deep learning. *arXiv preprint arXiv:2101.08576*.

Nguyen, Q. and Hein, M. (2017). The loss surface of deep and wide neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2603–2612.

Nguyen, Q. N. and Mondelli, M. (2020). Global convergence of deep networks with one wide layer followed by pyramidal topology. *Advances in Neural Information Processing Systems*, 33:11961–11972.

Noci, L., Li, C., Li, M., He, B., Hofmann, T., Maddison, C. J., and Roy, D. (2024). The shaped Transformer: Attention models in the infinite depth-and-width limit. *Advances in Neural Information Processing Systems*, 36.

Novak, R., Sohl-Dickstein, J., and Schoenholz, S. S. (2021). Fast finite width Neural Tangent Kernel. *Bayesian Deep Learning NeurIPS 2021 Workshop*.

Novak, R., Xiao, L., Bahri, Y., Lee, J., Yang, G., Abolafia, D. A., Pennington, J., and Sohl-dickstein, J. (2019). Bayesian deep convolutional networks with many channels are Gaussian processes. In *International Conference on Learning Representations*.

Novak, R., Xiao, L., Hron, J., Lee, J., Alemi, A. A., Sohl-Dickstein, J., and Schoenholz, S. S. (2020). Neural Tangents: Fast and easy infinite neural networks in Python. In *International Conference on Learning Representations*.

Obeid, D., Ramambason, H., and Pehlevan, C. (2019). Structured and deep similarity matching via structured and deep Hebbian networks. *Advances in neural information processing systems*, 32.

Ongie, G. and Willett, R. (2022). The role of linear layers in nonlinear interpolating networks. *arXiv preprint arXiv:2202.00856*.

Ongie, G., Willett, R., Soudry, D., and Srebro, N. (2020). A function space view of bounded norm infinite width ReLU nets: The multivariate case. In *International Conference on Learning Representations*.

Oymak, S. and Soltanolkotabi, M. (2020). Toward moderate overparameterization: Global convergence guarantees for training shallow neural networks. *IEEE Journal on Selected Areas in Information Theory*, 1(1):84–105.

Panageas, I. and Piliouras, G. (2017). Gradient descent only converges to minimizers: Non-isolated critical points and invariant regions. In *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Park, D. S., Lee, J., Peng, D., Cao, Y., and Sohl-Dickstein, J. (2020). Towards NNGP-guided neural architecture search. *arXiv preprint arXiv:2011.06006*.

Pastur, L. (2020). On random matrices arising in deep neural networks. Gaussian case. *arXiv preprint arXiv:2001.06188*.

Pastur, L. and Slavin, V. (2020). On random matrices arising in deep neural networks: General iid case. *arXiv preprint arXiv:2011.11439*.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Pehlevan, C., Sengupta, A. M., and Chklovskii, D. B. (2017). Why do similarity matching objectives lead to Hebbian/anti-Hebbian networks? *Neural computation*, 30(1):84–124.

Pennington, J. and Bahri, Y. (2017). Geometry of neural network loss surfaces via random matrix theory. In *International conference on machine learning*, pages 2798–2806. PMLR.

Pennington, J., Schoenholz, S., and Ganguli, S. (2017). Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice. In *Advances in neural information processing systems*, pages 4785–4795.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Pennington, J. and Worah, P. (2017). Nonlinear random matrix theory for deep learning. *Advances in neural information processing systems*, 30.

Poole, B., Lahiri, S., Raghu, M., Sohl-Dickstein, J., and Ganguli, S. (2016). Exponential expressivity in deep neural networks through transient chaos. In *Advances in neural information processing systems*, pages 3360–3368.

Power, A., Burda, Y., Edwards, H., Babuschkin, I., and Misra, V. (2022). Grokking: Generalization beyond overfitting on small algorithmic datasets. *arXiv preprint arXiv:2201.02177*.

Qin, S., Mudur, N., and Pehlevan, C. (2021). Contrastive similarity matching for supervised learning. *Neural computation*, 33(5):1300–1328.

Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018). Improving language understanding by generative pre-training. *OpenAI blog*.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Radhakrishnan, A., Stefanakis, G., Belkin, M., and Uhler, C. (2022). Simple, fast, and flexible framework for matrix completion with infinite width neural networks. *Proceedings of the National Academy of Sciences*, 119(16):e2115064119.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

Rotskoff, G. M. and Vanden-Eijnden, E. (2018). Neural networks as interacting particle systems: Asymptotic convexity of the loss landscape and universal scaling of the approximation error. *stat*, 1050:22.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533–536.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252.

Savarese, P., Evron, I., Soudry, D., and Srebro, N. (2019). How do infinite width bounded norm networks look in function space? In *Conference on Learning Theory*, pages 2667–2690. PMLR.

Saxe, A. M., Bansal, Y., Dapello, J., Advani, M., Kolchinsky, A., Tracey, B. D., and Cox, D. D. (2019). On the information bottleneck theory of deep learning. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12):124020.

Saxe, A. M., McClelland, J. L., and Ganguli, S. (2013). Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*.

Schmidhuber, J., Hochreiter, S., et al. (1997). Long short-term memory. *Neural Comput*, 9(8):1735–1780.

Shankar, V., Fang, A., Guo, W., Fridovich-Keil, S., Ragan-Kelley, J., Schmidt, L., and Recht, B. (2020). Neural kernels without tangents. In *International Conference on Machine Learning*, pages 8614–8623. PMLR.

Shwartz-Ziv, R. and Tishby, N. (2022). Opening the black box of deep neural networks via information. *Information Flow in Deep Neural Networks*, page 24.

Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations (ICLR 2015)*. Computational and Biological Learning Society.

Sirignano, J. and Spiliopoulos, K. (2020). Mean field analysis of neural networks: A law of large numbers. *SIAM Journal on Applied Mathematics*, 80(2):725–752.

Sirignano, J. and Spiliopoulos, K. (2022). Mean field analysis of deep neural networks. *Mathematics of Operations Research*, 47(1):120–152.

Song, Z. and Yang, X. (2019). Quadratic suffices for over-parametrization via matrix Chernoff bound. *arXiv preprint arXiv:1906.03593*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Tancik, M., Srinivasan, P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J., and Ng, R. (2020). Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in neural information processing systems*, 33:7537–7547.

Tishby, N., Pereira, F. C., and Bialek, W. (2000). The information bottleneck method. *arXiv preprint physics/0004057*.

Tishby, N. and Zaslavsky, N. (2015). Deep learning and the information bottleneck principle. In *2015 ieee information theory workshop (itw)*, pages 1–5. IEEE.

Tu, Z., Aranguri, S., and Jacot, A. (2024). Mixed dynamics in linear networks: Unifying the lazy and active regimes. *arXiv preprint arXiv:2405.17580*.

Vapnik, V. N. and Chervonenkis, A. Y. (1971). О равномерной сходимости частот появления событий к их вероятностям. *Теория вероятностей и ее применения*, 16(2):264–279.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*.

Vershynin, R. (2018). *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press.

Wang, T., Zhong, X., and Fan, Z. (2024). Universality of approximate message passing algorithms and tensor networks. *The Annals of Applied Probability*, 34(4):3943–3994.

Wigner, E. P. (1958). On the distribution of the roots of certain symmetric matrices. *Annals of Mathematics*, 67(2):325–327.

Xiao, L., Bahri, Y., Sohl-Dickstein, J., Schoenholz, S., and Pennington, J. (2018). Dynamical isometry and a mean field theory of CNNs: How to train 10,000-layer vanilla convolutional neural networks. In *International Conference on Machine Learning*, pages 5393–5402. PMLR.

Yang, G. (2019a). Scaling limits of wide neural networks with weight sharing: Gaussian process behavior, gradient independence, and Neural Tangent Kernel derivation. *arXiv preprint arXiv:1902.04760*.

Yang, G. (2019b). Wide feedforward or recurrent neural networks of any architecture are Gaussian processes. *Advances in Neural Information Processing Systems*, 32.

Yang, G. (2020a). Tensor Programs II: Neural Tangent Kernel for any architecture. *arXiv preprint arXiv:2006.14548*.

Yang, G. (2020b). Tensor Programs III: Neural matrix laws. *arXiv preprint arXiv:2009.10685*.

Yang, G., Hu, E., Babuschkin, I., Sidor, S., Liu, X., Farhi, D., Ryder, N., Pachocki, J., Chen, W., and Gao, J. (2021). Tuning large neural networks via zero-shot hyperparameter transfer. *Advances in Neural Information Processing Systems*, 34:17084–17097.

Yang, G. and Hu, E. J. (2021). Tensor Programs IV: Feature learning in infinite-width neural networks. In *International Conference on Machine Learning*, pages 11727–11737. PMLR.

Yang, G. and Littwin, E. (2021). Tensor Programs IIb: Architectural universality of Neural Tangent Kernel training dynamics. In *International Conference on Machine Learning*, pages 11762–11772. PMLR.

Yang, G. and Littwin, E. (2023). Tensor Programs IVb: Adaptive optimization in the infinite-width limit. *arXiv preprint arXiv:2308.01814*.

Yang, G., Santacroce, M., and Hu, E. J. (2022). Efficient computation of deep nonlinear infinite-width neural networks that learn features. In *International Conference on Learning Representations*.

Yarotsky, D. (2018). Collective evolution of weights in wide neural networks. *arXiv preprint arXiv:1810.03974*.

Yin, Y., Bai, Z., and Krishnaiah, P. (1984). On limit of the largest eigenvalue of the large dimensional sample covariance matrix. Technical report, Pittsburgh Univ PA Center for Multivariate Analysis.

Yu, X.-H. and Chen, G.-A. (1995). On the local minima free condition of backpropagation learning. *IEEE Transactions on Neural Networks*, 6(5):1300–1303.

Yue, K., Jin, R., Pilgrim, R., Wong, C.-W., Baron, D., and Dai, H. (2022). Neural Tangent Kernel empowered federated learning. In *International Conference on Machine Learning*, pages 25783–25803. PMLR.

Yun, C., Krishnan, S., and Mobahi, H. (2021). A unifying view on implicit bias in training linear neural networks. In *International Conference on Learning Representations*.

Yun, C., Sra, S., and Jadbabaie, A. (2019). Small ReLU networks are powerful memorizers: a tight analysis of memorization capacity. *Advances in Neural Information Processing Systems*, 32.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. (2017). Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*.

Zhou, W., Veitch, V., Austern, M., Adams, R. P., and Orbanz, P. (2019). Non-vacuous generalization bounds at the ImageNet scale: a PAC-bayesian compression approach. In *International Conference on Learning Representations*.

Evgenii (Eugene) Golikov

Education

2021 – 2025 **Ph.D. in Mathematics**, *École Polytechnique Fédérale de Lausanne (EPFL)*.
- **Advisor:** prof. Clément Hongler
- **Doctoral thesis:** "Deep Neural Networks: Large-Width Behavior and Generalization Bounds"
- **Defense date:** 27.01.2025

2017 – 2019 **M.Sc. in Applied Mathematics and Informatics**, *National Research University Higher School of Economics (HSE), dept. of Computer Science*.
- **Advisor:** prof. Dmitry Vetrov
- **Master thesis:** "Why does pre-training work with convolutional neural networks?"
- **Specialization:** Data Science, **GPA:** $8.5/10$ $(3.8/4)$

2017 – 2019 **Graduate**, *Yandex School of Data Analysis (SHAD)*.
- **Featured studies:** Algorithms and Data Structures, python, Machine Learning, Deep Learning, Reinforcement Learning, Bayesian ML

2010 – 2015 **Specialist (joint B.Sc./M.Sc. degree) in Mechanics**, *Lomonosov Moscow State University (MSU), dept. of Mechanics and Mathematics*.
- **Advisor:** prof. Vladislav Izmodenov
- **Qualification thesis:** "Modelling of thermal and supra-thermal populations in the region of the solar wind interaction with local interstellar medium"
- **Specialization:** Fluid Mechanics, *Diploma with Honours*, **GPA:** $4.92/5$

Work experience

2021 – 2025 **Doctoral assistant**, *Chair of Statistical Field Theory, École Polytechnique Fédérale de Lausanne*.

2017 – 2021 **Research scientist**, *DeepPavlov.ai, Neural Networks and Deep Learning lab., Moscow Institute of Physics and Technology*.

2015 – 2017 **Lab assistant**, *Institute of Space Research, Russian Academy of Sciences*.

Teaching

Fall 2021 **Deep Learning Theory**, *Moscow Institute of Physics and Technology*, Lecturer (remotely).
Web-page: `http://rairi.ru/wiki/index.php/Deep_learning_theory`

Fall 2020 **Deep Learning Theory**, *Yandex School of Data Analysis & Moscow Institute of Physics and Technology*, Lecturer.
- Videos: `https://yadi.sk/d/2wBuCT55B15SCw`
- Notes: `https://arxiv.org/abs/2012.05760`
- Chapter in Yandex.Handbook: `https://education.yandex.ru/handbook/ml/article/teoriya-glubokogo-obucheniya-vvedenie`

✉ golikov.e.a000@gmail.com
https://scholar.google.com/citations?user=EwGzknkAAAAJ

Spring & Fall 2019 **Deep Learning Theory**, *Moscow Institute of Physics and Technology*, Lecturer.

- Github:
  1. `https://github.com/deeppavlov/tdl`
  2. `https://github.com/deepmipt/tdl2`
- Videos:
  1. `https://www.youtube.com/playlist?list=PLt1IfGj6-_-dMa3Ff8mwjq1yOGijJ89Wa`
  2. `https://www.youtube.com/playlist?list=PLt1IfGj6-_-eiAGKvcZrHCp1mejmxMCiX`

Falls 2021–2024 **Analysis III**, *École Polytechnique Fédérale de Lausanne*, Assistant.

Selected publications

[1] Eugene Golikov. A Generalization Bound for Nearly-Linear Networks. *Transactions on Machine Learning Research (**TMLR**)*, 2025.

[2] Eugene Golikov and Greg Yang. Non-Gaussian Tensor Programs. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems (**NeurIPS, A***)*, 2022.

[3] Arthur Jacot, Eugene Golikov, Clément Hongler, and Franck Gabriel. Feature Learning in L_2-regularized DNNs: Attraction/Repulsion and Sparsity. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems (**NeurIPS, A***)*, 2022.

[4] Dilyara Baymurzina, Eugene Golikov, and Mikhail Burtsev. A review of neural architecture search. ***Neurocomputing (Q1)***, 474:82–93, 2022.

[5] Eugene Golikov, Eduard Pokonechnyy, and Vladimir Korviakov. Neural Tangent Kernel: A Survey. *arXiv preprint arXiv:2208.13614*, 2022.

[6] Eugene Golikov. Dynamically stable infinite-width limits of neural classifiers. *arXiv preprint arXiv:2006.06574*, 2020.

[7] Eugene Golikov. Towards a general theory of infinite-width limits of neural classifiers. In *International Conference on Machine Learning (**ICML, A***)*, pages 3617–3626. PMLR, 2020.

[8] EA Golikov, VV Izmodenov, DB Alexashov, and NA Belov. Two-jet astrosphere model: effect of azimuthal magnetic field. *Monthly Notices of the Royal Astronomical Society (**MNRAS, Q1**)*, 464(1):1065–1076, 2017.

Selected talks

27.11.2024 **Tensor Programs**, *EPFL*, Lausanne, Switzerland.
`https://drive.google.com/file/d/1kl1NpMcB2bJmLUOFbFPBMYWkmc6fNAnd/view?usp=drive_link`

25.10.2024 **Tensor Programs**, *Skoltech AI Center*, Moscow, Russia.
`https://drive.google.com/file/d/1cTn8DH45UFMvSVwi6YLuIYGa4Jv4WZB1/view?usp=sharing`

20.09.2024 **Recent Theoretical Results on Transformers**, *Huawei R-STW AI Innovation Summit*, St-Petersburg, Russia.

22.07.2023 **Tensor Programs**, *FLAIR tutorial (EPFL)*, Lausanne, Switzerland.

14.12.2022 **Non-Gaussian Tensor Programs**, *CIS NeurIPS regional event*, Lausanne, Switzerland.

6.12.2022 **Non-Gaussian Tensor Programs**, *NeurIPS'2022*, New Orleans, USA.
`https://nips.cc/virtual/2022/poster/54882`