Neural Networks: a Primer for Mathematicians

Boris Hanin
Princeton ORFE
bhanin@princeton.edu

March 28, 2026

Contents

1 Neural Networks: An Overview **5**
- 1.1 What is a NN? . . . 5
- 1.2 How are NNs used? . . . 7
- 1.3 Big Questions, Small Intuitions . . . 8
  - 1.3.1 Question: Why do we fit the data? . . . 9
  - 1.3.2 What do networks learn? . . . 12
  - 1.3.3 What are the scaling limits of neural networks? . . . 13

2 Neural Nets at Initialization **21**
- 2.1 Chapter Overview . . . 21
- 2.2 Random Neural Nets at Fixed Depth and Infinite Width . . . 22
  - 2.2.1 Sketch of Proof of Theorem 2.2.1 . . . 24
  - 2.2.2 Weight/bias variances and learning rates via Theorem 2.2.1 . . . 24
  - 2.2.3 Analyzing one step of GD to set the learning rate . . . 25
- 2.3 Finite Width Correction to the Gaussian Process Limit . . . 27
  - 2.3.1 Non-linearity Shaping at Large Depth . . . 29
  - 2.3.2 Open Problems . . . 30

3 Neural Nets in the Kernel Regime **33**
- 3.1 Chapter Overview . . . 33
- 3.2 Optimization and the Tangent Kernel . . . 34
  - 3.2.1 What is the NTK? . . . 34
  - 3.2.2 Optimization in the NTK Regime: heuristic derivation in shallow nets . . . 36
  - 3.2.3 Mean-Field vs. NTK and the μP Heuristic . . . 37

4 Mean Field Neural Nets **39**
- 4.1 Mean-Field One Hidden Layer Networks . . . 39
  - 4.1.1 Mean-Field Approximation . . . 40
  - 4.1.2 Mean-Field Optimization . . . 40
- 4.2 Neural Network Dynamical Mean-Field Theory . . . 42
  - 4.2.1 Sketch of DMFT Analysis in One Layer Networks . . . 43
  - 4.2.2 Open Problems . . . 45

5 Empirical Wonders **47**
- 5.1 Scaling Laws . . . 47
- 5.2 Grokking and Emergence . . . 48
- 5.3 Superposition . . . 49
- 5.4 Adversarial Examples . . . 50

Chapter 1

Neural Networks: An Overview

This chapter gives a self-contained introduction to neural networks. We begin by formally defining a particularly simple and fundamental class of networks, called fully connected networks in §1.1. While neural networks used in practice are typically more complex than these fully connected models, virtually all of them contain such fully connected networks as sub-parts. We will then briefly describe in §1.2 how neural networks are typically used in practice. Making sense of this procedure leads to many of the interesting mathematical questions in deep learning theory. We will discuss several of the most important in §1.3.

1.1 What is a NN?

A neural network is a family of functions $x \mapsto f(x;\theta)$ parameterized by a vector $\theta \in \mathbb{R}^{\#\text{parameters}}$. The simplest networks, and the focus of much of the exposition in these notes, are the so-called *fully connected* neural networks. They are defined as follows:

Definition 1.1.1 (Fully Connected Network)**.** *Fix a positive integer L as well as $L+2$ positive integers $N_0, \ldots, N_{L+1}$ and a function $\phi : \mathbb{R} \to \mathbb{R}$. A fully connected depth L neural network with input dimension N_0, output dimension N_{L+1}, hidden layer widths $N_1, \ldots, N_L$, and non-linearity ϕ is any function $x \in \mathbb{R}^{N_0} \mapsto z^{(L+1)}(x) \in \mathbb{R}^{N_{L+1}}$ of the following form*

$$z^{(\ell)}(x) = \begin{cases} W^{(1)}x + b^{(1)}, & \ell = 1 \\ W^{(\ell)}\phi(z^{(\ell-1)}(x)) + b^{(\ell)}, & \ell = 2, \ldots, L+1 \end{cases}, \tag{1.1.1}$$

where $W^{(\ell)} \in \mathbb{R}^{N_\ell \times N_{\ell-1}}$ are matrices, $b^{(\ell)} \in \mathbb{R}^{N_\ell}$ are vectors, and ϕ applied to a vector is shorthand for ϕ applied to each component. [1]

The parameters $L, N_0, \ldots, N_{L+1}$ are called the *network architecture*, and $z^{(\ell)}(x) \in \mathbb{R}^{N_\ell}$ is called the *vector of pre-activations at layer ℓ* corresponding to input x. A fully connected network with a given architecture and non-linearity is a family of functions parameterized by entries of the weight matrices $W^{(\ell)}$ and components of bias vectors $b^{(\ell)}$, which are refereed to as weights and biases, respectively.

Two Examples of Functions Computed by a Neural Network

In this section, we give two brief examples of functions computed by ReLU networks i.e. fully connected networks with the most popular non-linearity used in practice:

$$\sigma(t) = \text{ReLU}(t) := \max\{0, t\}.$$

[1] We will often find it convenient later in these notes to normalize layers but multiplying the weights $W^{(\ell+1)}$ by a factor like $1/\sqrt{N_\ell}$. This is simply a re-definition of the weights and biases.

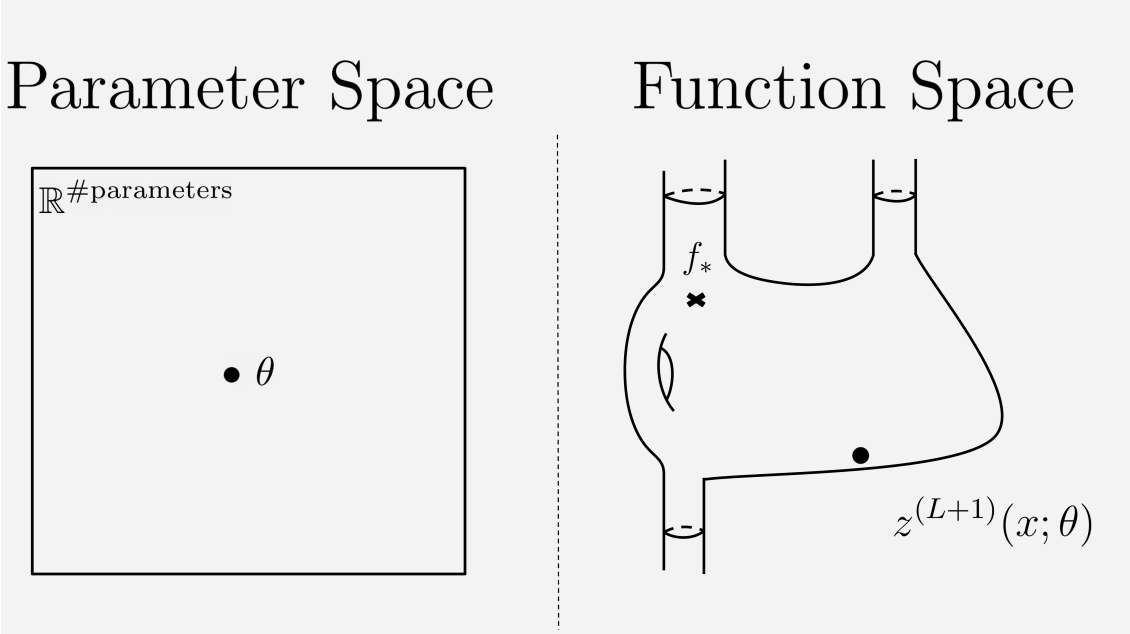

Figure 1.1: Two ways to visualize a neural network: either as a set of parameters to be optimized or as a set of functions to be searched over.

The first example is the following:

$$x \quad \mapsto \quad \sigma\left(\begin{pmatrix} 2 \\ -4 \end{pmatrix}^T \sigma\left(\begin{pmatrix} 1 \\ 1 \end{pmatrix} x + \begin{pmatrix} 0 \\ -1/2 \end{pmatrix}\right)\right) =: f(x).$$

A quick computation shows that this is the triangle or hat function:

$$f(x) = \begin{cases} 2x, & x \in [0, 1/2] \\ 1 - 2x, & x \in [1/2, 1] \\ 0, & x \in (-\infty, 0) \cup (1, \infty) \end{cases},$$

which plays a fundamental role in signal processing. The second example is more broad. Consider all one hidden layer ReLU networks with input and output dimensions equal to 1 (i.e. $L = 1, N_0 = N_2 = 1$):

$$z(x) = b^{(2)} + \sum_{i=1}^{N_1} W_i^{(2)} \sigma\left(W_i^{(1)} x + b_i^{(1)}\right), \qquad b_i^{(1)}, W_i^{(1)}, W_i^{(2)}, b^{(2)} \in \mathbb{R}. \tag{1.1.2}$$

The key point is the following

Lemma 1.1.2. *The space of functions obtained by varying $W_i^{(1)}, b_i^{(1)}, W_i^{(2)}, b^{(2)}$ in (1.1.2) contains all continuous piecewise linear functions with at most $N_1 - 1$ breakpoints (i.e. points of discontinuity for the derivative) and is contained in the space of all continuous piecewise linear functions with N_1 breakpoints.*

Proof Sketch. The idea is that the the i-th neuron $x \mapsto W_i^{(2)} \sigma\left(W_i^{(1)} x + b_i^{(1)}\right)$ is continuous and piecewise linear with a single breakpoint at

$$\xi_i := -b_i^{(1)} / W_i^{(1)}.$$

The remainder of the argument is left as an exercise. □

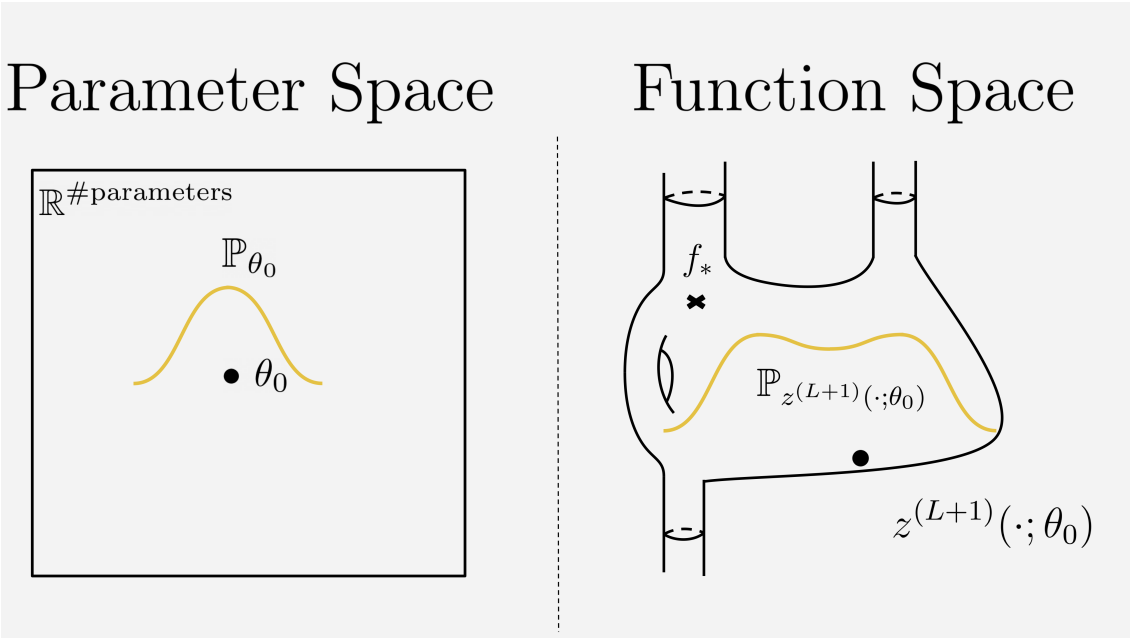

Figure 1.2: Simple probability measures in parameter space (e.g. those at initialization) correspond to complex distributions over functions.

1.2 How are NNs used?

Neural networks are used in practice to learn from data. This procedure can be roughly described as follows [2]

1. **Data Acquisition.** Collect a dataset $\mathcal{D} = \{(x_i, y_i))\}_{i=1}^n$ consisting of potentially noisy observations of an unknown function, e.g.

$$y_i = f_*(x_i) + \epsilon(x_i), \qquad \epsilon(x_i) \sim \mathcal{N}(0, \sigma_\epsilon^2). \tag{1.2.1}$$

   The goal is to find an approximation to the unknown function $f_* : \mathbb{R}^{N_0} \to \mathbb{R}^{N_{L+1}}$.

2. **Architecture Selection.** Choose an architecture, e.g. $L, N_1, \ldots, N_L, \phi$, which specifies the form of the computation $x \mapsto z^{(L+1)}(x;\theta)$ but does not determine the trainable parameters $\theta = \left\{W^{(\ell)}, b^{(\ell)}\right\}$. We seek to find θ_* so that $f_*(x) \approx z^{(L+1)}(x;\theta_*)$. See Figure 1.1.

3. **Initialization.** Select the weights and biases at random, e.g. as in (2.1.2). See Figure 1.2.

4. **Optimization.** Obtain a final setting of parameters θ_* using a first order optimization method such as gradient descent with a step size η

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta \mathcal{L}(\theta)\Big|_{\theta=\theta_t} \tag{1.2.2}$$

   to approximately minimize an empirical loss such as the mean squared error

$$\mathcal{L}(\theta) = \frac{1}{n}\sum_{i=1}^{n} \left|\left| y_i - z^{(L+1)}(x_i;\theta) \right|\right|^2. \tag{1.2.3}$$

   See Figure 1.3

[2] To be precise we are only really describing here the process of training neural networks from scratch, called pre-training. The modern deep learning pipeline also includes fine-tuning, other forms of post-training, and the use of large inference-time compute.

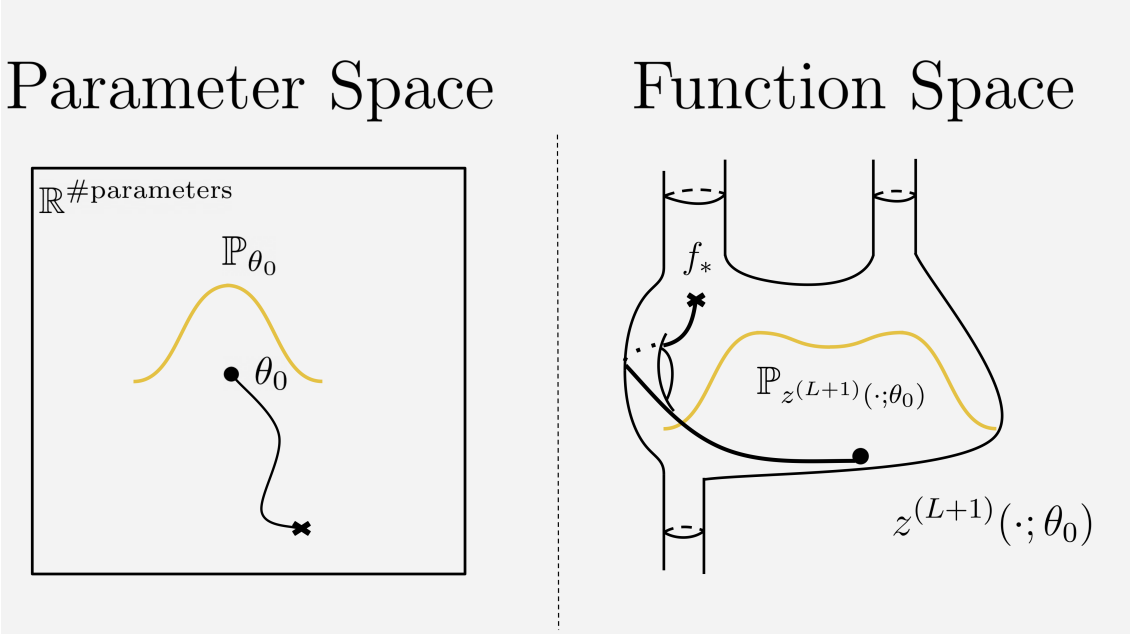

Figure 1.3: A cartoon illustration of neural network optimization dynamics in parameter space and in function space.

5. **Inference/Validation.** Check whether the model *generalizes* to unseen data by asking whether

$$f_*(x_{\text{new}}) \approx z^{(L+1)}(x_{\text{new}}; \theta_*).$$

On their face, steps 1 - 5 simply implement greedy search to minimize an *a priori* quite complicated function $\mathcal{L} : \mathbb{R}^{\#\text{parameters}} \to \mathbb{R}$. Steps $1 - 5$ in the previous section do not *always* yield a good approximation to the unknown function f_*. However, that this procedure is remotely successful in practice is remarkable and raises several fundamental questions that we will discuss in the next section.

1.3 Big Questions, Small Intuitions

In the previous section we outlined the core method by which neural networks are trained in practice. The fact that this method works is evidenced by the tremendous success of deep learning. It also raises several questions:

- **Q1. Why do we fit the data?.** The loss $\mathcal{L}_{\mathcal{D}}$ in (1.2.3) is non-convex as a function of the network parameters. Empirically, however, a greedy local search such as (stochastic) gradient descent (1.2.2) from a random initialization finds an (approximate) global minimum of $\mathcal{L}_{\mathcal{D}}$. Why and when is this possible? See §1.3.1.

- **Q2. What exactly are neural networks learning?** The output of a trained neural network can be viewed as a linear model applied to a learned, data-dependent set of basis functions, sometimes called features. It is precisely this kind of feature learning that distinguishes neural networks from more traditional forms of function approximation such as kernel methods. What kinds of features do neural networks actually learn? More generally, how do neural networks use structure in the training data to make predictions at test time? See §1.3.2.

- **Q3. What are the scaling limits of neural networks?** Neural networks are large in a variety of senses, e.g. they may have a large number of parameters, be trained on a large datasets, and be trained for many iterations. Can we relate neural network training across model, data, and compute scale? On the theory side: what are the possible scaling limits of neural networks when these fundamental quantities diverge and what can be said about them analytically? On the practical side: how can we take a large neural network and make it smaller while approximately preserving its training dynamics? See §1.3.3.

In the next section we discuss each question in more detail. The goal is to state more precisely important mathematical questions in deep learning theory and to introduce several important technical ideas in a less formal manner.

1.3.1 Question: Why do we fit the data?

As explained in §1.2 training a neural network $z(x;\theta)$ in practice almost always consists of minimizing an empirical loss

$$\mathcal{L}_{\mathcal{D}}(\theta) = \frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} \ell(y, z(x;\theta)) \tag{1.3.1}$$

using a first order method such a gradient descent

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta \mathcal{L}(\theta)\big|_{\theta=\theta_t}. \tag{1.3.2}$$

One should expect greedy local search such a gradient descent should converge to a stationary point (i.e. a point where $\nabla_\theta \mathcal{L}(\theta) = 0$) of the loss. With high probability or perhaps in the presence of noise, it therefore seems sensible to predict at large T that θ_T will be close to a *local minimum* of $\mathcal{L}$. However, in practice, what we find is that θ_T is close to a *global minimum.*

That this might be possible is not unprecedented. Spherical spin glasses, for example, are non-convex objective functions with a tremendous number of saddle points. However, while there are many distinct minima, it is known in certain models that with high probability almost all minima are approximately global minimizers [ABA13]. The situation with neural networks appears to be different, however. Rather than have a very large number of (approximate) global minima, neural network loss functions exhibit a phenomenon called *mode connectivity* [ESSN21, GIP$^+$18, DVSH18]. Namely, the set of global minima for $\mathcal{L}$ appears to consist of many large connected components. Moreover, all local minima for $\mathcal{L}$ appear to be exact, rather than merely approximate, global minima.

Explaining under what assumptions these statements can be made rigorous is an important open problem in modern machine learning, which is far from resolved. However, there is a simple intuition for at least one important mechanism driving the success of non-convex optimization. Namely, neural networks $z(x;\theta)$ in practice are almost always overparameterized in the sense that

$$\#\text{ parameters} \quad \gg \quad \#\text{ training datapoints.} \tag{1.3.3}$$

The key observation is that while $\mathcal{L}(\theta)$ is a complicated function of θ, it is a simple (e.g. convex) function in the variables

$$z(\mathcal{D};\theta) = (z(x;\theta), \quad (x,y)\in\mathcal{D}) \in \mathbb{R}^{\#\text{datapoints}}$$

that record the outputs on the training dataset. Moreover, precisely because of (1.3.3), a heuristic dimension-counting argument suggests

$$\theta \qquad \mapsto \qquad z(\mathcal{D};\theta) \text{ is a (surjective) submersion.} \tag{1.3.4}$$

Hence, by definition, we expect that the Jacobian

$$D_\theta z(\mathcal{D};\theta) \in \mathbb{R}^{\#\text{parameters}\times|\mathcal{D}|} \quad \text{has full rank for (almost) every } \theta.$$

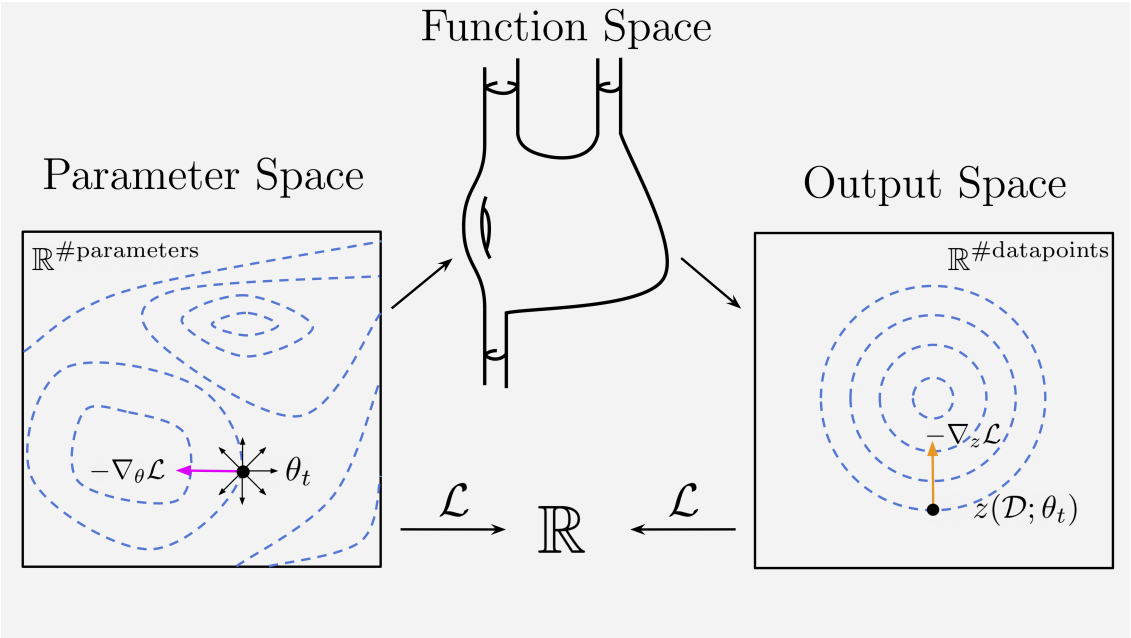

Figure 1.4: A commutative diagram when the loss depends on trainable parameters only through the values of network outputs on the training data. Blue dashed lines are level sets of the loss. Heuristically, in overparameterized models the map from parameter space to output space is a submersion. So the push-forward of the negative loss gradient $-\nabla_\theta \mathcal{L}$ in parameter space must have non-zero overlap with the loss gradient $-\nabla_z \mathcal{L}$ in output space.

Thus, since $\mathcal{L}(\theta)$ depends on θ only via $z(\mathcal{D};\theta)$, the chain rule yields

$$\nabla_\theta \mathcal{L}(\theta) = 0 \qquad \Longleftrightarrow \qquad D_\theta z(\mathcal{D};\theta) \;\cdot\; \nabla_z \mathcal{L}(z)\big|_{z=z(\mathcal{D};\theta)} = 0 \qquad \Longleftrightarrow \qquad \nabla_z \mathcal{L}(z)\big|_{z=z(\mathcal{D};\theta)} = 0.$$

If $\mathcal{L}$ is, say, a convex function of $z(\mathcal{D};\theta)$, we conclude that a stationary point of $\mathcal{L}$ with respect to θ necessarily corresponds to a global minimum of $\mathcal{L}$. A cartoon of this situation is illustrated in Figure 1.4.

The Jacobian $D_\theta z(\mathcal{D};\theta)$ play a key role in the theoretical analysis of neural networks where it usually enters through the neural tangent kernel (NTK), which is simply the kernel obtained by taking this Jacobian as its feature map:

$$\text{NTK}(x, x') := D_\theta z(x;\theta)^T D_\theta(x';\theta) \tag{1.3.5}$$

The subjectivity condition (1.3.4) is then equivalent to the statement that the $n \times n$ matrix obtained by evaluating the NTK on pairs of inputs from the training dataset is strictly positive definite.

The preceding discussion suggests that in an overparameterized neural network $z(x;\theta)$ we should expect the set of global minima of the empirical loss $\mathcal{L}$ (see (1.2.3)) to consist of infinitely many different values of θ. Among the many ways to fit the training data, it is natural to expect that some will correspond to rather wild/oscillatory interpolants, while others will not. In practice, vanilla gradient descent will typically find reasonable interpolants, otherwise it would not generalize to data outside the training set. A fundamental question is therefore: how, out of this sea of interpolants, the model and optimization algorithm conspire to select a particular minimum of the loss and why this interpolant is often well-behaved in practice?

A general language for explaining this so-called *implicit bias of optimization* remains elusive. However, there is a clear high-level intuition that might reasonably explain a part of why over-parameterized models trained using gradient descent are often well-behaved. The basic idea is

that in practice optimization is only possible with well-tuned architecture-dependent initialization schemes. The resulting distribution over functions $z(x;\theta_0)$ at the start of training (where θ_0 is chosen at random) specifies the initial conditions for optimization, and good initialization schemes used in practice are biased towards simple functions. Moreover, first order optimization methods are inherently local, corresponding to a kind of greedy search. Hence, at least intuitively, neural network optimization will seek out a way to fit the training data that is "as close as possible" to the simple functions represented by randomly initialized networks. This implicit bias due to initialization is one of motivations to studying random neural networks in the first place.

Perhaps the most canonical example of this cartoon of implicit bias is regression with overparameterized linear models. More precisely, we consider a linear model

$$z(x;\theta) = \sum_{j=1}^{P} \theta_j \phi_j(x), \tag{1.3.6}$$

where $\phi_j : \mathbb{R}^{N_0} \to \mathbb{R}$ are a given collection of features, and a collection of training datapoints

$$\mathcal{D} = \{(x_i, y_i),\, i = 1, \ldots, n\} \subseteq \mathbb{R}^{N_0+1}.$$

We will assume that the model is overparameterized, which in this case means

$$P = \#\text{ parameters } > \#\text{ datapoints } = n.$$

Consider an empirical loss

$$\mathcal{L}_{\mathcal{D}}(\theta) = \sum_{i=1}^{n} \ell\left(z(x_i;\theta), y_i\right),$$

where for each y the function $z \mapsto \ell(z,y)$ is strictly convex and achieves a minimum only at on the diagonal $z = y$ (e.g. $\ell(z,y) = (z-y)^2$). Minimizing $\mathcal{L}_{\mathcal{D}}$ is equivalent to solving an underdetermined systems of linear equations with n equations and P unknowns. The set of solutions to such a system of equations is a hyperplane that is parallel to the kernel of the $n \times P$ feature matrix Φ defined by

$$\Phi := \left(\phi_1(\mathcal{D}), \ldots, \phi_P(\mathcal{D})\right), \qquad \phi_j(\mathcal{D}) = \left(\phi_j(x_1) \cdots \phi_j(x_n)\right)^T.$$

This hyperplane is the analog in linear models of the infinitely many minima one generically expects for an empirical loss in an overparameterized neural network. Since the loss $\mathcal{L}$ is convex, any reasonable first order method such as gradient descent with a sufficiently small step size will converge to a global minimum.

The question of implicit bias is: what determines which particular minimizer one will obtain? In this case, the answer comes by noting that the gradient of the loss

$$\nabla_\theta \mathcal{L}_{\mathcal{D}}(\theta) = \sum_{i=1}^{n} \partial_z \ell(z,y)\big|_{z=z(x_i;\theta), y=y_i} \cdot \Phi(x_i) \in \mathrm{col}(\Phi^T) \tag{1.3.7}$$

always belongs to the column space of Φ^T. The key point is that now that $\mathrm{col}(\Phi^T)$ is perpendicular to the kernel of Φ and hence we conclude that gradient descent starting from θ_0 converges to the orthogonal projection of θ_0 onto the space of minima of $\mathcal{L}$. Put another way, gradient descent starting at θ_0 converges to the nearest, in the ℓ_2 sense, minimum of $\mathcal{L}$. This observation can be further understood by considering the decomposition

$$\theta = \theta_{||} + \theta_{\perp}, \qquad \theta_{\perp} \in \ker(\Phi), \quad \theta_{||} \in \mathrm{col}(\Phi^T). \tag{1.3.8}$$

Since $\nabla_\theta \mathcal{L}_{\mathcal{D}}(\theta) \in \mathrm{col}(\Phi^T)$, we see that $\theta_\perp$ does not change in the course of gradient descent. In contrast, $\mathcal{L}_{\mathcal{D}}$ is strictly convex on $\mathrm{col}(\Phi^T)$ and hence $\theta_{||}$ converges to the unique minimum of $\mathcal{L}_{\mathcal{D}}$ inside $\mathrm{col}(\Phi^T)$. This unique minimum is precisely the minimal ℓ_2-norm minimum of $\mathcal{L}$. It is natural to expect that the ℓ_2 will appear since we compute gradients using the ℓ_2 inner product.

An interesting consequence of the preceding discussion is that gradient descent starting at $\theta_0 = 0$ converges to the minimal ℓ_2-norm minimizer of $\mathcal{L}$. This helps to explain why implicit bias is sometimes referred to as implicit regularization: while we did not seem to explicitly penalize the ℓ_2 norm of the parameter vector θ when minimizing $\mathcal{L}_{\mathcal{D}}$, our choice of optimization algorithm implicitly does this for us.

Of course when $z(x;\theta)$ is non-linear in θ the preceding discussion – in particular the crucial estimate (1.3.7) – no longer holds. However, the general idea that starting near with a simple (in this case small ℓ_2-norm) guess for θ and performing a greedy local search will return a minimum of $\mathcal{L}_{\mathcal{D}}$ that is in some sense as close as possible to the simple predictors we started with. This general idea can have somewhat surprising manifestations.

Bibliography

The literature on the benign nature of the optimization landscape in deep learning and on the NTK in particular is rich. The early article [BH89], for instance, showed that for a natural class of one hidden layer linear networks with loss landscape of mean-squared error has many critical points but no minima aside from the global minima. Much later this was refined in [Kaw16, LK17] to deeper linear networks and more general training data. The structure of the loss landscape for one layer networks has also been related to low-rank tensor recovery [MM18] and minimization by gradient flow in Wasserstein space [CB18, RVE18, MMN18, SS21, SS20]. Finally, the simplicity of optimization for wide networks in the so-called kernel regime was first remarked in [DLT$^+$18] and introduced more generally in original NTK paper [JGH18]. This led to a number of important followups, notably [ACGH19, DZPS19, LZB22].

Next, the implicit bias of optimization in overparameterized models has also been studied in a range of settings. In the setting of classification with overparameterized linear models trained with exponential-type losses [SHN$^+$18] obtain a fundamental result about the implicit bias of gradient descent on separable data. Next, the influential article [WGL$^+$20] considered the implicit bias of quadratic re-parameterization of linear models. Effect of ℓ_2 penalty on network weights in one layer ReLU networks was considered in [SESS19, OWSS19, Han21]. Next, a line work work (e.g. [ACH18, SMG14, JGŞ$^+$21, Jac22]) concerns the implicit bias of optimization in deep linear networks. Finally, the Implicit bias of gradient descent in homogeneous networks was considered in [JT19, JT20].

1.3.2 What do networks learn?

Perhaps the most mathematically well-understood form of learning is approximation by linear spaces of functions. Such linear models take the form

$$z(x;\theta) = \sum_{j\geq 1} \theta_j \phi_j(x),$$

where $\Phi(x) = (\phi_j(x),\, j \geq 1)$ is a given set of functions. The choice of a "good" basis – one in which the unknown function f_* underlying the training data has an efficient expansion – is key to providing theoretical guarantees for how well one can estimate the vector of coefficients θ based on observations from f. In contrast, the components of the output of a neural network can be written as

$$z_i^{(L+1)}(x;\theta) = \sum_{j=1}^{N_L} W_{ij}^{(L+1)} \phi_j(x;\theta), \qquad \phi_j(x;\theta) = \phi\left(z_j^{(L)}(x;\theta)\right).$$

Hence, neural network training consists of learning both the basis and the coefficients in the linear combination simultaneously. It is widely believed that the ability to learn data-adaptive basis functions (a.k.a. features) is a key component of the success of neural networks. From this point of view, a core question in the theory of deep learning is to describe how learned features depend on the data generating process, network architecture, and optimization protocol. In the last few

years there has been a flurry of beautiful results characterizing, primarily for one hidden layer networks, how learned features reflect latent low-dimensional structure in the training data.

While in the preceding discussion the final layer components $\phi_j(x;\theta)$ play the role of features, this this is not the only interesting way to study feature learning. Indeed, it is also natural to consider the properties of the feature map

$$x \in \mathbb{R}^{N_0} \quad \mapsto \quad \nabla_\theta z^{(L+1)}(x;\theta) \in \mathbb{R}^{\#\text{parameters}}. \tag{1.3.9}$$

This is precisely the Jacobian used the define the NTK in (3.2.3) and it's role as a feature maps arises in two ways:

- **Residual feature learning.** First, one can think about gradient descent (1.3.8) with a small learning rate as a two step process. First, given the current setting θ_t of trainable parameters, we repldace $z^{(L+1)}(x;\theta)$ by its linearization around θ_t

$$z^{(L+1);\text{lin}}(x;\theta;\theta_t) = z^{(L+1)}(x;\theta_t) + \left\langle \nabla_\theta z^{(L+1)}(x;\theta_t), \theta - \theta_t \right\rangle.$$

  Second one step of gradient descent on the empirical risk $\mathcal{L}$ *using this linearized model.* To first order in the learning rate, this is equivalent to gradient descent. The key point is therefore that in each step of GD, we use the feature map $x \mapsto \nabla_\theta z^{(L+1)}(x;\theta)$ to construct an update for the residual $z^{(L+1);\text{lin}}(x;\theta) - y$ at two consecutive time steps of gradient descent.

- **Fine-tuning as linearization.** For a slightly different point of view on why the feature map (1.3.9) is natural, recall the common ML pipeline of pre-training and fine-tuning. This phrase means that we train an big model on an upstream task and then fine-tune = train a little on a downstream task. This second phase of training a bit can sometimes be modeled by training the linearized model. Hence, the Jacobian features are those used for fine-tuning on down-stream tasks.

Understanding feature learning for either the final layer features of the network Jacobian is largely open. The main avenue of approach has been to study them near initialization, where the analysis basically hinges on understand the statistics of networks at initialization:

Bibliography

There is a vast literature on feature learning in one layer networks. Foundational articles on learning single and multi-index models with gradient descent on shallow networks are [BBSS22, BBPV23, DLS22, MHPG$^+$22, MHWSE23]. There are also several influential analyses of features learned from one step of feature learning [BES$^+$22, BES$^+$23, DKL$^+$24, CPD$^+$24]. Further, several works concern the order of learning sparse boolean functions or low degree polynomials with SGD [AAM22, AAM23] with an important followup clarifying the crucial role of online SGD in the two preceding works [DTA$^+$24]. Next, building on the prescient work of Saad and Solla [SS95] is this important analysis [ASKL23] that provides a unified approach to studying the limits of growing input dimension, growing hidden layer width, and vanishing step size.

Beyond the setting of learning features in one layer networks, there are several influential early empirical works [YCBL14, YCN$^+$15] on features learned by early modern vision architectures. There is also an influential line of work on superposition in feature learning starting with [EHO$^+$22].

Finally, there are several interesting strands of work on feature learning in Bayesian inference with deep neural networks. This includes work on kernel renormalization [LS21, BGV$^+$25], analyses of hidden layer features in deep models [APP$^+$22, SNR23, NDSR21, NR21, FLD$^+$24, RRSH24, RFL$^+$25], and works specifically about the predictive posterior [ZVTP22, HZ23, HZ26].

1.3.3 What are the scaling limits of neural networks?

A hallmark of modern deep learning is *scale*, and state-of-the-art neural networks are often large in at least these senses:

- **Parameter count.** Neural networks often have from hundreds of millions to hundreds of billions trainable parameters.
- **Dataset size.** Neural networks are often trained to fit datasets with anywhere from tens of thousands to trillions of examples, each of which might itself be high-dimensional (e.g. an image or video).
- **Optimization steps.** Neural networks can often by trained for hundreds of thousands or even millions of optimization steps.

The large scale of neural networks used in practice suggests that learning with these models may be well-approximated by their scaling limits, i.e. idealized infinite sized models obtained by sending at some or all of parameter count, dataset size, and optimization steps to infinity. Mathematics and theoretical physics have long list of tools, including random matrix theory, mean field theory, ergodic theory, that enable an asymptotic analysis of large interacting systems, and these tools can all profitably be used to analyze deep learning. Before getting into the details of *how* these tools are used for studying deep learning we give both a theoretical and practical motivation for studying neural network scaling limits.

Neural Network Scaling Limits: Theoretical Motivation

From the theoretical point of view, infinite size scaling limits are often tractable and provide good approximations to large finite systems. Examples of this include the emergence of the semi-circle law for the empirical distribution of eigenvalues of random $n \times n$ Hermitian matrices in the regime when $n \to \infty$ and deterministic PDEs in the space of probability measures describing the evolution of the empirical measure of n particles with mean-field interactions.

A core goal of studying neural network scaling limits from this point of view is therefore to derive simple descriptions of learning that capture essential properties of real neural networks. There are two main difficulties. There are many different possible scaling limits in part because the limits as network depth, width, training data set size, and number of optimization steps grow do not commute! To illustrate this point let us briefly consider two relatively simple but illustrative examples: non-commutativity of growing width and number of samples for learning with linear models and non-commutativity of growing width and depth in randomly initialized deep linear networks.

Example: Non-commutativity of width and number of samples in linear models. Nothing in this section is new. It is meant to be a simple illustration of the well-known fact that care must be taken in taking scaling limits even in the classical setting of fitting a dataset

$$\mathcal{D} = \{(x_i, y_i)\}_{i=1,\dots,n}, \qquad x_i \in \mathbb{R}^N, \quad y_i \in \mathbb{R}$$

with independent isotropic Gaussian inputs and noisy linear labels:

$$y_i = \beta^T x_i + \epsilon_i, \qquad x_i \sim \mathcal{N}(0, I), \quad \epsilon_i \sim \mathcal{N}(0, \phi_\epsilon^2), \quad ||\beta|| = 1$$

using a *linear* model

$$z(x; W) = W^T x,$$

which can be thought of as a fully connected network with $L = 0$ hidden layers. A standard way to estimate the unknown vector β is by ridge regularized least squares:

$$W_*(\lambda) = \text{argmin}_{W \in \mathbb{R}^d} \mathcal{L}_\lambda(W), \qquad \mathcal{L}_\lambda(W) = \frac{1}{2n} \sum_{i=1}^{n} \left(z(x_i; W) - y_i\right)^2 + \frac{\lambda}{2} ||W||^2, \quad \lambda > 0.$$

This model of learning has two large parameters: the number of samples n and the number of features N, typically referred to as the number of features. Suppose now we seek to study this model when both n and N may be large, say with $n = 10^6$ and $N = 3 \cdot 10^6$, via a scaling limit in

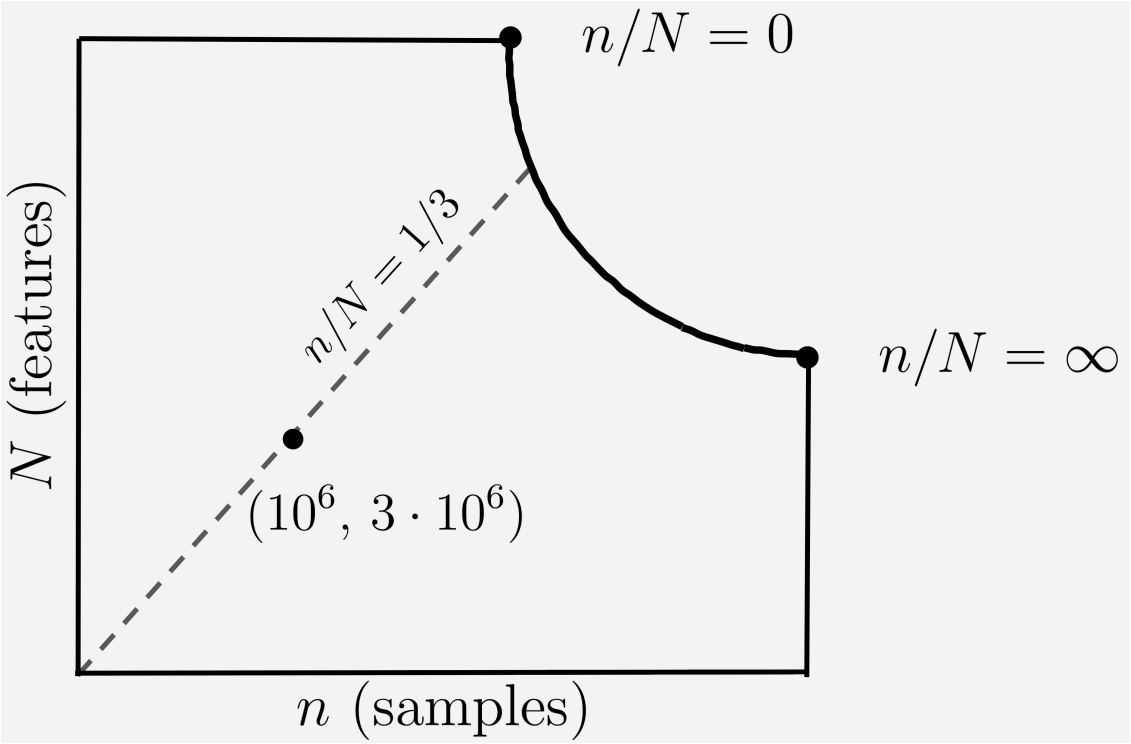

Figure 1.5: A cartoon illustration of blowing up the "moduli space" for learning with regularized linear models in N dimension with n isotropic covariates.

which we take at least one of n, N to infinity. Our goal is a method of taking $n, N \to \infty$ so that the quality of learning at infinite size, for instance as measured by the test error

$$\mathbb{E}\left[\frac{1}{2}\left(W_*^T x - y\right)^2\right], \qquad x \sim \mathcal{N}(0, I), \quad y = \beta^T x + \epsilon,$$

is a good approximation to learning when n, N are both large and finite. To proceed, note that in the simple setting we are considering there is an exact formula

$$W_*(\lambda) = (XX^T + \lambda I)^{-1} X^T Y, \qquad X^T = [x_1 \cdots x_n] \in \mathbb{R}^{N \times n}, \qquad Y = [y_1 \cdots y_n]^T \in \mathbb{R}^n \quad (1.3.10)$$

obtained by simply minimizing the quadratic objective $\mathcal{L}_\lambda$. This formula reveals an important lesson: the distribution of $W_*(\lambda)$ depends crucially on the statistics of the Wishart random matrix XX^T. The key is now to recall the Marchenko-Pastur law [AGZ10], which shows that at large n, N the eigenvalue distribution of XX^T depends sensitively on the ratio n/N.

The appearance of the ratio n/N, which can be thought of as measuring *the effective number of training samples*, underscores the fundamental fact that the limits and $n, N \to \infty$ don't commute. That is, if we were to study a scaling limit in which we first took $n \to \infty$ and then $N \to \infty$ or vise versa the resulting model, which is still well-defined, would not bear any resemblance to the finite but large model we started with. Instead, the correct way to compute scaling limits was to fix the ratio $n/N = 1/3$ and then take n, N to infinity together.

A cartoon illustration is in Figure 1.5. The situation is roughly that we have a *moduli space* of problems parameterized by $n, N \in \mathbb{Z}_+ \times \mathbb{Z}_+$. We seek to compactify this moduli space in such a way that the objects of interest – in this case the statistics of $W_*(\lambda)$ – are continuous on the boundary. One can then think of Figure 1.5 as blowing up the corder $n = N = \infty$ in the usual sense of replacing it by the inward pointing normal bundle, which parameterized by the ratio n/N.

The situation of realistic neural networks is much harder. Indeed, there is no longer simple formula akin to (1.3.10) and it is an important problem to analyze the dynamic of training.

Moreover, as discussed in §1.3.1, the empirical loss over a neural networks is non-convex and typically has infinitely many minima. Which minimum is found depends on the details of the initialization and the optimization procedure, meaning that one cannot hope to studying scaling limit of neural networks without also studying scaling limits of optimizers. Finally, as we will see in the next section, neural networks size is not captured by a single parameter such as number of features d. Instead, even for the simple case of fully connected networks, both network depth and width play a role.

Example: Non-commutativity of depth and width in deep linear networks at initialization

The purpose of this section is to introduce the non-commutativity of depth and width and in fully connected networks through the special case of so-called *deep linear networks*.

Definition 1.3.1 (Deep Linear Network)**.** *Fix a positive integer L as well as $L+2$ positive integers $N_0, \dots, N_{L+1}$. A depth L linear neural network with input dimension N_0, output dimension N_{L+1}, hidden layer widths $N_1, \dots, N_L$ is a fully connected network*

$$z^{(L+1)}(x;\theta) = W^{(L+1)} \cdots W^{(1)} x, \quad W^{(\ell)} \in \mathbb{R}^{n_\ell \times N_{\ell-1}}$$

with this architecture, identity non-linearity $\phi(t) = t$, and zero biases.

To keep the notation a simple as possible we will set $N_0 = \cdots = N_{L+1} = n$. In accordance with Definition 2.1.1 a random deep linear networks is obtained by choosing $W_{ij}^{(\ell)}$ at random:

$$W_{ij}^{(\ell)} \sim \mathcal{N}\left(0, \left(\sigma_W^{(\ell)}\right)^2\right),$$

where $(\sigma_W^{(\ell)})^2$ are constants that we are free to set. In short, randomly initialized linear networks are products of independent matrices with iid centered Gaussian entries. This model has been studied extensively in two regimes:

- **Free Probability Regime.** In this regime, L is fixed and $n \gg 1$. This is a kind of maximum entropy regime in which the global distribution of singular values can be characterized in terms of maximizing the non-commutative entropy (cf eg [AG97, BBCC11]).
- **Ergodic Regime.** In this regime n is fixed and $L \gg 1$. This is a kind of minimal entropy regime in which the Lyapunov exponents (and singular values of $X_{N,n}$) tend to almost sure limits. See [FK60, BLR85, New86, IN92, CPV12].

The free and ergodic regimes are studied by rather different tools. In the free probability regime, for example, one typically begins by studying the global distribution of singular values (or eigenvalues). Only then does one proceed to progressively resolve the statistics of individual singular values, starting with singular values in the bulk and finally moving to the edge of the spectrum. The usual approach to studying the ergodic regime is the opposite: the largest singular value (or equivalently) Lyapunov exponent is the simplest to analyze. One then proceeds to studying progressively smaller singular values until finally a characterization of the entire empirical distribution of the spectrum might be possible.

This significant difference between the free and ergodic regimes is reflected in the basic mathematical fact that the limits of large L and large n do not commute. Instead, at large n, L there should be a one-parameter family of matrix models indexed by the

$$\textit{effective network depth} \;=\; \frac{L}{N}$$

interpolates between the free and ergodic regimes. Ample evidence for this statement in the context of linear statistics for matrix products with rather general matrix entries was obtained

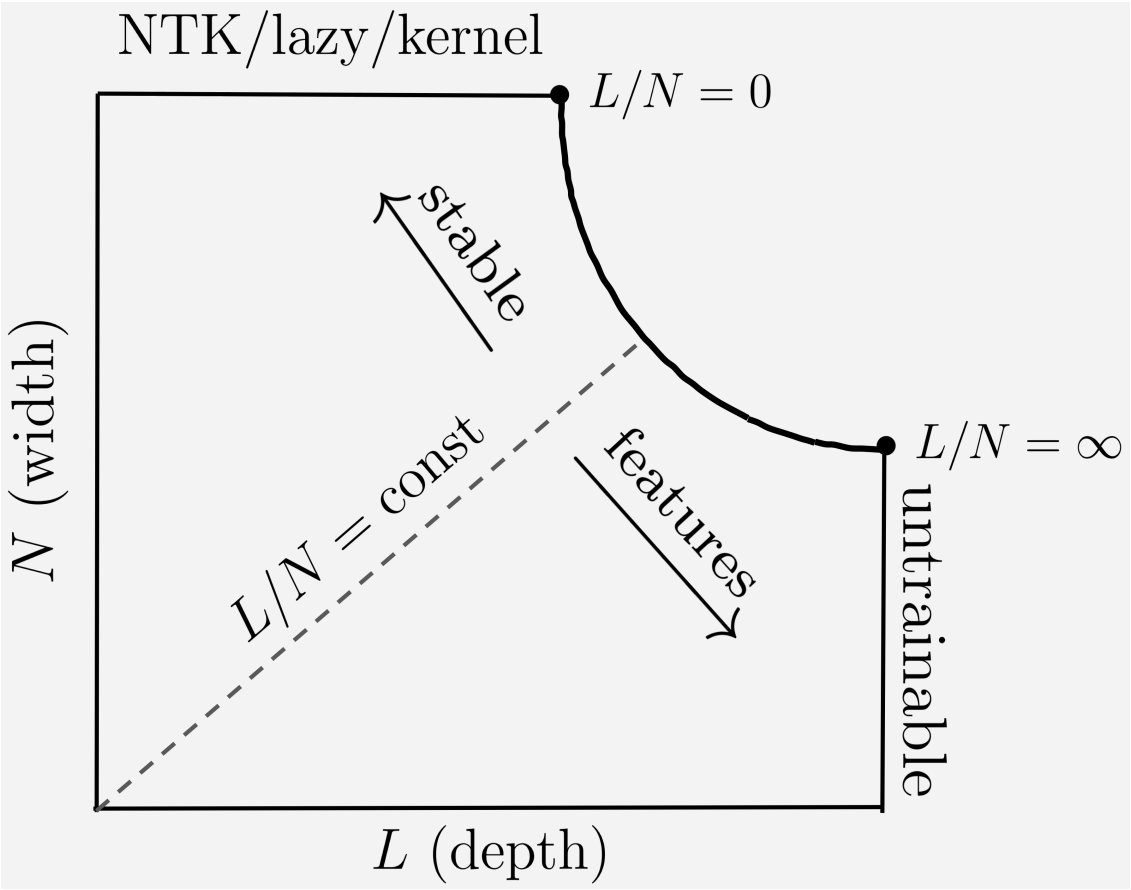

Figure 1.6: Many important properties of fully connected neural networks with NTK parameterization (see Chapter 3) are summarized at large width N and depth L by their ratio L/N, which plays the role an an effective depth. See §2.3 and [Han24, RYH22, BCCZ23].

[HJ25, HN20b, Han18], for matrix products with complex Gaussian entries in [Ahn22, GS18] and, at a physics level of rigor, for the full joint distribution of singular values [AB12, ABK20, LWW18]. Proving this in general matrix products remains an interesting open problem.

As we shall see in §2.3, the depth-to-width ratio L/N also plays an important role in determining the properties of random networks with general non-linearities. From this point of view, we will see that to understand deep and wide networks it is crucial to consider finite width corrections on the order of $1/N^k$ for various $k \geq 0$. The $L-$dependence of such corrections actually causes them to scale like $(L/N)^k$, making them leading order effects at large N, L (see [Han24, RYH22]). The rough picture, many parts of which are only partly proved, is that larger L/N corresponds to more non-linear but less stable models (see Figure 1.6)

We illustrate this point by a simple computation showing why it is L/N, rather than some other combination of L and N, that should determine the behavior of random deep linear networks at large N, L. For this, let's study the squared norm

$$X_{N,L+1} := \left|\left|z^{(L+1)}(x;\theta)\right|\right|^2, \qquad ||x|| = 1$$

of the output of a random deep linear network. In order to understand its distribution let's write

$$X_{N,L+1} = \left|\left|W^{(L+1)} \cdots W^{(1)} x\right|\right|^2 = \left|\left|W^{(L+1)} \cdots W^{(2)} \frac{W^{(1)}x}{||W^{(1)}x||}\right|\right|^2 \left|\left|W^{(1)}x\right|\right|^2 .$$

Note that

$$\left|\left|W^{(1)}x\right|\right|, \quad \frac{W^{(1)}x}{\left|\left|W^{(1)}x\right|\right|} \qquad \text{are independent}$$

with

$$\left|\left|W^{(1)}x\right|\right|^2 \overset{d}{=} (\sigma_W^{(1)})^2\chi_N^2, \qquad \frac{W^{(1)}x}{\left|\left|W^{(1)}x\right|\right|} \sim \text{Uniform}\left(S^{n-1}\right)$$

Thus, in fact the presentation above allows us to write $X_{N,L+1}$ as a product of two independent terms! Repeating this argument, we obtain the following equality in distribution:

$$X_{N,L+1} \overset{d}{=} \prod_{\ell=1}^{L+1} N\left(\sigma_W^{(\ell)}\right)^2 Y_\ell, \qquad Y_\ell \sim \frac{1}{N}\chi_N^2 \quad \text{independent.}$$

Taking the mean of both sides shows that

$$\mathbb{E}\left[X_{N,L+1}\right] = \prod_{\ell=1}^{L+1} N\left(\sigma_W^{(\ell)}\right)^2$$

This means that for deep linear networks to be well-behaved at large N, L requires $(\sigma_W^{(\ell)})^2 = 1/N$. This is an example of what is called *tuning to criticality*, which was thoroughly studied for general non-linearities in [PLR+16, PSG18, SGGSD17] and especially in [RYH22]. With this choice of weight variance, the central limit theorem implies

$$X_{N,L+1} \overset{d}{=} \exp\left[\sum_{\ell=1}^{L} \log\left(Y_\ell\right)\right] \overset{L \gg 1}{\approx} \exp\left[\mathcal{N}\left(-\frac{L}{2N}, \frac{L}{N}\right)\right].$$

Taking N large in each layer tries to make each $\log(Y_\ell)$ close to 0, with mean and variance both on the order of $1/N$. Adding together L such terms produces correction that is exponential in the effective network depth L/N.

Neural Network Scaling Limits: Practical Motivation

The hallmark of modern deep learning is scale: massive models trained on sprawling datasets over many iterations. Realizing the practical utility of scale requires a practitioner to make many choices in concert, e.g.

- **Architecture Selection.** Which architectures (e.g. fully connected, residual, convolutional, transformer-based, etc) are best for a given problem? Given an architecture type, say a transformer, how should one set its *shape*, i.e. scale models through growing depth, width, number of attention heads, number of experts, and so on?
- **Hyperparameter Tuning.** Given an architecture to be trained, how should one choose initialization variances, learning rates, batch sizes, weight decay strengths, and other *hyperparameters* needed to specify an optimization scheme?

If architecture and optimizer choices are not properly made, then large models perform poorly. At the same time, large models are so expensive that it is computationally impossible to find performance architecture and hyperparameters settings by trail and error.

Thus, while scaling limit theory seeks simplicity through increased scale, the practical application of scaling limits is to produce faithful small versions of large models. These scaled down network become cheap testing grounds for choices that matter at scale: training settings, architecture selection, and deployment strategies.

The idea of using scaling limits for hyperparameter transfer has had significant practical impact [YHB+21, DZN+25]. More broadly, the idea of relating the outcomes of experiment with small scale neural networks to properties of large models is one of the core problems faced by deep learning practitioners.

Bibliography

The literature on scaling limits of neural networks is expansive. Analyses of sgd and Bayesian inference in the infinite width limit of one layer neural networks has a long history going back at least to [SS95, Nea96]. The more modern work on the NTK-type infinite width limits goes back to [DLT+18, JGH18] and includes many influential follows including [ACGH19, LZB22, DZPS19, AZLL19].

The study of mean-field limits of one layer networks was initiated in [MMN18, RVE18, CB18] with important follows up [SS20, SS21, BMZ23] and many of the references from §1.3.2.

The limit of simultaneously large depth and width in fully connected networks was initiated via a combinatorial path counting approach in [Han18, HN20b]. Important followups include the neural covariance SDE work [LNR22] the kernel shaping article [MBD+21], which introduced the idea of shaped non-linearities, and the effective field theory approach of [RYH22] to studying deep and wide networks both at init and after training. Additionally, there are also the works [HZ23, HZ26] on Bayesian inference with deep and wide networks (see also the references on Bayesian inference with wide neural networks from §1.3.2).

Prior theoretical work on hyperparameter tuning goes back at least to [LBOM02, GB10, Nea96], which studied how to scale initialization with network width. More modern work, starting from [PLR+16, SGGSD17], focused on tuning order 1 constants in the initialization variance for stability of the forward pass across depth. This was taken to its logical conclusion for fully connected networks in the book [Rob21] and for transformers in the paper [DYZ23]. The preceding methods for hyperparameter turning were of NTK-type. In contrast, mean-field hyperparameter tuning/transfer was initiated in [YH21] and [YHB+21], which concerned width scaling. Followup work [YYZH23, BNL+23] concerned hyperparameter transfer for simultaneously growing depth and width in residual networks, while [BCP24] and the more empirical followup [DZN+25] concerned hyperparameter transfer in transformers. Finally, the recent article [JBPH26] studies hyperparameter transfer for mixture of expert models.

Chapter 2

Neural Nets at Initialization

2.1 Chapter Overview

In this chapter we will study neural networks at the start of training, when they have random weights and biases. To describe the precise setup recall from Definition 1.1.1 that given an input $x \in \mathbb{R}^{N_0}$ a fully connected network produces an output $z^{(L+1)}(x) \in \mathbb{R}^{N_{L+1}}$ through a series of hidden layer pre-activations $z^{(\ell)}(x) \in \mathbb{R}^{N_\ell}$, whose components are defined recursively as follows:

$$z_i^{(\ell+1)}(x) = \begin{cases} b_i^{(\ell+1)} + \sum_{j=1}^{N_\ell} W_{ij}^{(\ell+1)} \phi\left(z_j^{(\ell)}(x)\right), & \ell = 1, \dots, L+1 \\ b_i^{(1)} + \sum_{j=1}^{N_0} W_{ij}^{(1)} x_j, & \ell = 0 \end{cases} . \tag{2.1.1}$$

As explained in §1.2, neural network training begins by selecting the weights $W_{ij}^{(\ell)}$ and biases $b_i^{(\ell)}$ at random:

Definition 2.1.1 (Random Fully Connected Network)**.** *Fix $L, N_0, \dots, N_{L+1} \geq 1$ and $\phi : \mathbb{R} \to \mathbb{R}$. A random depth L neural network with input dimension N_0, hidden layer widths $N_1, \dots, N_L$, output dimension N_{L+1}, and non-linearity ϕ is obtained by taking random weights and biases in* (2.1.1)*:*

$$W_{ij}^{(\ell+1)} \sim \mathcal{N}\left(0, \left(\sigma_W^{(\ell)}\right)^2\right), \quad b_i^{(\ell+1)} \sim \mathcal{N}(0, \sigma_b^{(\ell)}) \qquad \textit{independent.} \tag{2.1.2}$$

It is not really important in Definition 2.1.1 that the weights and biases Gaussian, only that they are centered, independent, and have light tails. Moreover, as we'll explain below, it will be natural to choose

$$\left(\sigma_W^{(\ell)}\right)^2 = \frac{\sigma_W^2}{N_{\ell-1}}, \qquad \sigma_b^{(\ell)} = \sigma_b,$$

where σ_W^2, σ_b are order 1 constants independent of the network width.[1] It may seem at first glance that studying neural networks with random weights and biases is of no practical interest. After all, a neural network is only useful after it has been trained, i.e. one has found a setting of its parameters so that the resulting network function (at least approximately) interpolates a given training dataset of input-output pairs $(x, f(x))$ for an otherwise unknown function $f : \mathbb{R}^{N_0} \to \mathbb{R}^{N_{L+1}}$ (see §1.2). However, there are several compelling reasons to study random neural networks:

- The choice of a distribution of network parameters θ_0 at initialization induces a law on the first forward pass $x \mapsto z^{(L+1)}(x; \theta_0)$ and first backward pass $\theta_0 \mapsto \theta_1 = \theta_0 - \eta \nabla_\theta \mathcal{L}(\theta_0)$. Asking that the forward pass and backward pass as "well-behaved" gives a principled way to choose initialization scheme and learning rate. See §2.2 and §2.2.3.

[1] The main exception to this prescription is that for the final layer it is often natural to take $(\sigma_W^2)^{(L+1)} = \sigma_W^2 / N_L^2$. This seemingly small modification of the initialization scheme leads to profound differences during training and is a key difference between the the kernel of lazy limit of neural networks at large width considered in Chapter 3 and the mean-field limit of Chapter 4.

- There any usually many ways for a given neural network to fit the training data. Since optimization is essentially a greedy local search, the statistics of the neural network at initialization provide an important ingredient in determining the sort of function a neural network will learn after training as well. This is especially true in the NTK regime discussed in Chapter 3 and when analyzing Bayesian inference with neural networks, in which case the distribution at initialization is precisely the prior.
- Several influential lines of work on feature learning (primarily in shallow networks) give precise insights about the nature of feature learning from the first step of gradient descent. This analysis, in turn, comes down to computing network statistics at initialization. See the references after §1.3.2.

The remainder of this chapter is organized as follows. First, in §2.2 we derive a simple procedure for how to initialize a neural network so that the first forward pass, i.e. the map $x \mapsto z^{(L+1)}(x)$ with random weights and biases, is well-behaved at both large width N and large depth L. This will naturally lead to a fundamental result: randomly initialized neural networks converge to Gaussian processes in the limit of infinite width (see Theorem 2.2.1). Having analyzed the first forward pass in §2.2, we proceed in §2.2.3 to study the first backward pass, deriving a simple rule for scaling the learning rate as a function of depth and width. In §2.3 we explore the nature of finite width corrections to the Gaussian process behavior of random neural networks in the infinite width limit. We conclude in §2.3.2 with some open questions.

2.2 Random Neural Nets at Fixed Depth and Infinite Width: Gaussian Processes and Hyperparameter Tuning

In this section, we consider a depth L fully connected network, which for each input $x \in \mathbb{R}^{N_0}$ computes an output $z^{(L+1)}(x)$ through an associated sequence of pre-activation vectors $z^{(\ell)}(x) \in \mathbb{R}^{N_\ell}$ given as in Definition 1.1.1 by

$$z^{(\ell+1)}(x;\theta) = \begin{cases} W^{(\ell+1)}\phi(z^{(\ell)}(x)) + b^{(\ell+1)}, & \ell \geq 1 \\ W^{(1)}x + b^{(1)}, & \ell = 0 \end{cases}, \qquad W^{(\ell)} \in \mathbb{R}^{n_\ell \times N_{\ell-1}},\ b^{(\ell)} \in \mathbb{R}^{N_\ell}.$$

At the start of training such a network, we must choose a way to initialize the weights $W_{ij}^{(\ell)}$ and biases $b_i^{(\ell)}$. Virtually all initialization scheme in practice take all weights and biases to be drawn independently from some distribution that is symmetric around 0 (and hence has zero mean). Often, in fact, this distribution is Gaussian and hence we must only determine good choices for the variances

$$\left((\sigma_W^2)^{(\ell)}\right)^2 = \mathrm{Var}\left[W_{ij}^{(\ell)}\right], \qquad \sigma_b^{(\ell)} = \mathrm{Var}\left[b_i^{(\ell)}\right], \tag{2.2.1}$$

which by symmetry should only depend on ℓ. Our first goal is to answer two related questions:

Q1. How should the variance $\left((\sigma_W^2)^{(\ell)}\right)^2, \sigma_b^{(\ell)}$ depend on the layer widths N_ℓ?

Q2. What is the distribution of a random neural network $z^{(L+1)}(x;\theta)$ at large width?

The reason to consider the effect of changing width on initialization scheme is that a common way to increase the number of trainable parameters in practice is simply to increase the width of some or all hidden layers. Thus, it is convenient to have a prescription for initialization that requires little or no tuning of the variance on the part of the practitioner.

To approach Q1, we must decide on a criterion for measuring the quality of an initialization scheme. Of course the dream criterion is that this initialization scheme (together with an appropriate choice of learning rates, batch sizes, and other optimization hyperparameters) leads to trained networks that make reliable and accurate predictions on unseen data. However, such conclusions are usually too much to hope to deduce analytically. A typical alternative in developing principles for choosing initialization and optimization hyperparameters consists of three steps:

1. Posit one or more criteria about the behavior of neural networks at initialization (or say after one step of gradient descent) that capture at least heuristically the idea that we want networks that are both easy to train and can perform feature learning.
2. Validate empirically that these criteria are useful in practice
3. Determine which architectures, initialization schemes, and optimization algorithms provably display these criteria.

Perhaps the simplest criterion to use in step 1 is to ask that for any input $x \in \mathbb{R}^{N_0}$ the neuron pre-activations $z_i^{(\ell)}(x)$ are order 1 random variables at initialization in the sense that

$$\textbf{Criterion}: \qquad \mathbb{E}\left[z_i^{(\ell)}(x)\right] = O(1), \qquad \mathrm{Var}[z_i^{(\ell)}(x)] = \Theta(1), \tag{2.2.2}$$

where the implicit constants are uniform with respect to the choice of widths N_ℓ (and perhaps depth L as well). Since we are considering mean 0 weights and biases, we automatically have

$$\mathbb{E}\left[z_i^{(\ell+1)}(x)\right] = \mathbb{E}\left[b_i^{(\ell+1)} + \sum_{j=1}^{N_\ell} W_{ij}^{(\ell+1)} \phi(z_j^{(\ell)}(x))\right] = 0.$$

Moreover, recalling the notation in (2.2.1), we have

$$\begin{aligned}\mathrm{Var}[z_i^{(\ell+1)}(x)] &= \mathbb{E}\left[\left(b_i^{(\ell+1)} + \sum_{j=1}^{N_\ell} W_{ij}^{(\ell+1)} \phi(z_j^{(\ell)}(x))\right)^2\right] \\ &= \sigma_b^{(\ell)} + N_\ell \left((\sigma_W^2)^{(\ell)}\right)^2 \mathbb{E}\left[\frac{1}{N_\ell}\left|\left|\phi(z^{(\ell)}(x))\right|\right|^2\right].\end{aligned} \tag{2.2.3}$$

From this we at least heuristically derive the simple criterion that (2.2.2) ought to hold if we take

$$\left((\sigma_W^(\ell)\right)^2 := \frac{\sigma_W^2}{N_{\ell-1}}, \qquad \sigma_b^{(\ell)} = \sigma_b, \tag{2.2.4}$$

where $\sigma_b \geq 0, \sigma_W^2 > 0$ are fixed constants. Indeed, if $z_i^{(\ell)}(x)$ are order 1, then so are $\phi(z_i^{(\ell)}(x))$ and hence by (2.2.3) so is $z_i^{(\ell+1)}(x)$. That this guess is correct is confirmed by Theorem 2.2.1, which also answers Q2. To state it, let us agree that, given a $\mu : \mathbb{R}^{N_0} \to \mathbb{R}$ and a kernel $K : \mathbb{R}^{N_0} \times \mathbb{R}^{N_0} \to \mathbb{R}$, we will write $\mathcal{GP}(\mu, K)$ for the Gaussian process with mean μ and covariance K.

Theorem 2.2.1. *Fix L, ϕ, N_0, N_{L+1} and a polynomially bounded $\phi : \mathbb{R} \to \mathbb{R}$. With the initialization (2.2.4), the fields $x \mapsto z^{(L+1)}(x;\theta)$ converge in distribution as $N_1, \ldots, N_L \to \infty$ to a mean zero Gaussian process with iid components:*

$$\lim_{N_1,\ldots,N_L\to\infty} z^{(L+1)}(\cdot\,;\theta) \stackrel{d}{=} \mathcal{GP}(0, K^{(L+1)})^{\otimes N_{L+1}},$$

where

$$\lim_{N_1,\ldots,N_L\to\infty} \mathrm{Cov}\left(z_i^{(\ell+1)}(x), z_j^{(\ell+1)}(x')\right) = \delta_{ij} K^{(L+1)}(x,x'). \tag{2.2.5}$$

Further, the covariance function satisfies the following layer-wise recursion:

$$K^{(\ell+1)}(x,x') = \begin{cases} \sigma_b + \sigma_W^2 \mathbb{E}_{K^{(\ell)}}\left[\phi\left(z_1^{(\ell)}(x)\right)\phi\left(z_1^{(\ell)}(x')\right)\right], & \ell \geq 1 \\ \sigma_b + \frac{1}{N_0} x \cdot x', & \ell = 0 \end{cases}, \tag{2.2.6}$$

where $\mathbb{E}_{K^{(\ell)}}[\cdot]$ denotes the expectation over the distribution in which $z_i^{(\ell)}(x)$ are iid centered Gaussian processes with covariance $K^{(\ell)}$ that are independent for different i.

Remark 2.2.2. *The GP limit holds not only for fully connected architectures for also for virtually any feed-forward architecture. See e.g. [LBN+18, Yan19, HBSDN20]. See [Han23] for a more general statement for fully connected networks.*

2.2.1 Sketch of Proof of Theorem 2.2.1

As with virtually every analysis of random neural networks, the starting point for is the observation that the sequence $\left\{z^{(\ell)}(x),\, \ell = 1, \dots, L+1\right\}$ is a Markov chain. Moreover, the transition probabilities are Gaussian in the sense that, conditional on $z^{(\ell)}$, the fields $x \mapsto z_i^{(\ell+1)}(x)$ are iid mean zero Gaussian processes with conditional covariance

$$\widehat{K}^{(\ell+1)}(x,x') = \text{Cov}\left(z_m^{(\ell+1)}(x),\, z_m^{(\ell+1)}(x') \mid z^{(\ell)}\right) = \sigma_b + \frac{\sigma_W^2}{N_\ell}\sum_{j=1}^{N_\ell} \phi\big(z_j^{(\ell)}(x)\big)\phi\big(z_j^{(\ell)}(x')\big). \quad (2.2.7)$$

Moreover, the conditional covariance $\widehat{K}^{(\ell+1)}(x,x')$ is an example of a *collective observable*

$$\mathcal{O}_f^{(\ell)} := \frac{1}{N_\ell}\sum_{i=1}^{N_\ell} f\left(z_i^{(\ell)}(x),\ x \in A\right),$$

where A is some finite subset of $|A|$ point in $\mathbb{R}^{N_0}$ and $f : \mathbb{R}^{|A|} \to \mathbb{R}$ is a polynomially bounded function. The key technical result is the following:

Theorem 2.2.3 (Centered Moments of Collective Observables)**.** *Suppose $\phi : \mathbb{R} \to \mathbb{R}$ is polynomially bounded. For $k \geq 1$ suppose that $f_1, \dots, f_k : \mathbb{R}^{|A|} \to \mathbb{R}$ are polynomially bounded. Then*

$$\mathbb{E}\left[\prod_{j=1}^{k}\left(\mathcal{O}_{f_j}^{(\ell)} - \mathbb{E}\left[\mathcal{O}_{f_j}^{(\ell)}\right]\right)\right] = O\left(N^{-\lceil \frac{k}{2} \rceil}\right), \quad (2.2.8)$$

where

$$N := \min\{N_1, \dots, N_\ell\}$$

and the implied constant depends on k, ℓ, f_j, A, ϕ.

This result is proved in [Han24] (see Theorem 3.1 and Lemma 7.5). While the proof is somewhat involved, the intuition is simple: each collective observable $\mathcal{O}_f^{(\ell)}$ is a an average of identically distribution random variables. If they were independent then the estimate (2.2.8) is straightforward. When $\ell \geq 2$, the collections $\left\{z_i^{(\ell)}(x),\, x \in A\right\}$ are not quite independent for different i. But one may check by induction on ℓ that they are sufficiently weakly correlated that the estimate (2.2.8) still holds.

Theorem 2.2.1 follows almost immediately from Theorem 2.2.3. Indeed, we already said just before (2.2.7) that, for different i, the fields $z_i^{(\ell)}(x)$ can be viewed as independent Gaussian processes with the same random covariance $\widehat{K}^{(\ell)}$. However, by Theorem 2.2.3, the random covariance $\widehat{K}^{(\ell)}$ converges in probability to some fixed covariance $K^{(\ell)}$ as the minimal hidden layer width n tends to infinity. In this limit, $z_i^{(\ell)}(x)$ therefore converges to independent centered Gaussian processes with covariance $K^{(\ell)}$. □

2.2.2 Weight/bias variances and learning rates via Theorem 2.2.1

Theorem 2.2.1 points to a tremendous simplification of neural networks at the infinite width. For instance, the infinite width covariance structure (2.2.5) shows that components of the network output $z_i^{(L+1)}(x)$ are independent for different i. It also confirms that the ansatz (2.2.4) leads to initializations in which every neuron $z_i^{(\ell)}(x)$ is an order 1 random variable at large width. Having understood how to the initialization scale should vary with network width, we now to turn to the natural followup

Question: Which σ_W^2, σ_b give well-behaved networks at large depth?

The practical motivation is that one can increase network size by scaling both depth and width. If theory can provide prescriptions for setting σ_W^2, σ_b in a way that gives rise to well-behaved networks at any depth and width, then one can avoid in practice a potentially costly search for a good initialization.

At least at infinite width and with the initialization schemes (2.2.4), the behavior of the field $x \mapsto z^{(L+1)}(x;\theta)$ at the start of training is completely determined by $\sigma_b, \sigma_W^2, \phi$ through the recursion for (2.2.6). Since this recursion is non-linear due to the presence of ϕ, we expect that for generic choices of σ_b, σ_W^2 the kernel $K^{(\ell)}$ will either grow or converge to a fixed point exponentially in ℓ. This gives us the second well-studied criterion for choose initializations:

$$\textbf{Criterion}: \quad K^{(\ell)}(x,x') \text{ neither grows nor decays exponentially in } \ell. \tag{2.2.9}$$

A variety of prior articles have worked out how to do this [PLR+16, SGGSD17, RYH22]. The basic idea is to fix inputs x, x' and view the recursion (2.2.6) as a discrete time dynamical system for

$$(K^{(\ell)}(x,x), K^{(\ell)}(x',x'), K^{(\ell)}(x,x')) \quad \mapsto \quad (K^{(\ell+1)}(x,x), K^{(\ell+1)}(x',x'), K^{(\ell+1)}(x,x')).$$

One then seeks σ_b, σ_W^2 for which this dynamical system has critical stable fixed points so that $K^{(\ell)}$ approaches a fixed point polynomially, rather than exponentially, in ℓ. This is sometimes called *tuning to criticality*. We indicate here only the very simplest example of this analysis in which we take $\phi(t) = t$. In this setting, the variance recursion is affine

$$K^{(\ell+1)}(x,x') = \sigma_b + \sigma_W^2 K^{(\ell)}(x,x').$$

Combining the fixed point condition

$$\exists K_* \text{ s.t. } K_* = \sigma_b + \sigma_W^2 K_*$$

with the criticality requirement

$$\left.\frac{\partial K^{(\ell)}(x,x)}{\partial K^{(\ell)}(x,x)}\right|_{K^{(\ell)}(x,x)=K_*} = 1$$

yields the initialization $\sigma_b = 0, \sigma_W^2 = 1$. We leave it to the reader as an interesting exercise to derive the so-called *Kaiming He initialization*, which prescribes $\sigma_b = 0, \sigma_W^2 = 2$ for the important special case or ReLU activations $\phi(t) = \max\{0, t\}$.

2.2.3 Analyzing one step of GD to set the learning rate

So far, given a large network width N, a potentially large network depth L, and a non-linearity ϕ we've outlined one principled method for selecting an initialization scheme. The next logical task is to determine a prescription for setting learning rates. We set biases to 0 and focus only on deriving learning rates for network weights. Recall that the gradient descent update after t steps of gradient descent takes the form

$$W_{ij}^{(\ell+1)}(t+1) = W_{ij}^{(\ell+1)}(t) - \eta_W^{(\ell)} \partial_{W_{ij}^{(\ell)}} \mathcal{L}(\theta(t)). \tag{2.2.10}$$

Here $\eta_W^{(\ell)}$ is the learning rate for weights in layer ℓ, which by symmetry depends only on ℓ.[2] What then should our criterion be for choosing $\eta_W^{(\ell)}$? A natural choice, which is either implicitly or explicitly adopted in a range of prior work [Yai22, RYH22, HN20a, YH21] is the following:

$$\textbf{Criterion:} \quad \text{Choose } \eta_W^{(\ell)} \text{ so network output changes by order 1 uniformly in depth and width from one GD step.} \tag{2.2.11}$$

[2] In practice it is common to choose the same learning rate for all parameters. However, it is also common to experiment with adaptive first order methods which implicitly scale the learning rate for different parameters in different ways.

As we will see in Chapter 4 this Criterion is not sufficient by itself to ensure various desirable properties of the neural network training at large width. For now, however, we will try to satisfy (2.2.11) in the simplest case of output dimension $N_{L+1} = 1$. To first order in the learning rate we have

$$\begin{aligned} &z^{(\ell+1)}(x;\theta(t+1)) - z^{(\ell+1)}(x;\theta(t)) \\ &\quad\approx \sum_{\ell=1}^{L+1}\sum_{i=1}^{n_\ell}\sum_{j=1}^{N_{\ell-1}} \partial_{W_{ij}^{(\ell)}} z(x;\theta(t)) \left(W_{ij}^{(\ell)}(t+1) - W_{ij}^{(\ell)}(t)\right) \\ &\quad= -\sum_{\ell=1}^{L+1}\sum_{i=1}^{n_\ell}\sum_{j=1}^{N_{\ell-1}} \partial_{W_{ij}^{(\ell)}} z(x;\theta(t)) \partial_{W_{ij}^{(\ell)}} \mathcal{L}(\theta(t)). \end{aligned}$$

Recall $\mathcal{L}$ is an empirical loss over a training dataset

$$\mathcal{L}(\theta) = \sum_{(x,y)\in\mathcal{D}} \ell(z^{(L+1)}(x;\theta), y)$$

and depends on the network parameters θ only via the values of the network outputs $z(x;\theta)$ for x in the training dataset. Thus, by the chain rule we find to first order in the learning rate

$$z^{(\ell+1)}(x;\theta(t+1)) - z^{(\ell+1)}(x;\theta(t)) = -\sum_{(x',y')\in\mathcal{D}} H_\eta(x,x';\theta(t))\ \partial_z \ell(z,y)\big|_{z=z^{(L+1)}(x';\theta(t)),\, y=y'},$$

Here, for any setting of parameters θ we've denoted by $H_\eta(x,x';\theta)$ is the learning rate adjusted NTK (see (3.2.3))

$$H_\eta(x,x';\theta) = \left(\eta \odot \nabla_\theta z^{(L+1)}(x;\theta)\right)^T \left(\nabla_\theta z^{(L+1)}(x';\theta)\right),$$

where

$$\eta \odot \nabla_\theta z^{(L+1)}(x;\theta) = \left(\eta_W^{(\ell)} \partial_{W_{ij}^{(\ell)}} z^{(L+1)}(x;\theta)\right)_{\substack{\ell=1,\dots,L+1\\ i=1,\dots,N_\ell\\ j=1,\dots,N_{\ell-1}}}$$

is the Hadamard product between the vector η of learning rates and the gradient $\nabla_\theta z^{(L+1)}(x;\theta)$. We'll say much more about the NTK in Chapter 3 below. Note that $\partial_z \ell(z,y)$ evaluated at $z = z^{(L+1)}(x';\theta(t))$, $y = y'$ depends only on the value $z^{(L+1)}(x';\theta(t))$ and the data label y'. Due to our initialization scheme, it is order 1 at initialization $t = 0$ independent of depth and width. Hence, the proposed learning rate Criterion (2.2.11) requires that, at initialization,

$$\mathbb{E}\left[H_\eta(x,x;\theta(0))\right] = \Theta(1)$$

with the implicit constant is independent of N_ℓ, L. Note that

$$\mathbb{E}\left[\Theta(x,x;\theta(0))\right] = \sum_{\ell=1}^{L+1}\sum_{i=1}^{N_\ell}\sum_{j=1}^{N_{\ell-1}} \eta_W^{(\ell)} \mathbb{E}\left[\left(\partial_{W_{ij}^{(\ell)}} z^{(L+1)}(x;\theta(0))\right)^2\right].$$

This is ensured by setting

$$\eta_W^{(\ell)} := \left(LN_\ell N_{\ell-1}\mathbb{E}\left[\left(\partial_{W_{ij}^{(\ell)}} z^{(L+1)}(x;\theta(0))\right)^2\right]\right)^{-1}.$$

A simple computation at large width shows that

$$\mathbb{E}\left[\left(\partial_{W_{ij}^{(\ell)}} z^{(L+1)}(x;\theta(0))\right)^2\right] = \Theta(n_\ell^{-1}).$$

This suggests setting

$$\eta_W^{(\ell)} = \Theta(N_{\ell-1}^{-1} L^{-1}).$$

For constant depth networks this is precisely the what is proposed in so-called standard parameterization.

2.3 Finite Width Correction to the Gaussian Process Limit

In the previous sections §2.2 and §2.2.3 we analyzed random neural networks in the limit where first the hidden layer width N_ℓ tends to infinity and then the depth L grows. However, as we already saw in §1.3.3, the limits as width and depth tend to infinity do not commute. As a result, a refined understanding of fully connected networks at initialization requires studying the finite width corrections to the Gaussian process regime and their dependence on L. Prior work on this question includes [Han18, HN20b, HN20a, LNR22, BCCZ23, RYH22, Yai20] can be very roughly summarized by the following:

Message: The depth-to-width ratio $\dfrac{1}{N_1}+\cdots+\dfrac{1}{N_L}\simeq\dfrac{L}{N}$ measures effective depth.

More precisely, recall that in the infinite width limit $N_\ell\to\infty$ neurons pre-activations converge to independent Gaussians. It turns out that L/N controls the deviations from this regime, measuring both correlations between neurons and the non-Gaussianity of single-neuron distributions. Moreover, L/N controls also the numerical stability of optimization (the variance of the gradients in the first step of training) [Han18, RYH22, Han24] and extent of non-linearity of the model (measured by the spectral norm of the parameter Jacobian) [HN20a]. This suggests an interesting and still partly conjectural phase diagram (see Figure 1.6).

Our goal here is to give a glimpse into how such results might be derived by focusing on perhaps the simplest instance. Specifically, let us consider the recursion relation (2.2.6), which describes the infinite width Gaussian Process covariance $K^{(\ell+1)}$ in terms of $\sigma_b, \sigma_W^2, \phi$ and $K^{(\ell)}$. Perhaps the simplest setting in which to probe the finite width correction to this recursion is to analyze fourth cumulant at a single input

$$\kappa_4^{(\ell)}(x)=\frac{1}{3}\kappa\left(z_1^{(\ell)}(x),z_1^{(\ell)}(x),z_1^{(\ell)}(x),z_1^{(\ell)}(x)\right)=\frac{1}{3}\left(\mathbb{E}\left[\left(z_1^{(\ell)}(x)\right)^4\right]-3\mathbb{E}\left[\left(z_1^{(\ell)}(x)\right)^2\right]^2\right).$$

A simple computation shows that $\kappa_4^{(\ell)}(x)$ captures both non-Gaussian fluctuations

$$\mathrm{Var}\left[\left(z_i^{(\ell)}(x)\right)^2\right]=3\kappa_4^{(\ell)}(x)+2\mathbb{E}\left[\left(z_i^{(\ell)}(x)\right)^2\right]^2$$

and inter-neuron correlations

$$\kappa_4^{(\ell)}(x)=\mathrm{Cov}\left(\left(z_i^{(\ell)}(x)\right)^2,\left(z_j^{(\ell)}(x)\right)^2\right),\qquad i\neq j.$$

Since as $N_1,\cdots,N_L\to\infty$, neurons are independent and Gaussian, we have that

$$\lim_{N_1,\ldots,N_{\ell-1}\to\infty}\kappa_4^{(\ell)}(x)=0.$$

Theorem 2.3.1 shows that while in the large width and fixed depth limit $n\to\infty$ the fourth cumulant $\kappa_4^{(\ell)}(x)$ tends to zero at a rate like $1/N$, it actually grows linearly in the next worth. This is an example of how the large depth and large width limits do not commute, with depth accentuating finite-width effects.

This recursion is the beginning of a perturbatively solvable hierarchy of recursions that can be used to describe virtually every observable associated with a random neural network [Han24]. This includes the higher cumulants of the functions computed by network neurons, and we'll see an example below of how cumulants of order k in layer $\ell+1$ are determined only via cumulants of order $j\leq k$ at layer ℓ.

Theorem 2.3.1 ([Han24, RYH22, Yai20, BCCZ23]). *Fix L, N_0, N_{L+1}, ϕ. Suppose that the weights and biases are chosen as in* (2.1.2) *and that*

$$N_1,\ldots,N_L\simeq N\gg 1.$$

The fourth cumulant is of order $O(n^{-1})$ and satisfies the following recursion:

$$\kappa_4^{(\ell+1)}(x) = \frac{\sigma_W^2}{N_\ell}\mathrm{Var}_{K^{(\ell)}}\left[\phi^2\right] + \left(\chi_{||}^{(\ell)}(K^{(\ell)}(x,x))\right)^2 \kappa_4^{(\ell)}(x) + O(n^{-2}), \tag{2.3.1}$$

where

$$\chi_{||}^{(\ell)} := \mathbb{E}_{K^{(\ell)}}\left[\frac{\sigma_W^2}{2}\frac{\partial^2}{\partial z^2}\bigg|_{z=z_i^{(\ell)}(x)}\phi^2(z)\right].$$

Thus, at criticality and uniform width ($N_\ell = N$), we have

$$\frac{\kappa_4^{(L+1)}(x)}{\left(K^{(L+1)}(x,x)\right)^2} = C_\phi\frac{L}{N} + O_{L,\phi}(n^{-2}).$$

Moreover, for any fix $m \geq 1$ and any "reasonable" function $f : \mathbb{R}^m \to \mathbb{R}$ we may write

$$\begin{aligned}
&\mathbb{E}\left[f\left(z_i^{(\ell)}(x),\, 1 \leq i \leq m\right)\right] \\
&\quad = \mathbb{E}_{G^{(\ell)}}\left[f\left(z_i^{(\ell)}(x),\, 1 \leq i \leq m\right)\right] \\
&\quad + \frac{\kappa_4^{(\ell+1)}(x)}{8}\mathbb{E}_{K^{(\ell)}}\left[\left(\sum_{j=1}^m \partial_{z_j}^4 + \sum_{\substack{j_1,j_2=1\\ j_1\neq j_2}}^m \partial_{z_{j_1}}^2\partial_{z_{j_2}}^2\right) f\left(z_i^{(\ell)}(x),\, 1 \leq i \leq m\right)\right] \\
&\quad + O(n^{-2}).
\end{aligned} \tag{2.3.2}$$

Here, $G^{(\ell)}$ is the dressed two point function

$$G^{(\ell)}(x,x') := \mathbb{E}\left[z_1^{(\ell)}(x)z_1^{(\ell)}(x')\right].$$

This Theorem is originally derived in a physics way in the breakthrough paper of Yaida [Yai20]. It was then rederived, again at a physics level of rigor in Chapter 4 of [RYH22]. Finally, it was derived in a somewhat different, and more mathematical, way in [Han24]. This theorem can be interpreted as showing that the statistics of random neural networks at initialization be recast as perturbatively solvable hierarchies. The perturbative parameter is $1/N$ and the hierarchies are recursions in ℓ.

Proof Sketch of Theorem 2.3.1

For this proof, since we will suppress the network input x from out notation. So for instance $\widehat{K}^{(\ell)}(x,x)$ will simply be denoted by $\widehat{K}^{(\ell)}$. Since $\widehat{K}^{(\ell)}$ is a collective observable, it makes sense to consider

$$G^{(\ell)} := \mathbb{E}\left[\widehat{K}^{(\ell)}\right], \qquad \Delta^{(\ell-1)} := \widehat{K}^{(\ell)} - \mathbb{E}\left[\widehat{K}^{(\ell)}\right].$$

The scalar $G^{(\ell)}$ is sometimes referred to as a dressed two point function. Note that

$$\kappa_4^{(\ell)} = \mathbb{E}\left[\left(\Delta^{(\ell-1)}\right)^2\right]$$

Just as before, we have

$$\begin{aligned}
\mathbb{E}\left[f(z_i^{(\ell)},\, i = 1,\ldots,m)\right] &= \int_{\mathbb{R}^{n_m}} \widehat{f}\,(\xi)\,\mathbb{E}\left[e^{-\frac{1}{2}||\xi||^2\widehat{K}^{(\ell)}}\right]d\xi \\
&= \int_{\mathbb{R}^{n_m}} \widehat{f}\,(\xi)\,e^{-\frac{1}{2}||\xi||^2 G^{(\ell)}}\mathbb{E}\left[e^{-\frac{1}{2}||\xi||^2\Delta^{(\ell-1)}}\right]d\xi.
\end{aligned}$$

Applying Theorem 2.2.3 we may actually Taylor expand to find a power series expansion in $1/N$:

$$\mathbb{E}\left[e^{-\frac{1}{2}||\xi||^2\Delta^{(\ell-1)}}\right] = \sum_{q\geq 0}\frac{(-1)^q}{2^q q!}||\xi||^{2q}\,\mathbb{E}\left[\left(\Delta^{(\ell-1)}\right)^q\right] = 1+\frac{1}{8}||\xi||^4\,\mathbb{E}\left[\left(\Delta^{(\ell-1)}\right)^2\right]+O(N^{-2}).$$

Putting this all together yields

$$\mathbb{E}\left[f(z_i^{(\ell)},\, i=1,\ldots,m)\right] = \int_{\mathbb{R}^{nm}}\left(1+\frac{1}{8}||\xi||^4\,\mathbb{E}\left[\left(\Delta^{(\ell-1)}\right)^2\right]\right)\widehat{f}\,(\xi)\,e^{-\frac{1}{2}||\xi||^2 G^{(\ell)}}d\xi+O(N^{-2}).$$

In particular, we obtain

$$\mathbb{E}\left[f(z_i^{(\ell)},\, i=1,\ldots,m)\right] = \mathbb{E}_{G^{(\ell)}}[f]+\frac{1}{8}\mathbb{E}\left[\left(\Delta^{(\ell-1)}\right)^2\right]\mathbb{E}_{G^{(\ell)}}\left[\left(\sum_{j=1}^m \partial^2_{z_j^{(\ell)}}\right)^2 f\right]+O(N^{-2}). \quad (2.3.3)$$

A direct computation now shows that

$$\mathbb{E}_{G^{(\ell)}}[f] = \mathbb{E}_{K^{(\ell)}}[f]+O(N^{-1}).$$

This proves (2.3.2). Next, recall that

$$\kappa_4^{(\ell+1)}(x) = \mathbb{E}\left[\left(\Delta^{(\ell)}\right)^2\right].$$

Moreover,

$$\mathbb{E}\left[\left(\Delta^{(\ell)}\right)^2\right] = \frac{1}{N_\ell}\mathbb{E}\left[\left(X_{1;\alpha}^{(\ell)}\right)^2\right]+\left(1-\frac{1}{N_\ell}\right)\mathbb{E}\left[X_{1;\alpha}^{(\ell)}X_{2;\alpha}^{(\ell)}\right],$$

where

$$X_{j;\alpha}^{(\ell)} := \sigma_W^2\left(\phi(z_j^{(\ell)})^2-\mathbb{E}\left[\phi(z_j^{(\ell)})^2\right]\right).$$

Applying (2.3.3) and some algebra completes the proof of (2.3.1). □

2.3.1 Non-linearity Shaping at Large Depth

In this section we briefly outline a complementary approach to the the description of finite-width corrections to the Gaussian process behavior of random neural networks developed first in [LNR22]. To set the stage recall that according to Theorem 2.2.1 for *linear activations* $\phi(t)=t$ and $\sigma_b = 0, \sigma_W^2 = 1$ we found

$$K^{(\ell+1)}(x,x') = \sigma_W^2\mathbb{E}_{K^{(\ell)}}[z_1(x)z_1(x')] = \sigma_W^2 K^{(\ell)}(x,x').$$

Hence, the two covariance function for the infinite width Gaussian process is independent of depth. For ReLU activations, a simple computation shows that setting $\sigma_b = 0, \sigma_W^2 = 2$ also gives

$$K^{(\ell+1)}(x,x) = K^{(\ell)}(x,x).$$

However, while the variance is constant *on the diagonal*, the correlation operator

$$C^{(\ell)}(x,x') := \frac{K^{(\ell)}(x,x')}{\sqrt{K^{(\ell)}(x,x)K^{(\ell)}(x',x')}}$$

degenerates at large ℓ in the sense that

$$C^{(\ell)}(x,x') = 1+O(\ell^{-2}).$$

In other words, the infinite depth limit of the infinite width Gaussian processes for ReLU networks is degenerate in the sense that given any two inputs x, x' of the same norm, the Gaussian field assigns to both of them the same random (Gaussian) constant. Similar pathologies are present in other non-linearities as well. The key idea from [LNR22, MBD+21] is to consider *shaped, width-dependent activations* that are close in every layer to being linear. For example, we may consider shaped tanh and ReLU activations

$$\phi_{\text{shaped tanh}}(t) = \sqrt{N}\tanh(t/\sqrt{N}) \simeq t + \frac{\text{const}}{N}\cdot t^3 + O(N^{-2})$$
$$\phi_{\text{shaped ReLU}}(t) = t + \frac{\text{const}}{\sqrt{N}}\cdot \max\{0, t\}.$$

The shaping strength above is chosen precisely so that the aggregate difference between randomly initialized networks with shaped activations and the deep linear networks with $\phi(t) = t$ are order 1 for any fixed value of the depth-to-width ratio L/N. Indeed, the key result from [LNR22] is that the precise shaping in the previous displayed equations results in well-posed SDEs

$$d\Phi_{\alpha\beta}(\tau) = b(\Phi_{\alpha\alpha}(\tau), \Phi_{\beta\beta}(\tau), \Phi_{\alpha\beta})(\tau))d\tau + \Sigma^{1/2}(\Phi_{\alpha\alpha}(\tau), \Phi_{\beta\beta}(\tau), \Phi_{\alpha\beta})(\tau))dW(\tau) \tag{2.3.4}$$

for the finite width empirical covariance operator $\Phi_{\alpha\beta}$

$$\Phi_{\alpha\beta}(\tau) = \frac{1}{N}\left\langle z^{(\ell)}(x_\alpha), z^{(\ell)}(x_\beta)\right\rangle$$

in which $dW(\tau)$ is a standard Brownian motion, b, Σ are explicit drift and covariance functions, and

$$\tau = \ell/N, \quad \ell = 0, \ldots, L$$

is an effective *layer time*. The so-called *neural covariance SDE* (2.3.4) is an important tool for studying Bayesian inference with deep and width neural networks.

2.3.2 Open Problems

We conclude this Chapter we some open problems about neural networks at initialization. For these problems we consider a fully connected network $x \in \mathbb{R}^{N_0} \mapsto z^{(L+1)}(x) \in \mathbb{R}^{N_{L+1}}$ with

$$z^{(\ell+1)}(x) = \begin{cases} \frac{1}{\sqrt{N_\ell}} W^{(\ell+1)}\phi\left(z^{(\ell)}(x)\right) \in \mathbb{R}^{N_{\ell+1}}, & \ell \geq 1 \\ \frac{1}{\sqrt{N_0}} W^{(1)} x \in \mathbb{R}^{N_1}, & \ell = 0 \end{cases}$$

and $W_{ij}^{(\ell)} \sim N(0,1)$ iid. Throughout we will assume that N_0, N_{L+1} are fixed and that

$$N_1, \ldots, N_L \simeq N \to \infty.$$

Open Problem 1. *Fix $L \geq 1$ and $k \geq 1$. Suppose for simplicity that $N_{L+1} = 1$. Compute the rate of convergence to 0 of the joint cumulants of neuron pre-activations and partial Jacobians*

$$\left(z_{i_1}^{(\ell_1)}(x_1), \ldots, z_{i_m}^{(\ell_m)}(x_m), \frac{\partial z^{(L+1)}(x)}{\partial z_{i_{m+1}}^{(\ell_{m+1})}(x)}, \ldots, \frac{\partial z^{(L+1)}(x_{m+1})}{\partial z_{i_{2k}}^{(\ell_{2k})}(x_{2k})}\right)$$

and across different layers as $N \to \infty$. Can one obtain a layer-wise recursion for the cumulants similar to (2.3.1)*?*

Remark 2.3.2. *It was shown in [Han23] that when $m = 2k$ (i.e. no partial Jacobians) and all neurons are in the same layer (i.e. $\ell_j = \ell$), then the $2k$-th cumulants go to zero at a rate $1/N^k$. Moreover that same article, building on [Yai20, RYH22] obtained explicit but somewhat complicated layerwise recursion for these cumulants. A nice diagrammatic method for deriving these recursions was obtained in [BCCZ23].*

The next open problem we highlight is a conjecture from [DGA]. To state it's let's write

$$T(x;z) = z^{(L+1)}(x)$$

and denote more generally by

$$T_{\mu_1\dots\mu_k}(x;z) = \frac{\partial^k}{\partial\theta^{\mu_1}\cdots\partial\theta^{\mu_k}} z^{(L+1)}(x)$$

the partial derivative of the network output with respect to a fixed set of parameters. Given any even $k, m \geq 0$ and a partition $\pi \in S_{k_m}$ the authors of [DGA] then define for any a correlation function as to be an expectation of the form

$$C_\pi(x_1,\dots,x_m) = \sum_{\mu_1,\dots,\mu_{k_m}=1}^{\#\text{params}} \delta_{\mu_{\pi(1)}\mu_{\pi(2)}}\cdots\delta_{\mu_{\pi(k_m-1)}\mu_{\pi(k_m)}} \mathbb{E}\left[T_{\mu_1\dots\mu_k}(x_1)\cdots T_{\mu_{k_{m-1}+1}\dots\mu_{k_m}}(x_m)\right],$$

where the expectation is over initialization and $\delta_{\pi(a)\pi(b)}$ means that a, b are in the same block of the parition π. The goal is to understand the order of C as a power of the neural network width $N \gg 1$. To do this [DGA] introduced for each correlation function a *cluster graph*, i.e. a graph $G = (V, E)$ with vertex set $V = \{x_1, \dots, x_m\}$ and an edge from x_i to x_j if there exists $a \in \{k_{i-1}+1, \dots, k_i\}$ and $b \in \{k_{j-1}+1, \dots, k_j\}$ so that $\pi(a) = \pi(b)$.

Open Problem 2. *Fix L, N_0 and set $N_{L+1} = 1$. Prove that as $N \to \infty$ we have*

$$C_\pi(x_1,\dots,x_m) = O\left(N^{n_e+n_0/2-m/2}\right),$$

where n_e and n_o are the number of even and odd, respectively, connected components of G.

Remark 2.3.3. *The article [DGA] showed that this conjecture holds for linear networks and for ReLU networks when $x_1 = \cdots = x_m$.*

To understand the motivation for this conjecture recall that under gradient flow on mean squared error we have

$$\frac{d}{dt} z^{(L+1)}(x') = \frac{1}{\mathcal{D}} \sum_{(x,y)\in\mathcal{D}} \Delta_{(x,y);\theta_t} \mathrm{NTK}(x, x'; \theta_t), \tag{2.3.5}$$

where $\Delta_{(x,y);\theta_t}$ is the residual at time t and by definition

$$\mathrm{NTK}(x,x') = \sum_{\mu=1}^{\#\text{params}} \partial_{\theta_\mu} z^{(L+1)}(x) \partial_{\theta_\mu} z^{(L+1)}(x').$$

Thus, $\mathbb{E}[\mathrm{NTK}(x,x')]$ itself and also $\mathbb{E}\left[\frac{d}{dt} z^{(L+1)}\right]$ are both correlation functions in the nomenclature of [DGA]. More generally, a direct computation using (2.3.5) shows that derivatives of correlation functions are again correlation functions. Thus, understanding the order in N of various correlation functions allows one to organize in powers of $1/N$ time-derivatives of virtually any macroscopic observable (one that concerns time derivatives of network outputs and multiple sums over all parameters of derivatives of network outputs).

The preceding problems concerned the finite-dimensional distributions of the neuron pre-activations and partial Jacobians. There is a beautiful and important line of work starting with [FW20, BES+22, LHY25, CPD+24, MLHD23] that studies the distribution of eigenvalues and eigenvectors for both the covariance kernel

$$\Phi^{(\ell)}(x,x') = \frac{1}{N_\ell}\left\langle z^{(\ell)}(x), z^{(\ell)}(x')\right\rangle$$

and the NTK

$$\mathrm{NTK}^{(L+1)}(x,x') = \left\langle \nabla_\theta z^{(L+1)}(x), \nabla_\theta z^{(L+1)}(x') \right\rangle .$$

These articles study mainly the empirical kernels

$$\Phi^{(\ell)}_{\mathcal{D}} := \left(\Phi^{(\ell)}(x,x')\right)_{(x,y),(x',y)\in\mathcal{D}}, \qquad \mathrm{NTK}^{(L+1)}_{\mathcal{D}} := \left(\mathrm{NTK}^{(L+1)}(x,x')\right)_{(x,y),(x',y)\in\mathcal{D}}$$

at initialization or after one step of training and for either isotropic inputs $x \sim N(0, I_{N_0})$ or for inputs from Gaussian mixture models. An interesting open problem is to extend this analysis to anisotropic Gaussians

Open Problem 3. *Let $\Sigma = \Sigma_{N_0}$ be a sequence indexed by $N_0 \geq 1$ of $N_0 \times N_0$ psd matrices. Suppose $x_i \sim \mathcal{N}(0,\Sigma)$. Describe the empirical distribution of eigenvalues of the empirical feature kernel*

$$\Phi^{(\ell)} = \left\{(\Phi^{(\ell)}(x_i,x_j))\right\}_{i,j=1,\dots,n}$$

and similarly for the empirical NTK kernel as $n, N_0, N \to \infty$.

The motivation for this problem is that as a surrogate for studying the properties of deep neural networks it is common to first study associated linear models in high dimensions [GMMM19, HMRT22, MM23, MS24, AZVP24]. The analysis of the empirical kernels in the previous open problem is the key to understanding regression with corresponding random feature models. The final open problem seeks to repeat this analysis but after one step of training

Open Problem 4. *Let $\Sigma = \Sigma_{N_0}$ be a sequence indexed by $N_0 \geq 1$ of $N_0 \times N_0$ psd matrices. Suppose $x_i \sim \mathcal{N}(0,\Sigma)$ and*

$$y_i = f(x_i) + \epsilon_i, \qquad \epsilon_i \sim \mathcal{N}(0,\sigma_\epsilon^2).$$

Describe the empirical distribution of eigenvalues of the empirical feature kernel

$$\Phi^{(\ell)} = \left\{(\Phi^{(\ell)}(x_i,x_j))\right\}_{i,j=1,\dots,n}$$

and similarly for the empirical NTK kernel after one step of GD on $\mathcal{L}$ as $n, N_0, N \to \infty$.

Remark 2.3.4. *An interesting aspect of this problem is to understand how the answers differ between networks with NTK and mean-field parameterizations.*

Chapter 3

Neural Nets in the Kernel Regime

3.1 Chapter Overview

The parameters θ of a neural network $z^{(L+1)}(x;\theta)$ are typically optimized by gradient descent to (approximately) minimize an empirical loss

$$\mathcal{L}(\theta) = \frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} \ell(z^{(L+1)}(x,\theta), y)$$

over a training dataset $\mathcal{D}$. Even when the per-sample loss $\ell(\cdot,\cdot)$ is simple, since neural networks are non-linear in their parameters, the objective $\mathcal{L}$ is typically a complicated function of θ. Correspondingly, analyzing the properties of gradient-based optimization is a challenging topic core to deep learning theory. The purpose of the present and next chapters is to present some of the theory developed around understanding neural network training dynamics.

Our starting point in the present chapter is the discussion in §3.2 of an important line of work that studies gradient-based neural network optimization via the so-called neural tangent kernel (NTK) [DLT$^+$18, JGH18, LZB22, ACH18, DZPS19]. For notational simplicity we will focus on the case of networks with output dimension 1. The main idea, alluded to already §1.3.1 (see (1.3.5)), is that we expect for overparameterized networks

$$\text{the Jacobian } D_\theta z^{(L+1)}(\mathcal{D};\theta) \in \mathbb{R}^{\#\text{parameters}\times|\mathcal{D}|} \text{ has full rank,}$$

where

$$z^{(L+1)}(\mathcal{D};\theta) = \Big(z^{(L+1)}(x,\theta)\Big)_{(x,y)\in\mathcal{D}} \in \mathbb{R}^{|\mathcal{D}|}.$$

Thus, we expect the *neural tangent kernel*, i.e. the dataset-by-dataset Gram matrix

$$H_{\mathcal{D}}(\theta) := \big(H(x,x';\theta)\big)_{\substack{(x,y)\in\mathcal{D}\\(x',y')\in\mathcal{D}}}, \qquad H(x,x';\theta) := \Big(D_\theta z^{(L+1)}(x;\theta)\Big)^T D_\theta z^{(L+1)}(x';\theta)$$

to be positive definite for most values of θ. We connect the positivity of this matrix to the success of optimization in §3.2. While the NTK is well-defined for any neural network we focus here on using it to understand optimization in the so-called kernel or lazy regime, obtained by studying fully connected networks with fixed depth, fixed dataset size, growing width, and small learning rate. In this setting, as we shall see in §3.2.2, it turns out that the entire trajectory of optimization by gradient descent of a neural network $z^{(L+1)}(x;\theta)$ is very close to that of its linearization around initialization:

$$z^{\text{lin},(L+1)}(x;\theta) := z^{(L+1)}(x;\theta_0) + D_\theta z^{(L+1)}(x;\theta_0)\,(\theta-\theta_0)\,.$$

This was first observed in the special case of one layer ReLU networks in [DLT$^+$18] and developed more fully in [JGH18] (see also [ACH18, LGJ20, BMR21]). Viewing $x \mapsto D_\theta z^{(L+1)}(x;\theta_0)$ as a

feature map, the kernel underlying this associated linear model is the NTK, and we will give a heuristic explanation in §3.2.2 for why in the limit of infinite width it converges almost surely to deterministic kernel

$$\overline{H}(x, x') = \lim_{\text{width}\to\infty} \mathbb{E}\left[H(x, x'; \theta_0)\right].$$

This is a spectacular simplification in which neural networks become linear models. All properties of neural networks in this regime are therefore determined by studying the eigenvalues and eigenfunction of $\overline{H}(x, x')$, which is completely determined by studying networks at initialization.

The mathematical simplicity of neural networks in the NTK regime comes at a steep explanatory cost. Namely, linear models cannot learn data-dependent features, which are the hallmark of successful neural networks in practice. While the NTK regime sheds some light on the success of non-convex optimization it does not allow one to address and make predictions about learning in realistic networks. To do this, one must consider other scaling limits of neural networks. A variety of approaches are possible:

- **Networks at Finite Width.** In neural networks at finite width n the NTK is not constant during training. Its dynamics at fixed depth were worked out, as a perturbative series in $1/N$, in the breakthrough article [HY20]. The resulting dynamics equations are rather intricate and difficult to analyze in detail.
- **Networks with Comparable Width and Depth.** Recall from §2.3 that while random neural networks at initialization converge to Gaussian processes in the regime of fixed depth and infinite width, the $1/N$ corrections to this regime actually scale grow with depth and are more accurately reported by considering the depth-to-width ratio L/N. As we'll briefly explain §2.3, this analysis extends to optimization as well: both the fluctuations at initialization and the change in the NTK in the course of training grow with L/N. See [HN20a, RYH22].
- **Mean Field Parameterization.** We study in Chapter 4 an way of taking the infinite width limit of fully connected networks in which the last layer initialization and overall structure of the learning rates differs substantially from the NTK regime. The resulting infinite width training dynamics even at fixed depth and training dataset size are significantly richer than those in the NTK regime. In one layer networks they are Wasserstein gradient flows (see §4.1.2), and their analysis is an important facet of modern deep learning theory.

3.2 Optimization and the Tangent Kernel

In this section we introduce an influential line of work explaining why optimization is successful in certain wide neural networks. The start of this section largely follows the discussion in the excellent article [LZB22].

3.2.1 What is the NTK?

Consider any parameterized family of functions

$$x \in \mathbb{R}^{N_0} \quad \mapsto \quad z(x; \theta) \in \mathbb{R},$$

which for simplicity we've assumed has output dimension 1. Suppose that we seek to optimize the parameters θ by minimizing the empirical error

$$\mathcal{L}_{\mathcal{D}}(\theta) := \frac{1}{|\mathcal{D}|} \sum_{i=1}^{|\mathcal{D}|} \ell\left(z(x_i; \theta), y_i\right)$$

using gradient flow

$$\frac{d}{dt}\theta_t = -\nabla_\theta \mathcal{L}(\theta)\big|_{\theta=\theta_t}.^{1} \tag{3.2.1}$$

[1] The analysis below will carry over with minor modifications to gradient descent with a sufficiently small learning rate.

We will assume that $z(x;\theta)$ is overparameterized relative to $\mathcal{D}$ in the sense of (1.3.3). The heuristic discussion in §1.3.1 suggests that optimization is easier to understand after changing variables the vector of outputs $z(\mathcal{D};\theta)$ of the model on the training data defined in (1.3.4). In general, given any differentiable function $f(\theta)$ for which $\mathcal{L}(\theta)$ depends on θ only through $f(\theta)$, the gradient flow dynamics (3.2.1) yields

$$\frac{d}{dt}f(\theta_t) = \left\{ \left(D_\theta f(\theta)\big|_{\theta=\theta_t} \right)^T D_\theta f(\theta)\big|_{\theta=\theta_t} \right\} \nabla_f \mathcal{L}(f)\big|_{f=f(\theta_t)}.$$

Applying this with $f(\theta) = z(\mathcal{D};\theta)$ yields

$$\frac{d}{dt}z(\mathcal{D};\theta_t) = H_{\mathcal{D}}(\theta_t)\nabla_z \mathcal{L}(z)\big|_{z=z(\mathcal{D};\theta_t)}, \tag{3.2.2}$$

where

$$H_{\mathcal{D}}(\theta) := \left(D_\theta z(\mathcal{D};\theta)\right)^T D_\theta z(\mathcal{D};\theta) \in \mathbb{R}^{|\mathcal{D}|\times|\mathcal{D}|} \tag{3.2.3}$$

is called the *tangent kernel* of z. By construction, we may write the NTK as the Gram matrix of gradients $D_\theta z(x;\theta)$ over inputs x appearing in the training dataset:

$$H_{\mathcal{D}}(\theta) = \left(H_{ij}(\theta)\right)_{1\le i,j\le|\mathcal{D}|}, \qquad H_{ij}(\theta) = \left(D_\theta z(x_i;\theta)\right)^T D_\theta z(x_j;\theta). \tag{3.2.4}$$

The equation (3.2.2) admits a simple geometrical interpretation. Namely, $H_{\mathcal{D}}(\theta)$ plays the role of the inverse of a Riemannian metric tensor on $\mathbb{R}^{|\mathcal{D}|}$, and it is with respect to this pseudo-metric that we minimize $\mathcal{L}$, viewed as a function of $z(\mathcal{D};\theta)$ using Riemannian gradient flow. In this language, the fact that we expect the map from θ to $z(\mathcal{D};\theta)$ to be a surjective submersion is equivalent to the statement that $H_{\mathcal{D}}(\theta)$ is strictly positive definite and hence that its inverse defines a proper Riemannian metric.

The preceding discussion applied to optimization of virtually any overparameterized family of functions. In the special case of the squared error (or more generally loss that is convex as a function of $z(x_i;\theta)$),

$$\mathcal{L}_{\mathcal{D}}(\theta) = \mathcal{L}_{MSE}(\theta) = \frac{1}{2\,|\mathcal{D}|}\sum_{i=1}^{|\mathcal{D}|}\left(z(x_i;\theta)-y_i\right)^2$$

the dynamics in (3.2.2) take on a particularly simple form. Namely, writing

$$Y = (y_i,\, i = 1,\ldots,|\mathcal{D}|)$$

for vector of labels, (3.2.2) immediately yields

$$\frac{d}{dt}\left(z(\mathcal{D};\theta_t) - Y\right) = H_{\mathcal{D}}(\theta_t)\left(z(\mathcal{D};\theta_t) - Y\right) \tag{3.2.5}$$

and

$$\frac{d}{dt}\mathcal{L}_{MSE}(\theta_t) = \frac{1}{|\mathcal{D}|}\left(z(\mathcal{D};\theta_t) - Y\right)^T H_{\mathcal{D}}(\theta_t)\left(z(\mathcal{D};\theta_t) - Y\right). \tag{3.2.6}$$

The preceding equations highlight the sense in which $H_{\mathcal{D}}(\theta_t)$ controls "the geometry" of optimization. They yield the following fundamental result

Lemma 3.2.1 (NTK conditioning implies exponential convergence)**.** *Suppose*

$$\exists\,\delta > 0 \quad s.t. \quad H_{\mathcal{D}}(\theta_t) \ge \delta I \qquad \textit{for all } t \ge 0. \tag{3.2.7}$$

Then

$$\mathcal{L}_{MSE}(\theta_t) \le e^{-t\delta/|\mathcal{D}|}\mathcal{L}_{MSE}(\theta_0).$$

Two remarks are in order:

- The condition (3.2.7) is precisely the statement that (1.3.4) holds for θ on the trajectory of optimization.
- Note that the condition (3.2.7) is somewhat pessimistic. It can be replaced by

$$(z(\mathcal{D};\theta_t)-Y)^T H_{\mathcal{D}}(\theta_t)\,(z(\mathcal{D};\theta_t)-Y) \geq \delta\,||z(\mathcal{D};\theta_t)-Y||^2 \qquad \text{for all } t\geq 0.$$

In the next section, we take up the question of to how check that (3.2.7) holds.

3.2.2 Optimization in the NTK Regime: heuristic derivation in shallow nets

At first glance, checking (3.2.7) is seems had to verify since it must hold throughout training. However, in the special setting of wide neural networks at fixed depth, small learning rate, and fixed dataset size the articles [DLT+18, JGH18, LZB22] show it in two basic steps:

(a) Show at the start of training that $H_{\mathcal{D}}(\theta_0)$ is strictly positive definite with high probability using matrix concentration

(b) Show that $H_{\mathcal{D}}(\theta_t)$ changes very little in the course of training.

The purpose of this section is to explain the basic idea in the simplest setting of a one layer network

$$z^{(2)}(x;\theta)=\sum_{i=1}^{N_1}\frac{1}{\sqrt{N_1}}W_i^{(2)}\phi\left(W_i^{(1)}\frac{x}{\sqrt{N_0}}\right), \qquad x,W_i^{(1)}\in\mathbb{R}^{N_0},\, W_i^{(2)}\in\mathbb{R} \tag{3.2.8}$$

trained by gradient flow (3.2.1) on

$$\theta_t=\left(W_i^{(1)}(t),W_i^{(2)}(t)\right), \qquad W_i^{(1)}(0)\sim\mathcal{N}(0,1), \quad W_i^{(2)}(0)\sim\mathcal{N}(0,I_{N_1})$$

starting from a Gaussian initialization.[2] Let us start by checking (a). A direct computation shows that

$$H_{\mathcal{D}}(\theta)=\frac{1}{N_1}\sum_{j=1}^{N_1}H_{\mathcal{D},j}(\theta), \tag{3.2.9}$$

where $H_{\mathcal{D},j}(\theta)$ is the NTK of the $j-$th neuron:

$$H_{\mathcal{D},j}(\theta)=\text{Gram}\left(D_\theta\left\{W_j^{(2)}\phi\left(W_j^{(1)}\frac{x_i}{\sqrt{N_0}}\right)\right\},\ i=1,\dots,|\mathcal{D}|\right).$$

At initialization, the $|\mathcal{D}|\times|\mathcal{D}|$ matrices $H_{\mathcal{D},j}(\theta)$ are iid and hence, by standard matrix concentration inequalities [Tro15] we find that with high probability

$$N_1\gg 1 \qquad\Longrightarrow\qquad H_{\mathcal{D}}(\theta_0)\approx\mathbb{E}\left[H_{\mathcal{D},1}(\theta_0)\right].$$

Hence, (a) follows as soon as the network width N_1 is sufficiently large and the NTK mean $\mathbb{E}\left[\Theta_1(\theta_0)\right]$ is strictly positive definite. The latter is equivalent to asking that the $|D|$ vectors $D_\theta\left\{W_j^{(2)}\phi\left(W_j^{(1)}\frac{x_i}{\sqrt{N_0}}\right)\right\}$ are linearly independent with high probability over x_i in the infinite-dimensional Gaussian Hilbert space associated with $W_j^{(1)},W_j^{(2)}$. There is of course something to

[2] Remark on NTK parameterization and learning rates?

check here, but the result is true under quite generic assumptions [DLT+18]. Next, the key point in deriving (b) is to notice that the Hessian of $z^{(2)}(x;\theta)$ is block-diagonal:

$$\mathrm{Hess}_\theta z^{(2)}(x;\theta) = \begin{pmatrix} \mathrm{Hess}_1(\theta) & 0 & \cdots & 0 \\ 0 & \mathrm{Hess}_2(\theta) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \mathrm{Hess}_{N_1}(\theta) \end{pmatrix},$$

with each block being the Hessian of $z^{(2)}(x;\theta)$ just with respect to the parameters in different neurons:

$$\mathrm{Hess}_i(\theta) = \mathrm{Hess}_{W_i^{(1)}, W_i^{(2)}} z^{(2)}(x;\theta).$$

Further, the explicit $N_1^{-1/2}$ scaling in (3.2.8) implies that, at initialization each Hessian block is small:

$$\left|\left|\mathrm{Hess}_\theta z^{(2)}(x;\theta_0)\right|\right| = O\left(\frac{\mathrm{poly}(|\mathcal{D}|)}{\sqrt{N_1}}\right) \qquad \text{with high probability.}$$

Further, some simple perturbation theory shows that with high probability

$$||\theta - \theta_0|| \text{ not too large} \qquad \Longrightarrow \qquad \left|\left|\mathrm{Hess}_\theta z^{(2)}(x;\theta)\right|\right| = O\left(\frac{\mathrm{poly}(|\mathcal{D}|)}{\sqrt{N_1}}\right) \qquad \text{with high probability.}$$

The fact that at sufficiently large width N_1, the Hessian can be made uniformly small in a neighborhood of the initialization θ_0 shows that the $D_\theta z(\mathcal{D};\theta)$ (and hence the NTK) changes very little in a large ball around initialization. In conjunction with (a) and the fact that the Jacobian $D_\theta z(\mathcal{D};\theta)$ (and hence the NTK) remains order 1, we conclude from Lemma 3.2.1 that the loss $\mathcal{L}(\theta_t)$ will start to decrease exponentially in t. This causes optimization to converge to a minimum of $\mathcal{L}$ nearby to θ_0 and never leave a sufficiently large ball around initialization.

While our analysis in this section focused on one layer networks, the fundamental mechanism underlying the asymptotic linearization of neural networks with NTK initialization and learning rates at infinite width is true and is similar at any fixed depth.

3.2.3 Mean-Field vs. NTK and the μP Heuristic

We've seen that in the NTK parameterization the infinite width limit of a fully connected network is a linear model. Our goal in this section is to give an alternative explanation for why this is and do so in a way that previews the mean-field parameterization of neural networks we study in the next chapter. We will still focus on the case of one layer networks

$$z^{(2)}(x;\theta) = \frac{1}{\gamma\sqrt{N_1}} \sum_{i=1}^{N_1} W_i^{(2)} \phi\left(z_i^{(1)}(x)\right), \qquad z_i^{(1)}(x) := \frac{1}{\sqrt{N_0}} W_i^{(1)} \cdot x \tag{3.2.10}$$

in which weights are initialized as iid standard Gaussians. Here $\gamma = 1$ corresponds to the NTK regime and, as we'll see in the next chapter, $\gamma = \sqrt{N_1}$ is the mean-field scaling. At initialization the fields $x \mapsto z_i^{(1)}(x)$ are centered Gaussian processes with

$$\mathrm{Cov}\left(z_i^{(1)}(x), z_j^{(1)}(x')\right) = \delta_{ij} \frac{x \cdot x'}{N_0}.$$

Similarly, again at initialization, the network output is order $1/\gamma$, which is either order 1 in the NTK scaling or order $1/\sqrt{N_1}$ in the mean-field parameterization. Following the basic idea in [YH21], let us now consider the *change* in the network output and in $z_i^{(1)}(x)$ as measured by the time-derivative at initialization from gradient flow

$$\frac{d}{dt}\theta_t = -\eta \nabla \mathcal{L}(\theta_t), \qquad \mathcal{L}(\theta) = \frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} \ell(z^{(2)}(x;\theta), y)$$

As we saw in (3.2.2), we have

$$\left.\frac{d}{dt}\right|_{t=0} z^{(2)}(x';\theta_t) = -\frac{\eta}{\mathcal{D}} \sum_{(x,y)\in\mathcal{D}} \Delta_{(x,y);t}\mathrm{NTK}(x,x';\theta_0), \qquad \Delta_{(x,y);\theta} = \partial_z \ell(z,y)\Big|_{z=z^{(2)}(x;\theta)}$$

where the NTK was defined in (3.2.3). A direct computation shows that

$$\mathrm{NTK}(x,x';\theta_0) = \frac{1}{\gamma^2 N_1}\sum_{i=1}^{N} \mathrm{NTK}_i(x,x';\theta_{i;0}),$$

where

$$\mathrm{NTK}_i(x,x';\theta) = \left\langle \nabla_{W_i^{(1)},W_i^{(2)}}\left(W_i^{(2)}\phi(z_i^{(1)}(x))\right), \nabla_{W_i^{(1)},W_i^{(2)}}\left(W_i^{(2)}\phi(z_i^{(1)}(x'))\right)\right\rangle.$$

Hence, at initialization, we have that $\mathrm{NTK}(x,x';\theta_0)$ is an average of N_1 iid per-neuron NTKs, which hav non-zero mean. Thus we find that

$$\left.\frac{d}{dt}\right|_{t=0} z^{(2)}(x;\theta_t) = \Theta\left(\frac{\eta}{\gamma^2}\right). \tag{3.2.11}$$

We also have

$$\frac{d}{dt} z_i^{(1)}(x;\theta_t) = -\frac{\eta}{|\mathcal{D}|}\sum_{(x,y)\in\mathcal{D}} \Delta_{(x,y);\theta_t}\frac{1}{\gamma\sqrt{N_1}} W_{i;t}^{(2)}\phi'\left(z_i^{(1)}(x;\theta_t)\right)\frac{x_\alpha \cdot x_\beta}{N_0}. \tag{3.2.12}$$

Hence, evaluating the scale of this expression as a function of N_1, γ, η at initialization yields

$$\frac{d}{dt} z_i^{(1)}(x;\theta_t) = \Theta\left(\frac{\eta}{\gamma\sqrt{N_1}}\right). \tag{3.2.13}$$

Comparing (3.2.11) and (3.2.13) shows that the only way to select η, γ which ensures that *both* the hidden layer representation $z_i^{(1)}(x)$ and the output $z^{(2)}(x)$ change by order 1 is to set

$$\gamma = \Theta(\sqrt{N_1}), \qquad \eta = \Theta(N_1),$$

which leads to the mean-field parameterization. This heuristic: that we want to find a learning rate in which all hidden layer pre-activations and network outputs move $\Theta(1)$ in each step of training is the heuristic proposed in the μP paper [YH21]. In contrast, in the NTK formulation where $\gamma = 1$, we must take $\eta = \Theta(1)$ in order to ensure that the change in the network output doesn't diverge as $N_1 \to \infty$. But this forces the change in the hidden layer activations to be $\Theta(1/\sqrt{N_1})$, much smaller than their initialization scale.

Chapter 4

Mean Field Neural Nets

In this Chapter we discuss the mean-field approach to neural networks, pioneered for one layer networks by [MMN18, CB18, RVE18, SS20] and extended to deeper networks and more complex architectures first in [BP22] and then in [BNL$^+$23, BCP24]. To introduce the main idea we begin in §4.1 by considering the fundamental setting of a one layer neural network

$$z^{(2)}(x;\theta) = \frac{1}{N_1}\sum_{i=1}^{N_1} W_i^{(2)}\phi\left(\frac{1}{\sqrt{N_0}}W_i^{(1)}x + b_i^{(1)}\right), \qquad x, W_i^{(1)} \in \mathbb{R}^{N_0}, \quad b_i^{(1)}, W_i^{(2)} \in \mathbb{R}, \tag{4.0.1}$$

where unlike in the NTK parameterization of Chapter 3 we've introduced a seemingly innocuous $1/N_1$-normalization in the output instead of the $1/\sqrt{N_1}$ prescription that leads to the kernel regime in the limit $N_1 \to \infty$[1]. A fundamental observation going to at least to [Cyb89] is that to each one layer network (4.0.1) we can associate a probability measure

$$z^{(2)}(x;\theta) = \int_{\mathbb{R}^{N_0+2}} W^{(2)}\phi(W^{(1)}x + b^{(1)})d\mu_\theta(W^{(1)}, b^{(1)}, W^{(2)}), \qquad \mu_\theta = \frac{1}{N}\sum_{i=1}^{N}\delta_{(W_i^{(1)}, b_i^{(1)}, W_i^{(2)})}. \tag{4.0.2}$$

A key advantage of working directly with μ_θ is that networks of different widths can all be analyzed in the same space, namely the space of Borel probability measures on $\mathbb{R}^{N_0+2}$. Moreover, passing to μ_θ naturally quotients by the action of the symmetric group S_N permuting neuronal indices $i = 1, \ldots, N_0$. The remainder of this chapter is organized as follows:

- In §4.1 we review how the mapping to empirical measures (4.0.2) has been used to understand approximation theory [Cyb89, Bar92] (see §4.1) and gradient flow optimization with shallow networks [MMN18, CB18, RVE18, SS20] (see §4.1.2).
- In §4.2 we outline an important extension of the mean-field analysis of one layer networks that is valid for studying the infinite width limit of virtually any neural network architecture using tools from *dynamical mean-field theory* (DMFT) [BP22]. This point of view, developed in large far at a physics level of rigor, leads to many important some open problems that we will briefly outline.

4.1 Mean-Field One Hidden Layer Networks

In this section, we explain how question about the approximation power and optimization dynamics of one hidden layer neural networks look when phrased in terms of the empirical measures μ_θ of (4.0.2).

[1] We will sometimes omit the $1/\sqrt{N_0}$ pre-fact in front of the first layer weights for convenience.

4.1.1 Mean-Field Approximation

An important line of work in the mathematical neural network literature in late 1980's and early/mid 1990's focused on neural network *approximation theory*: which functions can be represented and how efficiently is the approximation accomplished per-parameter? One of the foundational results in this direction is the so-called *universal approximation theorem* [Cyb89, Pin99].

Theorem 4.1.1 (Universal Approximation Theorem, roughly)**.** *Fix any $N_0 \geq 1$ and a "nice" non-polynomial $\phi : \mathbb{R} \to \mathbb{R}$. Then for any compact set $\Omega \subseteq \mathbb{R}^{N_0}$, continuous function $f : \Omega \to \mathbb{R}$, and $\epsilon > 0$ there exists $N_1 \geq 1$ and parameters $\theta \in \mathbb{R}^{N_1(N_0+2)}$ so that*

$$\left|\left|f(x) - z^{(2)}(x;\theta)\right|\right|_{C^0(\Omega)} \leq \epsilon,$$

where $z^{(2)}(x;\theta)$ is as in (4.0.1)*.*

One natural approach to Theorem 4.1.1 is to directly use the representation (4.0.2). Since networks with width N_1 correspond to measures μ_θ with N_1 atoms, modulo a few technical details, infinitely wide one layer neural network can compute any function of the form

$$f_\mu(x) = \int_{\mathbb{R}^{N_0+2}} W^{(2)}\phi(W^{(1)}x + b^{(1)})d\mu(b^{(1)}, W^{(1)}, W^{(2)}), \tag{4.1.1}$$

where μ is any "reasonable" probability measure. From this perspective the universal approximation is the statement that for "most" non-linearities ϕ, any continuous function can be represented in the form f_μ for some μ. If $\phi(t) = \cos(t)$, then this is a simple exercise in Fourier analysis. For more general activations more intricate arguments are required [Pin99].

An important continuation of this idea was devised by Andrew Barron in the influential papers [Bar93, Bar94]. He was interested in understanding the dependence between the approximation rate ϵ and network width N_1 in Theorem 4.1.1. For any measure μ he considered an empirical approximation

$$\mu \approx \mu_{N_1}, \qquad \mu_N := \frac{1}{N_1}\sum_{i=1}^{N_1}\delta_{X_i}, \qquad X_i \overset{\text{iid}}{\sim} \mu,$$

which gives an approximation of any f_μ from (4.1.1) by a width N_1 neural network $f_{\mu_{N_1}}$. A key contribution of Barron's work was to introduce the so-called Barron space,

$$\mathcal{B}(\mathbb{R}^{N_0}) := \left\{f : \mathbb{R}^{N_0} \to \mathbb{R} \mid f \in C^0(\mathbb{R}^{N_0}) \quad \text{and} \quad ||f||_{\mathcal{B}} < \infty\right\}, \qquad ||f||_{\mathcal{B}} = \int_{\mathbb{R}^{N_0}} ||\xi|| \left|\left|\widehat{f}(\xi)\right|\right| d\xi,$$

where $\widehat{f}$ is the Fourier transform. A short computation shows that for some natural non-linearities ϕ the Fourier decay of f allows one to obtain a central limit theorem for

$$\int_{\mathbb{R}^{N_0+2}} W^{(2)}\phi(W^{(1)}x + b^{(1)})d\left[\sqrt{N_1}\left(\mu - \mu_{N_1}\right)\right]$$

uniformly over μ for which $||f_\mu||_{\mathcal{B}} \leq 1$.[2] The Barron space is thus a surprisingly simple class of functions for which the width of a one layer network required to uniformly approximate functions from this class is independent of the input dimension, decaying like $1/\sqrt{\text{width}}$. This is an example of how coupling smoothness to input dimension allows one to break the *curse of dimensionality.*

4.1.2 Mean-Field Optimization

In the previous section, we saw how the parameter-to-measure correspondence (4.0.2) can be used to understand which functions a one layer network can represent when no constraints are placed on its parameters. As we discussed in Chapter 1, however, neural network parameters θ are

[2] We are glossing over the distinction between spectral and neural network-based Barron spaces.

set in practice by using some variant of gradient descent to minimize an empirical risk starting from a random initialization. Our purpose in this section is to connect the resulting dynamics on empirical measures to mean-field interacting particles systems and more precisely Wasserstein gradient flows. To do this, suppose that as in (4.0.1) and (4.0.2) we are given a one hidden layer network (with no biases for simplicity)

$$z^{(2)}(x;\theta) = \frac{1}{N_1}\sum_{i=1}^{N_1} W_i^{(2)}\phi(W_i^{(1)}x) = \int_{\mathbb{R}^{N_0+1}} W^{(2)}\phi(W^{(1)}x)d\mu_\theta(W^{(1)},W^{(2)})$$

and we've introduced as before

$$\mu_\theta = \frac{1}{N_1}\sum_{i=1}^{N_1}\delta_{(W_i^{(1)},W_i^{(2)})}.$$

Suppose for simplicity that the network parameters are optimized by gradient flow[3]

$$\frac{d}{dt}\theta_t = -N_1\nabla_\theta\mathcal{L}(\theta)$$

on the empirical risk

$$\mathcal{L}(\theta_t) = \frac{1}{2\left|\mathcal{D}\right|}\sum_{(x,y)\in\mathcal{D}}\left(z^{(2)}(x;\theta_t)-y\right)^2.$$

Note that the network output is a linear functional of μ_θ:

$$z^{(2)}(x;\theta) = \langle\psi_x,\mu_\theta\rangle, \qquad \psi_x(W^{(1)},W^{(2)}) := W^{(2)}\phi(W^{(1)}x)$$

Since the loss $\mathcal{L}$ depends on θ only through μ_θ the dynamics on θ_t therefore yield well-defined dynamics on $\mu_t := \mu_{\theta_t}$. This evolution can be equivalently described as a Wasserstein gradient flow or as a mean-field interacting particle system. Specifically, let us write

$$\theta_t = (\theta_{i;t},\, i=1,\dots,N_1), \qquad \theta_{i;t} := \left(W_{i;t}^{(1)}, W_{i;t}^{(2)}\right).$$

We will refer to $\theta_{i;t}$ as the *state of neuron* i at time t. The key observation is that

$$\frac{d}{dt}\theta_{i;t} = -\nabla_{\theta_i}\frac{\partial\mathcal{L}}{\partial\mu}(\mu_t;\theta_i), \tag{4.1.2}$$

where $\partial\mathcal{L}/\partial\mu$ is the Fréchet derivative

$$\frac{\partial\mathcal{L}}{\partial\mu}(\mu;\theta) = \frac{1}{\left|\mathcal{D}\right|}\sum_{(x,y)\in\mathcal{D}}(\langle\mu,\psi_x\rangle - y)\psi_x(\theta).$$

There are two equivalent ways to parse equation (4.1.2):

- **Mean-field interacting particles.** By inspection, the evolution of the i-th neuron state depends on the other neuron states only through their empirical measure. Thus, the neuron states $\theta_{i;t}$ form a *mean-field* interacting particle system, a well-studied class of non-linear dynamical systems [Szn91, Kac56, Mél96]. An important caveat is that often interacting particle systems studied in the literature include an additional Gaussian noise as a driving term in (4.1.2) but the only randomness present in (4.1.2) is through the random initial condition.

[3]While the N_1 pre-factor is only a time reparameterization from the perspective of gradient flow, it is the natural rescaling for a well-defined $N_1\to\infty$ limit and is critical when studying gradient descent with small but positive steps sizes.

- **Wasserstein gradient flow.** By pairing with a test function and integrating by parts the relation (4.1.2) can be written as a continuity equation

$$\partial \mu_t = \operatorname{div}\left(\mu_t \nabla \frac{\partial \mathcal{L}}{\partial \mu}(\mu_t)\right),$$

  which is precisely the Wasserstein-2 gradient flow on the energy $\mathcal{L}$ [CNWR25]. While this identification with Wasserstein gradient flows holds only for one layer networks, it is a powerful tool in understanding their training dynamics [CB18, CB20].

An important corollary of the interacting particle interpretation of (4.1.2) is that the distribution of particle states $\{\theta_{i;t}\}$ in one one-layer networks exhibits *propagation of chaos* in the infinite width limit [Szn91, Lac23]. That is, for all $t > 0$ and fixed finite $k \geq 1$ the random variables $\{\theta_{i;t},\, 1 \leq i \leq k\}$ are iid. In particular, in the infinite width limit the finite-dimensional pre-activation stochastic processes

$$x \in \mathbb{R}^{N_0} \mapsto W_{i;t}^{(1)} x \in \mathbb{R}$$

are iid over i. In the next section, we briefly sketch an important generalization of this interacting particle point of view on neural network training dynamics that holds for virtually any neural network architecture and still exhibits this propagation of chaos structure. This generalization has been developed only a physics level of rigor and making it mathematically precise presents and interesting and important challenge in deep learning theory.

4.2 Neural Network Dynamical Mean-Field Theory

Extend the one layer parameters-to-measures correspondence from the previous section to deeper networks and beyond fully connected architectures is non-trivial. A straightforward generalization in which one defines the state of a neuron to be the collection weights going in / out faces two challenges:

- **Growing state dimension.** As soon as there is more than one hidden layer, the number of weights going in / out of each neuron grows with network width. This means the analog of the particle dynamics (4.1.2) are no longer defined on a single space independent of width.

- **Shared states.** As soon as there is more than one hidden layer, the weights going in and/or out of each neuron are shared with other neurons. This requires analyzing the analog of the particle dynamics (4.1.2) with weakly correlated neuron states.

The most well-developed approach to overcoming these two technical challenges was pioneered by Bordelon and Pehlevan in the breakthrough article [BP22]. This work uses *dynamical mean-field theory* to provide, at a physics level of rigor, a blueprint for analyzing the training dynamics of virtually any neural network architectures. Our goal in this section is to give a short introduction to the basic idea. For this let us consider a depth L fully connected network. Recall from Definition 1.1.1 that such a model maps an input $x \in \mathbb{R}^{N_0}$ to an output $z^{(L+1)}(x;\theta) \in \mathbb{R}^{N_{L+1}}$ recursively as follows

$$z^{(\ell)}(x) = \begin{cases} \frac{1}{N_L} W^{(L)} \phi(z^{(L)}(x)), & \ell = L+1 \\ \frac{1}{\sqrt{N_\ell}} W^{(\ell)} \phi(z^{(\ell-1)}(x)) \in \mathbb{R}^{N_{\ell+1}}, & 1 \leq \ell \leq L \\ \frac{1}{\sqrt{N_0}} W^{(1)} x \in \mathbb{R}^{N_1}, & \ell = 1. \end{cases}$$

For simplicity we have omitted the biases and will focus on the case of output dimension $N_{L+1} = 1$. Consider now a weight $W_{ij}^{(\ell+1)}$ connecting neuron j in layer ℓ to neuron i in layer $\ell+1$. For any training datapoint (x, y) we may use the chain rule to write the following identity for the derivative

of the sample loss $\ell(z^{(L+1)}(x;\theta), y)$:

$$\begin{aligned}\partial_{W_{ij}^{(\ell+1)}}\ell(z^{(L+1)}(x;\theta),y) &= \frac{\phi\left(z_i^{(\ell+1)}(x)\right)}{\partial W_{ij}^{(\ell+1)}}\cdot\frac{\partial z^{(L+1)}(x)}{\partial\phi\left(z_i^{(\ell+1)}(x)\right)}\cdot\left.\frac{\partial\ell(z,y)}{\partial z}\right|_{z=z^{(L+1)}(x)}\\ &= \frac{1}{\sqrt{N_\ell}}\phi(z_j^{(\ell)}(x))\cdot\phi'\left(z_i^{(\ell+1)}(x)\right)\cdot\frac{\partial z^{(L+1)}(x)}{\partial\phi\left(z_i^{(\ell+1)}(x)\right)}\cdot\left.\frac{\partial\ell(z,y)}{\partial z}\right|_{z=z^{(L+1)}(x)}.\end{aligned}$$

The neuron pre-activations (first and third terms) and *partial* Jacobians (derivatives of network output with respect to a hidden layer output) therefore determine the structure of loss gradients. This suggests defining

$$S_{i;t}^{(\ell)} = \text{ state of neuron } i \text{ in layer } \ell \text{ at time } t = \left\{ z_i^{(\ell)}(x),\ \frac{\partial z^{(L+1)}(x)}{\partial\phi\left(z_i^{(\ell)}(x)\right)},\ (x,y)\in\mathcal{D}\right\}. \tag{4.2.1}$$

The core insight of [BP22] is twofold. First, gradient flow training dynamics

$$\frac{d}{dt}\theta_t = -\nabla\mathcal{L}(\theta_t)$$

can be rewritten at any finite width as a multi-species mean-field interacting particle system in the sense that

$$\frac{d}{dt}S_{i;t}^{(\ell)} = \mathcal{F}(S_{i;t}^{(\ell)}, \mu_t^{(\ell')},\ \ell'=1,\dots,L), \qquad \mu_t^{(\ell)} := \frac{1}{N_\ell}\sum_{i=1}^{N_\ell}\delta_{S_{i;t}^{(\ell)}}$$

for some explicit $\mathcal{F}$. Second, when $N_1,\dots,N_L\to\infty$, a saddle point argument shows that the evolution of $\{\mu_t^{(\ell)},\ \ell=1,\dots,L\}$ obeys a complicated but deterministic coupled set of PDEs. Proving well-posedness of these limiting equations and making precise this saddle point analysis of [BP22] are important mathematical open problems. Because the saddle point argument of [BP22] is so elegant and extends naturally to virtually any neural architecture (see e.g. [BNL$^+$23, BCP24, JBPH26]), we will present in §4.2.1 the argument in the simplest case of one layer networks. Before doing so, we first briefly discuss in §3.2.3 a unified derivation of the mean-field and NTK parameterizations.

4.2.1 Sketch of DMFT Analysis in One Layer Networks

We now return to the setting of a one layer mean-field network (3.2.10) with $\gamma = \sqrt{N_1}$ and $\eta = \eta_0 N_1$ and restrict our attention to mean-squared error loss. With this convention (3.2.12) becomes

$$\frac{d}{dt}z_i^{(1)}(x';\theta_t) = -\frac{\eta_0}{|\mathcal{D}|}\sum_{(x,y)\in\mathcal{D}}\Delta_{(x,y);\theta_t}W_{i;t}^{(2)}\phi'\left(z_i^{(1)}(x;\theta_t)\right)\frac{x\cdot x'}{N_0}.$$

Note that $\partial z^{(2)}/\partial\phi(z_i^{(1)}(x)) = N^{-1}W_i^{(2)}$, and we have

$$\frac{d}{dt}W_{i;t}^{(2)} = -\frac{\eta}{|\mathcal{D}|}\sum_{(x,y)\in\mathcal{D}}\Delta_{(x,y);\theta_t}\phi\left(z_i^{(1)}(x;\theta_t)\right). \tag{4.2.2}$$

The dynamics of the state $S_{i;t}^{(1)}$ of neuron i depend on the other neurons only through the residual

$$\Delta_{(x,y);\theta_t} = z^{(2)}(x;\theta_t) - y = \frac{1}{N}\sum_{i=1}^{N}\left\{W_{i;t}^{(2)}\phi\left(z_{i;t}^{(1)}(x)\right) - y\right\},$$

which is an empirical average at time the neurons. Moreover, from the definition of the NTK (see e.g. (3.2.2)) we have

$$\begin{aligned}\frac{d}{dt}\Delta_{(x',y');\theta_t} &= -\frac{\eta}{|\mathcal{D}|}\sum_{(x,y)\in\mathcal{D}}\Delta_{(x,y);\theta_t}\mathrm{NTK}\,(x,x';\theta_t)\\ &= -\frac{\eta_0}{|\mathcal{D}|}\sum_{(x,y)\in\mathcal{D}}\Delta_{(x,y);\theta_t}\left(\Phi(x,x';\theta_t)+G(x,x';\theta_t)\frac{x\cdot x'}{N_0}\right),\end{aligned}\tag{4.2.3}$$

where

$$G(x,x';\theta_t)=\frac{1}{N}\sum_{i=1}^{N}g_{i;t}(x)g_{i;t}(x'),\quad g_{i;t}(x)=W^{(2)}_{i;t}\phi'\left(z^{(1)}_{i;t}(x)\right)$$

$$\Phi(x,x';\theta_t)=\frac{1}{N}\sum_{i=1}^{N}\phi\left(z^{(1)}_{i}(x)\right)\phi\left(z^{(1)}_{i}(x')\right)$$

We will abbreviate

$$G_t:=\left(G(x,x';\theta_t)\right)_{(x,y),(x',y')\in\mathcal{D}},\qquad \Phi_t:=\left(\Phi(x,x';\theta_t)\right)_{(x,y),(x',y')\in\mathcal{D}}.$$

If these kernels were deterministic (they are not at finite N), then the dynamics of $(z^{(1)}_{i;t}(x),W_{i;t})$ would be iid across i since these kernels determine the dynamics of $\Delta_{(x,y);\theta_t}$, and it is only these residuals that coupled the dynamics across neurons. To make sure of this, consider the characteristic function

$$Z[\chi]:=\mathbb{E}\left[\exp\left\{-i\int_0^t\sum_{(x,y)\in\mathcal{D}}\chi_{x,s}\Delta_{(x,y);\theta_s}ds\right\}\right],\tag{4.2.4}$$

which is a generating function for the joint moments of residuals over time and training datapoints[4]. We now use a standard trick in statistical physics, often called the Hubbard-Stantonovich method, which uses the Fourier representation of a delta function

$$\delta_0(x-x')=(2\pi)^{-d}\int_{\mathbb{R}^d}e^{-i(x-x')\cdot\xi}d\xi,\qquad x,x'\in\mathbb{R}^d,$$

to write for any $f:\mathbb{R}^d\to\mathbb{R}$

$$\int_{\mathbb{R}^d}f(x)dx=\int_{\mathbb{R}^{2d}}f(x')\delta_0(x-x')dxdx'=(2\pi)^{-d}\int_{\mathbb{R}^{3d}}f(x')e^{-i(x-x')\cdot\xi}dxdx'd\xi.$$

The point of this trick is to turn x, which may have a complicated definition, into three integration variables x,x',ξ. We will specifically use

$$\begin{aligned}&\delta_0\left(NG(x,x';\theta_s)-\sum_{j=1}^{N}g_{i;s}(x)g_{j;s}(x')\right)\\ &\quad\propto\int_{\mathbb{R}^2}\exp\left\{-i\widehat{G}(x,x';\theta_s)\left(NG(x,x';\theta_s)-\sum_{j=1}^{N}g_{i;s}(x)g_{j;s}(x')\right)\right\}dG(x,x';\theta_s)d\widehat{G}(x,x';\theta_s)\end{aligned}$$

and its analogs for the Φ kernel. Inserting these two delta functions into (4.2.4) and using that conditional on $G,\widehat{G},\Phi,\widehat{\Phi}$ neurons are independent gives

$$\begin{aligned}Z[\chi]\propto\mathbb{E}\Bigg[\int_{\mathbb{R}^{4|\mathcal{D}|^2}}&\prod_{(x,y),(x',y')\in\mathcal{D}}dG(x,x';\theta_s)d\widehat{G}(x,x';\theta_s)d\Phi(x,x';\theta_s)d\widehat{\Phi}(x,x';\theta_s)\\ &\times\exp\left\{-i\int_0^t\sum_{(x,y)\in\mathcal{D}}\chi_{x,s}\Delta_{(x,y);\theta_s}ds-NS\left[\left\{G_s,\widehat{G}_s,\Phi_s,\widehat{\Phi_s},\,0\le s\le t\right\}\right]\right\}\Bigg],\end{aligned}$$

[4]This analysis can also be done in discrete time as in Appendix M in [BP22]

where

$$S = -i \int_0^t \left[\log Z_{\text{neuron}}[\chi, G, \widehat{G}, \Phi, \widehat{\Phi}] + \operatorname{Tr}\left(G_s \widehat{G}_s\right) - \operatorname{Tr}\left(\Phi_s \widehat{\Phi}_s\right) \right] ds,$$

and single-neuron characteristic function Z_{neuron} satisfies

$$Z_{\text{neuron}} = \mathbb{E}\Bigg[\exp\Bigg\{ -i \sum_{\substack{(x,y)\in\mathcal{D} \\ (x',y')\in\mathcal{D}}} \int_0^t \Big(\widehat{\Phi}(x,x';\theta_s)\phi\left(z_{i;s}^{(1)}(x)\right)\phi\left(z_{i;s}^{(1)}(x)\right) + \widehat{G}(x,x';\theta_s) g_{i;s}(x) g_{i;s}(x') \Big) ds \Bigg\} \Bigg]$$

with an average only over $W_{i;0}^{(1)}, W_{i;0}^{(2)}$. We are now in a position to use the Laplace method to understand the large N behavior of the integral defining $Z[\chi]$ since the integral is finite-dimensional (we keep the number of datapoints fixed). A direct computation shows that the critical point equations for G, Φ take the form

$$\begin{aligned} \widehat{G}_s &= \int_0^s \left(a_\tau(G, \widehat{G}, \Phi, \widehat{\Phi}) \widehat{G}_\tau + b_\tau(G, \widehat{G}, \Phi, \widehat{\Phi}) \widehat{G}_\tau \widehat{\Phi}_\tau \right) d\tau \\ \widehat{\Phi}_s &= \int_0^s \left(c_\tau(G, \widehat{G}, \Phi, \widehat{\Phi}) \widehat{G}_\tau + d_\tau(G, \widehat{G}, \Phi, \widehat{\Phi}) \widehat{G}_\tau \widehat{\Phi}_\tau \right) d\tau. \end{aligned}$$

An explicit analysis of is needed to the only critical points are $\widehat{\Phi} = \widehat{G} = 0$ (see e.g. [BP24]). The critical point equations for G, Φ then read

$$G(x,x';\theta_s) = \mathbb{E}_s\left[g_{i;s}(x) g_{i;s}(x') \right], \qquad \Phi(x,x';\theta_s) = \mathbb{E}_s\left[\phi\left(z_{i;s}^{(1)}(x)\right) \phi\left(z_{i;s}^{(1)}(x')\right) \right]. \tag{4.2.5}$$

In conclusion, at large N the kernels G_s, Φ_s satisfy a *deterministic* evolution given by the saddle point equations. This evolution determines a deterministic evolution of single-neuron probability distributions via $Z_{\text{neuron}}[\chi]$. They are related by self-consistent equations (4.2.5). In particular, neurons are independent, giving a physics-style proof of propagation of chaos.

4.2.2 Open Problems

The mean-field analysis of neural networks is an active field of research with many still-open problems. We state a few of them here. An important strength of the DMFT approach to the analysis of neural network training dynamics is that it applies to architectures with an arbitrary number of layers. However, at least the approach presented here is limited fundamentally the setting of learning from a fixed training dataset. This is unsatisfactory since feature learning is most interesting in the regime where model size and dataset size grow together.

Open Problem 5. *Consider an L hidden layer fully connected mean-field neural network $x \mapsto z^{(L+1)}(x)$ with input dimension N_0, output dimension $N_{L+1} = 1$, hidden layer widths $N_1, \ldots, N_L \simeq N \gg 1$ and activation function ϕ. Fix a probability distribution ρ on $\mathbb{R}^{N_0+1}$, and suppose we train this model using gradient descent to minimize the mean-squared error over a training dataset*

$$\mathcal{D} = \{(x_\mu, y_\mu)\, \mu = 1, \ldots, n\} \sim \rho^{\otimes n}$$

consisting of n iid draws from ρ. Extend the mean-field/DMFT analysis of neural network training to the setting where n, N jointly tend to infinity.

Remark 4.2.1. *In the very special case of deep linear networks the article [BP25] solved this problem.*

The neural network-centric motivation to study this problem is self-evident. As we saw in the present Chapter the mean-field analysis of neural networks can be viewed as the analysis of mean-field interacting particle systems. Each network layer corresponds to a different species of particle. This make the preceding open problem also an interesting from the pure math point of view in two senses:

- In the setting of one layer networks we saw in §4.1.2 optimization corresponds to Wasserstein gradient flow. The extension to deeper networks will therefore generalize Wasserstein gradient flows to some kind of coupled family of PDEs over tuples of probability measures. These PDEs may well have an interesting geometric and analytic structure.
- The DMFT approach to analyzing neural networks essentially defined each particle state to be a vector whose dimension is linear in the number of training datapoints (see (4.2.1)). Thus, extending DMFT to the setting of simultaneously growing dataset and model size means studying interacting particle systems in which the number of particles and the dimension of the state of each particle grow together. This seems like a fundamental problem.

The next open problem we discuss concerns extending the DMFT analysis to completely arbitrary architectures. The state it, note that the most general definition of a (feed-forward) neural network is as a directed acyclic graph (DAG) $G = (V, E)$. A subset of the vertices are the inputs, a subset of the vertices are the outputs, and any other vertex v can be thought of a hidden layer of some width N_v and some non-linearity. In this formalism, one thinks of the trainable weights as living on the edge of G.

Open Problem 6. *Extend the DMFT analysis of neural network training to a general DAG in the regime of fixed dataset size and growing hidden layer widths.*

Remark 4.2.2. *Articles such as [ZLB22] study learning in DAGs with the NTK parameterization.*

Chapter 5

Empirical Wonders

One of the joys of deep learning theory is prospect of understanding and predicting the plethora of fascinating and poorly understood empirical phenomena in deep learning. In this chapter we discuss several of them:

- In §5.1 we discuss *neural scaling laws*, empirical power law relationships between scale (e.g. number of parameters, number of datapoints, number of FLOPs used in training) and neural network performance.
- In §5.2 we discuss *grokking and emergence*, empirically observed abrupt leaps in neural network capacity.
- In §5.3 we discuss *superposition*, empirically observed sparse linear structure in learned representation $x \mapsto z^{(\ell)}(x)$ from hidden layers in large pre-trained models.
- In §5.4 we discuss the ubiquitous appearance of *adversarial examples*, surprisingly small perturbations of correctly classified inputs that a trained model classifies confidently and incorrectly.

5.1 Scaling Laws

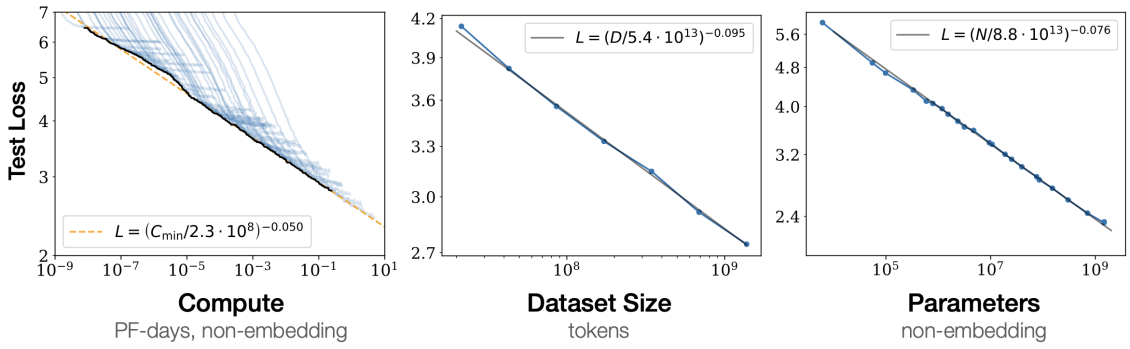

Figure 5.1: Scaling laws for test loss as reported in [KMH+20].

An important part of the success of modern deep learning is improvement through a coordinated increase in scale of models, training data, and overall compute budgets. To visualize this progress and to effectively scale computational resources practitioners often rely on scaling laws, empirically observed power law relationships between scale and model quality (see Figure 5.1). Some of the first scaling laws were discovered in [HNA+17] and [KMH+20]. Influential followup work can be

found in [HBM+22]. Prior theoretical work on theory for scaling laws can be divided roughly into three kinds

- **Weyl Law-type Arguments.** Even for linear models seemingly exotic 1/effective-dimension power law scaling exponents naturally appear in at least two ways: as the asymptotic decay of kernel eigenvalues from the Weyl Law and as typical near-neighbor spacings between training datapoint inputs. Such ideas were extended and made precise in [BDK+21] to propose a taxonomy of variance-limited vs resolution-limited scaling for both model size and dataset size. This work was preceded by two important statistical physics articles [BCP20] and [CBP21].
- **Linear Models in High-Dimensions.** The articles [PPAP25] and [PPXP24] use random matrix theory and homogenization to analyze train and test loss dynamics for SGD on random quadratic objective in high dimensions. This results in a surprisingly rich phase diagram and associated power-law exponents for learning in high dimensions. Other important contributions of this kind are [LWK+24, MM19, DLM24, MS24, AZVP24].
- **Solvable Statistical Mechanics Models.** The excellent paper [MRS22] and the followup [Zha25] provide solvable models for explaining how dataset structure and feature statistics interact to produce scaling laws in random feature models. The articles [BAP24, BCP20, BAP25] use dynamical mean-field theory to analyze scaling laws in both linear and certain non-linear models.

The seemingly ubiquitous nature of scaling laws raises several fundamental theory questions:

- **Q1.** Can one explain and predict in terms of data and model architecture the seemingly exotic exponents appearing in neural network scaling laws?
- **Q2.** Can one predict *compute-optimal* neural network scaling? That is, given a total FLOP budget for training, can one predict the optimal combination of model size (e.g. depth, width), dataset size (e.g. number of tokens per parameter), and total number of training iterations to achieve best performance?

5.2 Grokking and Emergence

Grokking [PBE+22] and emergence [WTB+22] both refer to abrupt change in model performance. For grokking, this happens is during training, where test accuracy suddenly transitions from near random chance to 100% long after training accuracy reaches close to 100% (see Figure 5.2). The somewhat puzzling phenomenon has been observed in a variety of settings [PBE+22, LMT23, LKN+22, NCL+23, Gro23]. Intuitively, it occurs when there are many ways to fit almost perfect fit the training data but optimization is biased to move, albeit slowly, along this manifold of empirical loss minima towards a particularly parsimonious solution. Notable theoretical approaches to this problem include [KBGP24, LJL+24, LBBS24, RSR24, ŽI24, DHDG24] but a general theory of grokking is still open. In particular, important theory questions include

- **Q1.** In the special case of learning modular operations, how much training data is necessary for the model to grok and the precise algorithm the model learns to implement?
- **Q2.** For which combinations of data generating process, model, and optimizer will grokking occur?

Unlike grokking, emergence in neural networks occurs when models suddenly exhibit the ability to solve a given task after a certain model scale (often measured by the total number of FLOPs used during training). This is illustrated in Figure 5.3. The term emergence was coined in [WTB+22] and important empirical followups include [C+23, G+22, S+23, WWS+22]. An important caveat is the objection made popular in [SMK23, LBS+24, DZDT24], which argued that the apparent

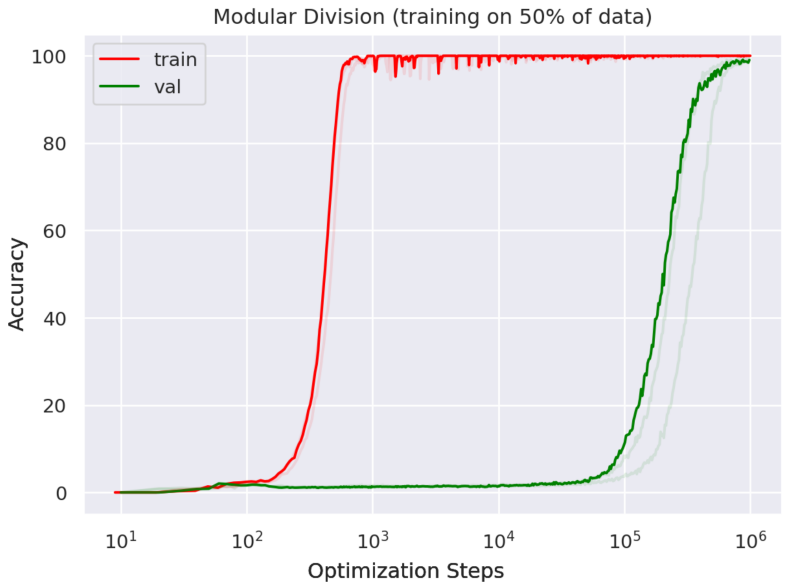

Figure 5.2: Grokking in modular division from [PBE+22]

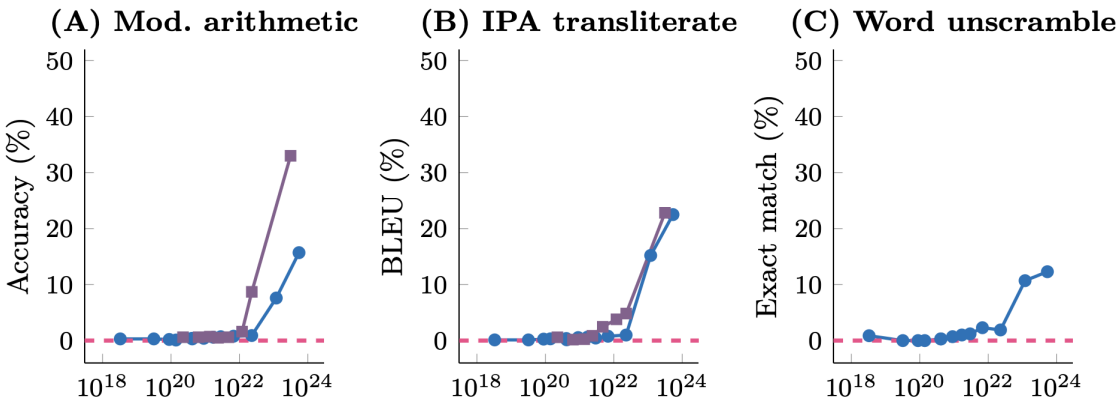

Figure 5.3: Illustration of emergence in LLMs from [WTB+22]. x-axis is training FLOPs.

phenomenon of emergence is a mirage in the sense that by replacing metrics such as accuracy by *smoother* variant the apparently sudden emergence of various capabilities in LLMs in fact occurs gradually. From the deep learning theory perspective, a precise characterization, and even definition, of emergence is still open.

5.3 Superposition

Across a variety of domains and architectures *directions in post-activation space* correspond to interpretable abstractions [EHO+22, MCCD13, HCH+23, BTB+23, SSJ+22][1]. This phenomenon was termed superposition in [EHO+22]. Perhaps the most famous example came from the influential early Word2Vec work [MCCD13], which observed that in latent space one often observes

[1]Post-activation space is the vector space in which the vectors $\phi(z^{(\ell)}(x))$ of layer ℓ post-activations live for various network inputs x

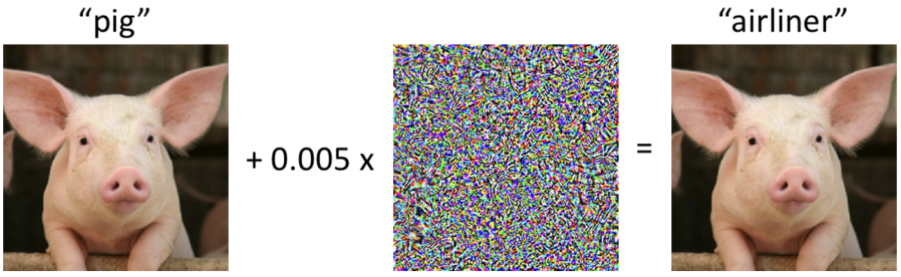

Figure 5.4: An adversarial example from [Pau20].

relationships such as

$$\mathrm{V}(\text{“king”}) - \mathrm{V}(\text{“queen”}) \approx \mathrm{V}(\text{“man”}) - \mathrm{V}(\text{“woman”})$$

where V(“word”) is the latent representation of a given word. Many times a direction in post-activation space that corresponds to an interpretable concept is sparse in the neuron basis (i.e. the basis in which we apply the non-linearity component-wise). As pointed out in [EHO+22], the number of independent concepts represented in a given layer often far exceeds the width of this layer, meaning that sparsity is required for linear decodability. There is relatively little theory to explain superposition (see [EHO+22, CDI24]), leading to several important questions:

- **Q1.** For which data generating processes and neural networks can one prove that superposition superposition will occur?
- **Q2.** What is the relationship between superposition and compressed sensing (both are about simultaneously decoding many sparse vectors in high dimensions)?

5.4 Adversarial Examples

An adversarial example is small perturbation of an input to a neural network that radically changes the model prediction (see Figure 5.4). The presence of such perturbations is ubiquitous across model architecture, training datasets, and optimization procedure [SZS+14, GSS15, MDFF16, CW17]. Despite much practical interest and effort it is unclear whether one can avoid adversarial examples in trained models [MMS+18, ZYJ+19, CH20, CRK19]. This leads to a number of theory questions:

- **Q1.** Is it possible to devise a training procedure that avoids adversarial examples but can still give test performance? Much effort has been devoted to this question but every purported defense against adversarial examples has been broken or leads to strong degradation in *clean* test performance.
- **Q2.** Are adversarial examples a necessary consequence of efficient learning in high dimensions?
- **Q3.** Is there a natural metric in which to measure smallest of a perturbation in which it would be clear whether adversarial examples either must or cannot occur in a ball of a fixed radius?

An important contribution to the theory of adversarial examples is the work of Bubeck and Sellke on the so-called universal law of robustness [BBC21]. See also [BBC21] for an analysis of adversarial examples in randomly initialized ReLU networks.

Bibliography

[AAM22] Emmanuel Abbe, Enric Boix Adsera, and Theodor Misiakiewicz. The merged-staircase property: a necessary and nearly sufficient condition for sgd learning of sparse functions on two-layer neural networks. In *Conference on Learning Theory*, pages 4782–4887. PMLR, 2022.

[AAM23] Emmanuel Abbe, Enric Boix Adsera, and Theodor Misiakiewicz. Sgd learning on neural networks: leap complexity and saddle-to-saddle dynamics. In *The Thirty Sixth Annual Conference on Learning Theory*, pages 2552–2623. PMLR, 2023.

[AB12] Gernot Akemann and Zdzislaw Burda. Universal microscopic correlation functions for products of independent ginibre matrices. *Journal of Physics A: Mathematical and Theoretical*, 45(46):465201, 2012.

[ABA13] Antonio Auffinger and Gerard Ben Arous. Complexity of random smooth functions on the high-dimensional sphere. 2013.

[ABK20] Gernot Akemann, Zdzislaw Burda, and Mario Kieburg. Universality of local spectral statistics of products of random matrices. *Physical Review E*, 102(5):052134, 2020.

[ACGH19] Sanjeev Arora, Nadav Cohen, Noah Golowich, and Wei Hu. A convergence analysis of gradient descent for deep linear neural networks. *ICLR*, 2019.

[ACH18] Sanjeev Arora, Nadav Cohen, and Elad Hazan. On the optimization of deep networks: Implicit acceleration by overparameterization. *ICML*, 2018.

[AG97] G Ben Arous and Alice Guionnet. Large deviations for wigner's law and voiculescu's non-commutative entropy. *Probability theory and related fields*, 108(4):517–542, 1997.

[AGZ10] Greg W Anderson, Alice Guionnet, and Ofer Zeitouni. *An introduction to random matrices*. Number 118. Cambridge university press, 2010.

[Ahn22] Andrew Ahn. Fluctuations of beta-jacobi product processes. *Probability Theory and Related Fields*, 183(1):57–123, 2022.

[APP+22] S Ariosto, R Pacelli, M Pastore, F Ginelli, M Gherardi, and P Rotondo. Statistical mechanics of deep learning beyond the infinite-width limit. *arXiv preprint arXiv:2209.04882*, 2022.

[ASKL23] Luca Arnaboldi, Ludovic Stephan, Florent Krzakala, and Bruno Loureiro. From high-dimensional & mean-field dynamics to dimensionless odes: A unifying approach to sgd in two-layers networks. In *The Thirty Sixth Annual Conference on Learning Theory*, pages 1199–1227. PMLR, 2023.

[AZLL19] Zeyuan Allen-Zhu, Yuanzhi Li, and Yingyu Liang. Learning and generalization in overparameterized neural networks, going beyond two layers. In *Advances in neural information processing systems*, pages 6158–6169, 2019.

[AZVP24] Alexander Atanasov, Jacob A. Zavatone-Veth, and Cengiz Pehlevan. Scaling and renormalization in high-dimensional regression. *arXiv preprint arXiv:2405.00592*, 2024.

[BAP24] Blake Bordelon, Alexander Atanasov, and Cengiz Pehlevan. A dynamical model of neural scaling laws. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 4345–4382. PMLR, 2024.

[BAP25] Blake Bordelon, Alexander Atanasov, and Cengiz Pehlevan. How feature learning can improve neural scaling laws. *Journal of Statistical Mechanics: Theory and Experiment*, 2025(8):084002, 2025.

[Bar92] Andrew R Barron. Neural net approximation. In *Proc. 7th Yale Workshop on Adaptive and Learning Systems*, volume 1, pages 69–72, 1992.

[Bar93] Andrew R. Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory*, 39(3):930–945, 1993.

[Bar94] Andrew R. Barron. Approximation and estimation bounds for artificial neural networks. *Machine Learning*, 14(1):115–133, 1994.

[BBC21] Peter Bartlett, Sébastien Bubeck, and Yeshwanth Cherapanamjeri. Adversarial examples in multi-layer random relu networks. *Advances in Neural Information Processing Systems*, 34:9241–9252, 2021.

[BBCC11] Teodor Banica, Serban Teodor Belinschi, Mireille Capitaine, and Benoit Collins. Free bessel laws. *Canadian Journal of Mathematics*, 63(1):3–37, 2011.

[BBPV23] Alberto Bietti, Joan Bruna, and Loucas Pillaud-Vivien. On learning gaussian multi-index models with gradient flow. *arXiv preprint arXiv:2310.19793*, 2023.

[BBSS22] Alberto Bietti, Joan Bruna, Clayton Sanford, and Min Jae Song. Learning single-index models with shallow neural networks. *Advances in Neural Information Processing Systems*, 35:9768–9783, 2022.

[BCCZ23] Ian Banta, Tianji Cai, Nathaniel Craig, and Zhengkang Zhang. Structures of neural network effective theories. *arXiv preprint arXiv:2305.02334*, 2023.

[BCP20] Blake Bordelon, Abdulkadir Canatar, and Cengiz Pehlevan. Spectrum dependent learning curves in kernel regression and wide neural networks. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1024–1034. PMLR, 2020.

[BCP24] Blake Bordelon, Hamza Tahir Chaudhry, and Cengiz Pehlevan. Infinite limits of multi-head transformer dynamics. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[BDK+21] Yasaman Bahri, Ethan Dyer, Jared Kaplan, Jaehoon Lee, and Utkarsh Sharma. Explaining neural scaling laws. *arXiv preprint arXiv:2102.06701*, 2021.

[BES+22] Jimmy Ba, Murat A Erdogdu, Taiji Suzuki, Zhichao Wang, Denny Wu, and Greg Yang. High-dimensional asymptotics of feature learning: How one gradient step improves the representation. *arXiv preprint arXiv:2205.01445*, 2022.

[BES+23] Jimmy Ba, Murat A Erdogdu, Taiji Suzuki, Zhichao Wang, and Denny Wu. Learning in the presence of low-dimensional structure: a spiked random matrix perspective. *Advances in Neural Information Processing Systems*, 36:17420–17449, 2023.

[BGV+25] P Baglioni, L Giambagli, A Vezzani, R Burioni, P Rotondo, and R Pacelli. Kernel shape renormalization explains output-output correlations in finite bayesian one-hidden-layer networks. *Physical Review E*, 111(6):065312, 2025.

[BH89] Pierre Baldi and Kurt Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural networks*, 2(1):53–58, 1989.

[BLR85] Philippe Bougerol, Peter Lacroix, Jean as well as Huber, and Murray (eds) Rosenblatt. *The concentration of measure phenomenon*. Progress in Probability, 1985.

[BMR21] Peter L Bartlett, Andrea Montanari, and Alexander Rakhlin. Deep learning: a statistical viewpoint. *arXiv preprint arXiv:2103.09177*, 2021.

[BMZ23] Raphaël Berthier, Andrea Montanari, and Kangjie Zhou. Learning time-scales in two-layers neural networks. *arXiv preprint arXiv:2303.00055*, 2023.

[BNL+23] Blake Bordelon, Lorenzo Noci, Mufan Bill Li, Boris Hanin, and Cengiz Pehlevan. Depthwise hyperparameter transfer in residual networks: Dynamics and scaling limit. *arXiv preprint arXiv:2309.16620*, 2023.

[BP22] Blake Bordelon and Cengiz Pehlevan. Self-consistent dynamical field theory of kernel evolution in wide neural networks. *Advances in Neural Information Processing Systems*, 35:32240–32256, 2022.

[BP24] Blake Bordelon and Cengiz Pehlevan. Dynamics of finite width kernel and prediction fluctuations in mean field neural networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2024(10):104021, 2024.

[BP25] Blake Bordelon and Cengiz Pehlevan. Deep linear network training dynamics from random initialization: Data, width, depth, and hyperparameter transfer. *arXiv preprint arXiv:2502.02531*, 2025.

[BTB+23] Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E. Burke, Tristan Hume, Shan Carter, Tom Henighan, and Chris Olah. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023. https://transformer-circuits.pub/2023/monosemantic-features.

[C+23] Aakanksha Chowdhery et al. PaLM: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24:240:1–240:113, 2023.

[CB18] Lenaic Chizat and Francis Bach. On the global convergence of gradient descent for over-parameterized models using optimal transport. In *Advances in neural information processing systems*, pages 3036–3046, 2018.

[CB20] Lenaic Chizat and Francis Bach. 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, 2020.

[CBP21] Abdulkadir Canatar, Blake Bordelon, and Cengiz Pehlevan. Spectral bias and task-model alignment explain generalization in kernel regression and infinitely wide neural networks. *Nature Communications*, 12(1):2914, 2021.

[CDI24] Aditya Cowsik, Kfir Dolev, and Alex Infanger. The persian rug: solving toy models of superposition using large-scale symmetries. *arXiv preprint arXiv:2410.12101*, 2024.

[CH20] Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. *arXiv preprint arXiv:2003.01690*, 2020.

[CNWR25] Sinho Chewi, Jonathan Niles-Weed, and Philippe Rigollet. *Statistical Optimal Transport*, volume 2364 of *Lecture Notes in Mathematics*. Springer, Cham, 2025. École d'Été de Probabilités de Saint-Flour XLIX—2019.

[CPD+24] Hugo Cui, Luca Pesce, Yatin Dandi, Florent Krzakala, Yue M Lu, Lenka Zdeborová, and Bruno Loureiro. Asymptotics of feature learning in two-layer networks after one gradient-step. *arXiv preprint arXiv:2402.04980*, 2024.

[CPV12] Andrea Crisanti, Giovanni Paladin, and Angelo Vulpiani. *Products of random matrices: in Statistical Physics*, volume 104. Springer Science & Business Media, 2012.

[CRK19] Jeremy M. Cohen, Elan Rosenfeld, and J. Zico Kolter. Certified adversarial robustness via randomized smoothing. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1310–1320. PMLR, 2019.

[CW17] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57, 2017.

[Cyb89] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, 2(4):303–314, 1989.

[DGA] Ethan Dyer and Guy Gur-Ari. Asymptotics of wide networks from feynman diagrams. In *2019 ICML Workshop on Physics for Deep Learning*.

[DHDG24] Darshil Doshi, Tianyu He, Aritra Das, and Andrey Gromov. Grokking modular polynomials. *arXiv preprint arXiv:2406.03495*, 2024.

[DKL+24] Yatin Dandi, Florent Krzakala, Bruno Loureiro, Luca Pesce, and Ludovic Stephan. How two-layer neural networks learn, one (giant) step at a time. *Journal of Machine Learning Research*, 25(349):1–65, 2024.

[DLM24] Leonardo Defilippis, Bruno Loureiro, and Theodor Misiakiewicz. Dimension-free deterministic equivalents and scaling laws for random feature regression. In *Advances in Neural Information Processing Systems*, volume 37, pages 104630–104693, 2024.

[DLS22] Alexandru Damian, Jason Lee, and Mahdi Soltanolkotabi. Neural networks can learn representations with gradient descent. In *Conference on Learning Theory*, pages 5413–5452. PMLR, 2022.

[DLT+18] Simon S Du, Jason D Lee, Yuandong Tian, Barnabas Poczos, and Aarti Singh. Gradient descent learns one-hidden-layer cnn: Don't be afraid of spurious local minima. *ICML*, 2018.

[DTA+24] Yatin Dandi, Emanuele Troiani, Luca Arnaboldi, Luca Pesce, Lenka Zdeborová, and Florent Krzakala. The benefits of reusing batches for gradient descent in two-layer networks: Breaking the curse of information and leap exponents. *arXiv preprint arXiv:2402.03220*, 2024.

[DVSH18] Felix Draxler, Kambis Veschgini, Manfred Salmhofer, and Fred Hamprecht. Essentially no barriers in neural network energy landscape. In *International conference on machine learning*, pages 1309–1318. PMLR, 2018.

[DYZ23] Emily Dinan, Sho Yaida, and Susan Zhang. Effective theory of transformers at initialization. *arXiv preprint arXiv:2304.02034*, 2023.

[DZDT24] Zhengxiao Du, Aohan Zeng, Yuxiao Dong, and Jie Tang. Understanding emergent abilities of language models from the loss perspective. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

[DZN+25] Nolan Dey, Bin Claire Zhang, Lorenzo Noci, Mufan Li, Blake Bordelon, Shane Bergsma, Cengiz Pehlevan, Boris Hanin, and Joel Hestness. Don't be lazy: Completep enables compute-efficient deep transformers. *arXiv preprint arXiv:2505.01618*, 2025.

[DZPS19] Simon S. Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks. In *International Conference on Learning Representations*, 2019.

[EHO+22] Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, et al. Toy models of superposition. *arXiv preprint arXiv:2209.10652*, 2022.

[ESSN21] Rahim Entezari, Hanie Sedghi, Olga Saukh, and Behnam Neyshabur. The role of permutation invariance in linear mode connectivity of neural networks. *arXiv preprint arXiv:2110.06296*, 2021.

[FK60] Harry Furstenberg and Harry Kesten. Products of random matrices. *The Annals of Mathematical Statistics*, 31(2):457–469, 1960.

[FLD+24] Kirsten Fischer, Javed Lindner, David Dahmen, Zohar Ringel, Michael Krämer, and Moritz Helias. Critical feature learning in deep neural networks. *arXiv preprint arXiv:2405.10761*, 2024.

[FW20] Zhou Fan and Zhichao Wang. Spectra of the conjugate kernel and neural tangent kernel for linear-width neural networks. *Advances in neural information processing systems*, 33:7710–7721, 2020.

[G+22] Deep Ganguli et al. Predictability and surprise in large generative models. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*, pages 1747–1764, 2022.

[GB10] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[GIP+18] Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P Vetrov, and Andrew G Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns. *Advances in neural information processing systems*, 31, 2018.

[GMMM19] Behrooz Ghorbani, Song Mei, Theodor Misiakiewicz, and Andrea Montanari. Linearized two-layers neural networks in high dimension. *arXiv preprint arXiv:1904.12191*, 2019.

[Gro23] Andrey Gromov. Grokking modular arithmetic. *arXiv preprint arXiv:2301.02679*, 2023.

[GS18] Vadim Gorin and Yi Sun. Gaussian fluctuations for products of random matrices. *arXiv preprint arXiv:1812.06532*, 2018.

[GSS15] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.

[Han18] Boris Hanin. Which neural net architectures give rise to exploding and vanishing gradients? In *Advances in Neural Information Processing Systems*, 2018.

[Han21] Boris Hanin. Ridgeless interpolation with shallow relu networks in 1d is nearest neighbor curvature extrapolation and provably generalizes on lipschitz functions. *arXiv preprint arXiv:2109.12960*, 2021.

[Han23] Boris Hanin. Random neural networks in the infinite width limit as gaussian processes. *Annals of Applied Probability. Vol. 33, No. 6A, 4798 – 4819*, 2023.

[Han24] Boris Hanin. Random fully connected neural networks as perturbatively solvable hierarchies. *Journal of Machine Learning Research*, 25(267):1–58, 2024.

[HBM+22] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

[HBSDN20] Jiri Hron, Yasaman Bahri, Jascha Sohl-Dickstein, and Roman Novak. Infinite attention: Nngp and ntk for deep attention networks. In *International Conference on Machine Learning*, pages 4376–4386. PMLR, 2020.

[HCH+23] Tom Henighan, Shan Carter, Tristan Hume, Nelson Elhage, Robert Lasenby, Stanislav Fort, Nicholas Schiefer, and Chris Olah. Superposition, memorization, and double descent. *Transformer Circuits Thread*, 2023. https://transformer-circuits.pub/2023/toy-double-descent/index.html.

[HJ25] Boris Hanin and Tianze Jiang. Global universality of singular values in products of many large random matrices. *arXiv preprint arXiv:2503.07872*, 2025.

[HMRT22] Trevor Hastie, Andrea Montanari, Saharon Rosset, and Ryan J Tibshirani. Surprises in high-dimensional ridgeless least squares interpolation. *The Annals of Statistics*, 50(2):949–986, 2022.

[HN20a] Boris Hanin and Mihai Nica. Finite depth and width corrections to the neural tangent kernel. *ICLR 2020*, 2020.

[HN20b] Boris Hanin and Mihai Nica. Products of many large random matrices and gradients in deep neural networks. *Communications in Mathematical Physics*, 376(1):287–322, 2020.

[HNA+17] Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md Patwary, Mostofa Ali, Yang Yang, and Yanqi Zhou. Deep learning scaling is predictable, empirically. *arXiv preprint arXiv:1712.00409*, 2017.

[HY20] Jiaoyang Huang and Horng-Tzer Yau. Dynamics of deep neural networks and neural tangent hierarchy. In *International Conference on Machine Learning*, pages 4542–4551. PMLR, 2020.

[HZ23] Boris Hanin and Alexander Zlokapa. Bayesian interpolation with deep linear networks. *Proceedings of the National Academy of Sciences*, 120(23):e2301345120, 2023.

[HZ26] Boris Hanin and Alexander Zlokapa. Gibbs measures from deep shaped multilayer perceptrons. *Physical Review Letters*, 136(6):067301, 2026.

[IN92] Marco Isopi and Charles M Newman. The triangle law for lyapunov exponents of large random matrices. *Communications in mathematical physics*, 143(3):591–598, 1992.

[Jac22] Arthur Jacot. Implicit bias of large depth networks: a notion of rank for nonlinear functions. *arXiv preprint arXiv:2209.15055*, 2022.

[JBPH26] Tianze Jiang, Blake Bordelon, Cengiz Pehlevan, and Boris Hanin. Hyperparameter transfer with mixture-of-expert layers. *arXiv preprint arXiv:2601.20205*, 2026.

[JGH18] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in neural information processing systems*, pages 8571–8580, 2018.

[JGŞ+21] Arthur Jacot, François Ged, Berfin Şimşek, Clément Hongler, and Franck Gabriel. Saddle-to-saddle dynamics in deep linear networks: Small initialization training, symmetry, and sparsity. *arXiv preprint arXiv:2106.15933*, 2021.

[JT19] Ziwei Ji and Matus Telgarsky. Gradient descent aligns the layers of deep linear networks. *ICLR*, 2019.

[JT20] Ziwei Ji and Matus Telgarsky. Directional convergence and alignment in deep learning. *Advances in Neural Information Processing Systems*, 33:17176–17186, 2020.

[Kac56] Mark Kac. Foundations of kinetic theory. In *Proceedings of the Third Berkeley Symposium on Mathematical Statistics and Probability, Volume III: Contributions to Astronomy and Physics*, pages 171–197. 1956.

[Kaw16] Kenji Kawaguchi. Deep learning without poor local minima. In *Advances in Neural Information Processing Systems*, pages 586–594, 2016.

[KBGP24] Tanishq Kumar, Blake Bordelon, Samuel J. Gershman, and Cengiz Pehlevan. Grokking as the transition from lazy to rich training dynamics. In *The Twelfth International Conference on Learning Representations*, 2024.

[KMH+20] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[Lac23] Daniel Lacker. Hierarchies, entropy, and quantitative propagation of chaos for mean field diffusions. *Probability and Mathematical Physics*, 4(2):377–432, 2023.

[LBBS24] Noam Levi, Alon Beck, and Yohai Bar-Sinai. Grokking in linear estimators – a solvable model that groks without understanding. In *The Twelfth International Conference on Learning Representations*, 2024.

[LBN+18] Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. Deep neural networks as gaussian processes. *ICML 2018 andarXiv:1711.00165*, 2018.

[LBOM02] Yann LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–50. Springer, 2002.

[LBS+24] Sheng Lu, Irina Bigoulaeva, Rachneet Sachdeva, Harish Tayyar Madabushi, and Iryna Gurevych. Are emergent abilities in large language models just in-context learning? In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5098–5139. Association for Computational Linguistics, 2024.

[LGJ20] Meng Liu, Hongyang Gao, and Shuiwang Ji. Towards deeper graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 338–348, 2020.

[LHY25] Xiangchao Li, Xiao Han, and Qing Yang. Eigen analysis of conjugate kernel and neural tangent kernel. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 34490–34508. PMLR, 2025.

[LJL+24] Kaifeng Lyu, Jikai Jin, Zhiyuan Li, Simon Shaolei Du, Jason D. Lee, and Wei Hu. Dichotomy of early and late phase implicit biases can provably induce grokking. In *The Twelfth International Conference on Learning Representations*, 2024.

[LK17] Haihao Lu and Kenji Kawaguchi. Depth creates no bad local minima. *arXiv preprint arXiv:1702.08580*, 2017.

[LKN+22] Ziming Liu, Ouail Kitouni, Niklas S. Nolte, Eric J. Michaud, Max Tegmark, and Mike Williams. Towards understanding grokking: An effective theory of representation learning. In *Advances in Neural Information Processing Systems*, volume 35, pages 34651–34663, 2022.

[LMT23] Ziming Liu, Eric J. Michaud, and Max Tegmark. Omnigrok: Grokking beyond algorithmic data. In *The Eleventh International Conference on Learning Representations*, 2023.

[LNR22] Mufan Bill Li, Mihai Nica, and Daniel M Roy. The neural covariance sde: Shaped infinite depth-and-width networks at initialization. *NeurIPS 2022*, 2022.

[LS21] Qianyi Li and Haim Sompolinsky. Statistical mechanics of deep linear neural networks: The backpropagating kernel renormalization. *Physical Review X*, 11(3):031059, 2021.

[LWK+24] Licong Lin, Jingfeng Wu, Sham M. Kakade, Peter L. Bartlett, and Jason D. Lee. Scaling laws in linear regression: Compute, parameters, and data. In *Advances in Neural Information Processing Systems*, volume 37, pages 60556–60606, 2024.

[LWW18] Dang-Zheng Liu, Dong Wang, and Yanhui Wang. Lyapunov exponent, universality and phase transition for products of random matrices. *arXiv preprint arXiv:1810.00433*, 2018.

[LZB22] Chaoyue Liu, Libin Zhu, and Mikhail Belkin. Loss landscapes and optimization in over-parameterized non-linear systems and neural networks. *Applied and Computational Harmonic Analysis*, 59:85–116, 2022.

[MBD+21] James Martens, Andy Ballard, Guillaume Desjardins, Grzegorz Swirszcz, Valentin Dalibard, Jascha Sohl-Dickstein, and Samuel S Schoenholz. Rapid training of deep neural networks without skip connections or normalization layers using deep kernel shaping. *arXiv preprint arXiv:2110.01765*, 2021.

[MCCD13] Tomáš Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013, Workshop Track Proceedings*, 2013.

[MDFF16] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: A simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2574–2582, 2016.

[Mél96] Sylvie Méléard. Asymptotic behaviour of some interacting particle systems; mckean–vlasov and boltzmann models. In Denis Talay and Luciano Tubaro, editors, *Probabilistic Models for Nonlinear Partial Differential Equations*, volume 1627 of *Lecture Notes in Mathematics*, pages 42–95. Springer, Berlin, Heidelberg, 1996.

[MHPG+22] Alireza Mousavi-Hosseini, Sejun Park, Manuela Girotti, Ioannis Mitliagkas, and Murat A Erdogdu. Neural networks efficiently learn low-dimensional representations with sgd. *arXiv preprint arXiv:2209.14863*, 2022.

[MHWSE23] Alireza Mousavi-Hosseini, Denny Wu, Taiji Suzuki, and Murat A Erdogdu. Gradient-based feature learning under structured data. *arXiv preprint arXiv:2309.03843*, 2023.

[MLHD23] Behrad Moniri, Donghwan Lee, Hamed Hassani, and Edgar Dobriban. A theory of non-linear feature learning with one gradient step in two-layer neural networks. *arXiv preprint arXiv:2310.07891*, 2023.

[MM18] Marco Mondelli and Andrea Montanari. On the connection between learning two-layers neural networks and tensor decomposition. *arXiv preprint arXiv:1802.07301*, 2018.

[MM19] Song Mei and Andrea Montanari. The generalization error of random features regression: Precise asymptotics and the double descent curve. *Communications on Pure and Applied Mathematics*, 2019.

[MM23] Theodor Misiakiewicz and Andrea Montanari. Six lectures on linearized neural networks. *arXiv preprint arXiv:2308.13431*, 2023.

[MMN18] Song Mei, Andrea Montanari, and Phan-Minh Nguyen. A mean field view of the landscape of two-layer neural networks. *Proceedings of the National Academy of Sciences*, 115(33):E7665–E7671, 2018.

[MMS+18] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

[MRS22] Alexander Maloney, Daniel A Roberts, and James Sully. A solvable model of neural scaling laws. *arXiv preprint arXiv:2210.16859*, 2022.

[MS24] Theodor Misiakiewicz and Basil Saeed. A non-asymptotic theory of kernel ridge regression: deterministic equivalents, test error, and GCV estimator. *arXiv preprint arXiv:2403.08938*, 2024.

[NCL+23] Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. In *The Eleventh International Conference on Learning Representations*, 2023.

[NDSR21] Gadi Naveh, Oded Ben David, Haim Sompolinsky, and Zohar Ringel. Predicting the outputs of finite deep neural networks trained with noisy gradients. *Physical Review E*, 104(6):064301, 2021.

[Nea96] Radford M Neal. Priors for infinite networks. In *Bayesian Learning for Neural Networks*, pages 29–53. Springer, 1996.

[New86] Charles M Newman. The distribution of lyapunov exponents: Exact results for random matrices. *Communications in mathematical physics*, 103(1):121–126, 1986.

[NR21] Gadi Naveh and Zohar Ringel. A self consistent theory of gaussian processes captures feature learning effects in finite cnns. *Advances in Neural Information Processing Systems*, 34, 2021.

[OWSS19] Greg Ongie, Rebecca Willett, Daniel Soudry, and Nathan Srebro. A function space view of bounded norm infinite width relu nets: The multivariate case. *ICRL 2020. arXiv:1910.01635*, 2019.

[Pau20] Sayak Paul. An introduction to adversarial examples in deep learning. `https://wandb.ai/authors/adv-dl/reports/An-Introduction-to-Adversarial-Examples-in-Deep-Learning--VmlldzoyMTQwODM`, 2020. Weights & Biases report. Created August 23, 2020. Updated November 1, 2022.

[PBE+22] Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets. *arXiv preprint arXiv:2201.02177*, 2022.

[Pin99] Allan Pinkus. Approximation theory of the MLP model in neural networks. *Acta Numerica*, 8:143–195, 1999.

[PLR+16] Ben Poole, Subhaneil Lahiri, Maithra Raghu, Jascha Sohl-Dickstein, and Surya Ganguli. Exponential expressivity in deep neural networks through transient chaos. In *Advances in neural information processing systems*, pages 3360–3368, 2016.

[PPAP25] Courtney Paquette, Elliot Paquette, Ben Adlam, and Jeffrey Pennington. Homogenization of SGD in high-dimensions: exact dynamics and generalization properties. *Mathematical Programming*, 214(1–2):1–90, 2025.

[PPXP24] Elliot Paquette, Courtney Paquette, Lechao Xiao, and Jeffrey Pennington. 4+3 phases of compute-optimal neural scaling laws. In *Advances in Neural Information Processing Systems*, volume 37, pages 16459–16537, 2024.

[PSG18] Jeffrey Pennington, Samuel Schoenholz, and Surya Ganguli. The emergence of spectral universality in deep networks. In *International Conference on Artificial Intelligence and Statistics*, pages 1924–1932. PMLR, 2018.

[RFL+25] Noa Rubin, Kirsten Fischer, Javed Lindner, Inbar Seroussi, Zohar Ringel, Michael Krämer, and Moritz Helias. From kernels to features: A multi-scale adaptive theory of feature learning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 52225–52257, 2025.

[Rob21] Daniel A Roberts. Sgd implicitly regularizes generalization error. *arXiv preprint arXiv:2104.04874*, 2021.

[RRSH24] Noa Rubin, Zohar Ringel, Inbar Seroussi, and Moritz Helias. A unified approach to feature learning in bayesian neural networks. In *High-dimensional Learning Dynamics 2024: The Emergence of Structure and Reasoning*, 2024.

[RSR24] Noa Rubin, Inbar Seroussi, and Zohar Ringel. Grokking as a first order phase transition in two layer networks. In *The Twelfth International Conference on Learning Representations*, 2024.

[RVE18] Grant Rotskoff and Eric Vanden-Eijnden. Parameters as interacting particles: long time convergence and asymptotic error scaling of neural networks. *Advances in neural information processing systems*, 31, 2018.

[RYH22] Daniel A Roberts, Sho Yaida, and Boris Hanin. *The Principles of Deep Learning Theory: An Effective Theory Approach to Understanding Neural Networks*. Cambridge University Press, 2022.

[S+23] Aarohi Srivastava et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*, 2023.

[SESS19] Pedro Savarese, Itay Evron, Daniel Soudry, and Nathan Srebro. How do infinite width bounded norm networks look in function space? *COLT arXiv:1902.05040*, 2019.

[SGGSD17] Samuel S Schoenholz, Justin Gilmer, Surya Ganguli, and Jascha Sohl-Dickstein. Deep information propagation. *ICLR 2017 and arXiv:1611.01232*, 2017.

[SHN+18] Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. *The Journal of Machine Learning Research*, 19(1):2822–2878, 2018.

[SMG14] Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *ICLR*, 2014.

[SMK23] Rylan Schaeffer, Brando Miranda, and Sanmi Koyejo. Are emergent abilities of large language models a mirage? In *Advances in Neural Information Processing Systems*, volume 36, 2023.

[SNR23] Inbar Seroussi, Gadi Naveh, and Zohar Ringel. Separation of scales and a thermodynamic description of feature learning in some cnns. *Nature Communications*, 14(1):908, 2023.

[SS95] David Saad and Sara A Solla. Exact solution for on-line learning in multilayer neural networks. *Physical Review Letters*, 74(21):4337, 1995.

[SS20] Justin Sirignano and Konstantinos Spiliopoulos. Mean field analysis of neural networks: A central limit theorem. *Stochastic Processes and their Applications*, 130(3):1820–1852, 2020.

[SS21] Justin Sirignano and Konstantinos Spiliopoulos. Mean field analysis of deep neural networks. *Mathematics of Operations Research*, 2021.

[SSJ+22] Adam Scherlis, Kshitij Sachan, Adam S. Jermyn, Joe Benton, and Buck Shlegeris. Polysemanticity and capacity in neural networks. *arXiv preprint arXiv:2210.01892*, 2022.

[Szn91] Alain-Sol Sznitman. Topics in propagation of chaos. In *École d'Été de Probabilités de Saint-Flour XIX—1989*, volume 1464 of *Lecture Notes in Mathematics*, pages 165–251. Springer, Berlin, Heidelberg, 1991.

[SZS+14] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.

[Tro15] Joel A. Tropp. An introduction to matrix concentration inequalities. *Foundations and Trends in Machine Learning*, 8(1–2):1–230, 2015.

[WGL+20] Blake Woodworth, Suriya Gunasekar, Jason D Lee, Edward Moroshko, Pedro Savarese, Itay Golan, Daniel Soudry, and Nathan Srebro. Kernel and rich regimes in overparametrized models. *arXiv preprint arXiv:2002.09277*, 2020.

[WTB+22] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent abilities of large language models. *Transactions on Machine Learning Research*, 2022.

[WWS+22] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837, 2022.

[Yai20] Sho Yaida. Non-gaussian processes and neural networks at finite widths. *MSML*, 2020.

[Yai22] Sho Yaida. Meta-principled family of hyperparameter scaling strategies. *arXiv preprint arXiv:2210.04909*, 2022.

[Yan19] Greg Yang. Tensor programs i: Wide feedforward or recurrent neural networks of any architecture are gaussian processes. *arXiv preprint arXiv:1910.12478*, 2019.

[YCBL14] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *Advances in neural information processing systems*, 27, 2014.

[YCN+15] Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*, 2015.

[YH21] Greg Yang and Edward J Hu. Tensor programs iv: Feature learning in infinite-width neural networks. In *International Conference on Machine Learning*, pages 11727–11737. PMLR, 2021.

[YHB+21] Ge Yang, Edward Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tuning large neural networks via zero-shot hyperparameter transfer. *Advances in Neural Information Processing Systems*, 34:17084–17097, 2021.

[YYZH23] Greg Yang, Dingli Yu, Chen Zhu, and Soufiane Hayou. Tensor programs vi: Feature learning in infinite-depth neural networks. *arXiv preprint arXiv:2310.02244*, 2023.

[Zha25] Zhengkang Zhang. Neural scaling laws from large-n field theory: solvable model beyond the ridgeless limit. *Machine Learning: Science and Technology*, 6(2):025010, 2025.

[ŽI24] Bojan Žunkovič and Enej Ilievski. Grokking phase transitions in learning local rules with gradient descent. *Journal of Machine Learning Research*, 25(199):1–52, 2024.

[ZLB22] Libin Zhu, Chaoyue Liu, and Misha Belkin. Transition to linearity of general neural networks with directed acyclic graph architecture. *Advances in neural information processing systems*, 35:5363–5375, 2022.

[ZVTP22] Jacob A. Zavatone-Veth, William L. Tong, and Cengiz Pehlevan. Contrasting random and learned features in deep bayesian linear regression. *Phys. Rev. E*, 105:064118, 2022.

[ZYJ+19] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P. Xing, Laurent El Ghaoui, and Michael I. Jordan. Theoretically principled trade-off between robustness and accuracy. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 7472–7482. PMLR, 2019.