The Principles of Deep Learning Theory

An Effective Theory Approach to Understanding Neural Networks

Daniel A. Roberts and Sho Yaida

based on research in collaboration with

Boris Hanin

drob@mit.edu, shoyaida@fb.com

Contents

Preface

> *This has necessitated a complete break from the historical line of development, but this break is an advantage through enabling the approach to the new ideas to be made as direct as possible.*
>
> P. A. M. Dirac in the 1930 preface of *The Principles of Quantum Mechanics* [1].

This is a research monograph in the style of a textbook about the theory of deep learning. While this book might look a little different from the other deep learning books that you've seen before, we assure you that it is appropriate for everyone with knowledge of linear algebra, multivariable calculus, and informal probability theory, and with a healthy interest in neural networks. Practitioner and theorist alike, we want all of you to enjoy this book. Now, let us tell you some things.

First and foremost, in this book we've strived for pedagogy in every choice we've made, placing intuition above formality. This doesn't mean that calculations are incomplete or sloppy; quite the opposite, we've tried to provide full details of every calculation – of which there are certainly very many – and place a particular emphasis on the tools needed to carry out related calculations of interest. In fact, understanding how the calculations are done is as important as knowing their results, and thus often our pedagogical focus is on the details therein.

Second, while we present the details of all our calculations, we've kept the experimental confirmations to the privacy of our own computerized notebooks. Our reason for this is simple: while there's much to learn from explaining a derivation, there's not much more to learn from printing a verification plot that shows two curves lying on top of each other. Given the simplicity of modern deep-learning codes and the availability of compute, it's easy to verify any formula on your own; we certainly have thoroughly checked them all this way, so if knowledge of the existence of such plots are comforting to you, know at least that they do exist on our personal and cloud-based hard drives.

Third, our main focus is on realistic models that are used by the deep learning community in practice: we want to study *deep* neural networks. In particular, this means that *(i)* a number of special results on single-hidden-layer networks will not be discussed and *(ii)* the *infinite-width limit* of a neural network – which corresponds to a zero-hidden-layer network – will be introduced only as a starting point. All such idealized models will eventually be *perturbed* until they correspond to a real model. We certainly acknowledge that there's a vibrant community of deep-learning theorists devoted to

exploring different kinds of idealized theoretical limits. However, our interests are fixed firmly on providing explanations for the tools and approaches used by practitioners, in an effort to shed light on what makes them work so well.

Fourth, a large part of the book is focused on deep multilayer perceptrons. We made this choice in order to pedagogically illustrate the power of the effective theory framework – not due to any technical obstruction – and along the way we give pointers for how this formalism can be extended to other architectures of interest. In fact, we expect that many of our results have a broad applicability, and we've tried to focus on aspects that we expect to have lasting and universal value to the deep learning community.

Fifth, while much of the material is novel and appears for the first time in this book, and while much of our framing, notation, language, and emphasis breaks with the historical line of development, we're also very much indebted to the deep learning community. With that in mind, throughout the book we will try to reference important prior contributions, with an emphasis on recent seminal deep-learning results rather than on being completely comprehensive. Additional references for those interested can easily be found within the work that we cite.

Sixth, this book initially grew out of a research project in collaboration with Boris Hanin. To account for his effort and then support, we've accordingly commemorated him on the cover. More broadly, we've variously appreciated the artwork, discussions, encouragement, epigraphs, feedback, management, refereeing, reintroduction, and support from Rafael Araujo, Léon Bottou, Paul Dirac, Ethan Dyer, John Frank, Ross Girshick, Vince Higgs, Yoni Kahn, Yann LeCun, Kyle Mahowald, Eric Mintun, Xiaoliang Qi, Mike Rabbat, David Schwab, Stephen Shenker, Eva Silverstein, PJ Steiner, DJ Strouse, and Jesse Thaler. Organizationally, we're grateful to FAIR and Facebook, Diffeo and Salesforce, MIT and IAIFI, and Cambridge University Press and the arXiv.

Seventh, given intense (and variously uncertain) spacetime and energy-momentum commitment that writing this book entailed, Dan is grateful to Aya, Lumi, and Lisa Yaida; from the dual sample-space perspective, Sho is grateful to Adrienne Rothschilds and would be retroactively grateful to any hypothetical future Mark or Emily that would have otherwise been thanked in this paragraph.

Eighth, we hope that this book spreads our optimism that it *is* possible to have a general theory of deep learning, one that's both derived from first principles and at the same time focused on describing how realistic models actually work: nearly-simple phenomena in practice should correspond to nearly-simple effective theories. We dream that this type of thinking will not only lead to more *[redacted]* AI models but also guide us towards a unifying framework for understanding universal aspects of intelligence.

As if that eightfold way of prefacing the book wasn't nearly-enough already, please note: this book has a website, `deeplearningtheory.com`, and you may want to visit it in order to determine whether the error that you just discovered is already common knowledge. If it's not, please let us know. There may be pie.

Dan Roberts & Sho Yaida
Remotely Located
June, 2021

Chapter 0

Initialization

> *The simulation is such that [one] generally perceives the sum of many billions of elementary processes simultaneously, so that the leveling law of large numbers completely obscures the real nature of the individual processes.*
>
> John von Neumann [2]

Thanks to substantial investments into computer technology, modern **artificial intelligence** (AI) systems can now come equipped with many billions of elementary components. When these components are properly initialized and then trained, AI can accomplish tasks once considered so incredibly complex that philosophers have previously argued that only *natural* intelligence systems – i.e. humans – could perform them.

Behind much of this success in AI is **deep learning**. Deep learning uses artificial **neural networks** as an underlying model for AI: while loosely based on *biological* neural networks such as your brain, *artificial* neural networks are probably best thought of as an especially nice way of specifying a flexible set of functions, built out of many basic computational blocks called **neurons**. This model of computation is actually quite different from the one used to power the computer you're likely using to read this book. In particular, rather than *programming* a specific set of instructions to solve a problem directly, deep learning models are *trained* on data from the real world and learn how to solve problems.

The real power of the deep learning framework comes from *deep* neural networks, with many neurons in parallel organized into sequential computational layers, *learning* useful representations of the world. Such **representation learning** transforms data into increasingly refined forms that are helpful for solving an underlying task, and is thought to be a hallmark of success in intelligence, both artificial and biological.

Despite these successes and the intense interest they created, deep learning *theory* is still in its infancy. Indeed, there is a serious disconnect between theory and practice: while practitioners have reached amazing milestones, they have far outpaced the theorists, whose analyses often involve assumptions so unrealistic that they lead to conclusions that are irrelevant to understanding deep neural networks as they are typically

used. More importantly, very little theoretical work directly confronts the *deep* of deep learning, despite a mass of empirical evidence for its importance in the success of the framework.

The goal of this book is to put forth a set of **principles** that enable us to theoretically analyze *deep* neural networks of *actual relevance*. To initialize you to this task, in the rest of this chapter we'll explain at a very high-level both *(i)* why such a goal is even attainable in theory and *(ii)* how we are able to get there in practice.

0.1 An Effective Theory Approach

> *Steam navigation brings nearer together the most distant nations. . . . their theory is very little understood, and the attempts to improve them are still directed almost by chance. . . . We propose now to submit these questions to a deliberate examination.*
>
> Sadi Carnot, commenting on the need for a theory of deep learning [3].

While modern deep learning models are built up from seemingly innumerable elementary computational components, a first-principles *microscopic* description of *how* a trained neural network computes a function from these low-level components is entirely manifest. This microscopic description is just the set of instructions for transforming an input through the many layers of components into an output. Importantly, during the training process, these components become very finely-tuned, and knowledge of the particular tunings is necessary for a system to produce useful output.

Unfortunately, the complexity of these tunings obscures any first-principles *macroscopic* understanding of *why* a deep neural network computes a particular function and not another. With many neurons performing different tasks as part of such a computation, it seems hopeless to think that we can use theory to understand these models at all, and silly to believe that a small set of mathematical *principles* will be sufficient for that job.

Fortunately, **theoretical physics** has a long tradition of finding simple **effective theories** of complicated systems with a large number of components. The immense success of the program of physics in modeling our physical universe suggests that perhaps some of the same tools may be useful for theoretically understanding deep neural networks. To motivate this connection, let's very briefly reflect on the successes of thermodynamics and statistical mechanics, physical theories that together explain from microscopic first principles the macroscopic behavior of systems with many elementary constituents.

A scientific consequence of the Industrial Age, **thermodynamics** arose out of an effort to describe and innovate upon the steam engine – a system consisting of many many particles and perhaps the original *black box*. The laws of thermodynamics, derived from careful empirical observations, were used to codify the mechanics of steam, providing a high-level understanding of these macroscopic *artificial* machines that were transforming society. While the advent of thermodynamics led to tremendous improvements in the

efficiency of steam power, its laws were in no way fundamental.

It wasn't until much later that Maxwell, Boltzmann, and Gibbs provided the missing link between experimentally-derived effective description on the one hand and a first-principles theory on the other hand. Their **statistical mechanics** explains how the macroscopic laws of thermodynamics describing human-scale machines could arise *statistically* from the deterministic dynamics of many microscopic elementary constituents. From this perspective, the laws of thermodynamics were *emergent* phenomena that only appear from the collective statistical behavior of a very large number of microscopic particles. In fact, it was the detailed theoretical predictions derived from statistical mechanics that ultimately led to the general scientific acceptance that matter is really comprised of molecules and atoms. Relentless application of statistical mechanics led to the discovery of *quantum mechanics*, which is a precursor to the invention of the transistor that powers the Information Age, and – taking the long view – is what has allowed us to begin to realize artificial machines that can think intelligently.

Notably, these physical theories originated from a desire to understand *artificial* human-engineered objects, such as the steam engine. Despite a potential misconception, physics doesn't make a distinction between natural and artificial phenomena. Most fundamentally, it's concerned with providing a unified set of principles that account for past empirical observations and predict the result of future experiments; the point of theoretical calculations is to connect measurable outcomes or **observables** directly to the fundamental underlying constants or **parameters** that define the theory. This perspective also implies a tradeoff between the predictive accuracy of a model and its mathematical tractability, and the former must take precedence over the latter for any theory to be successful: a short tether from theory to physical reality is essential. When successful, such theories provide a comprehensive understanding of phenomena and empower practical advances in technology, as exemplified by the statistical-physics bridge from the Age of Steam to the Age of Information.

For our study of deep learning, the key takeaway from this discussion is that a theoretical *matter* simplifies when it is made up of many elementary constituents. Moreover, unlike the molecules of water contained in a box of steam – with their existence once being a controversial conjecture in need of experimental verification – the neurons comprising a deep neural network are put in (the box) by hand. Indeed, in this case we already understand the microscopic laws – *how* a network computes – and so instead our task is to understand the new types of regularity that appear at the macroscopic scale – *why* it computes one particular function rather than another – that emerge from the statistical properties of these gigantic deep learning models.

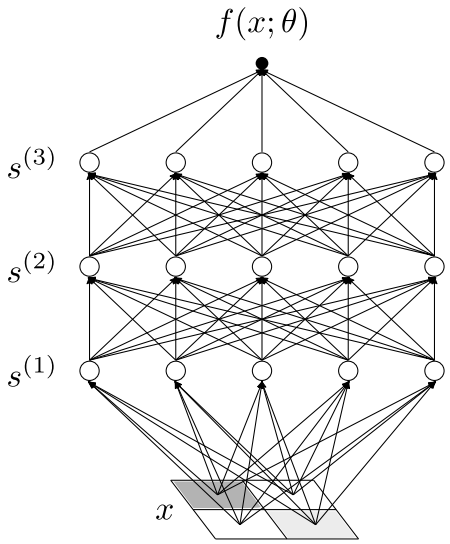

Figure 1: A graph of a simple multilayer neural network, depicting how the input x is transformed through a sequence of intermediate signals, $s^{(1)}$, $s^{(2)}$, and $s^{(3)}$, into the output $f(x;\theta)$. The white circles represent the neurons, the black dot at the top represents the network output, and the parameters θ are implicit; they weight the importance of the different arrows carrying the signals and bias the firing threshold of each neuron.

0.2 The Theoretical Minimum

> *The method is more important than the discovery, because the correct method of research will lead to new, even more valuable discoveries.*
>
> Lev Landau [4].

In this section, we'll give a high-level overview of our method, providing a minimal explanation for why we should expect a first-principles theoretical understanding of deep neural networks to be possible. We'll then fill in all the details in the coming chapters.

In essence, a **neural network** is a recipe for computing a function built out of many computational units called **neurons**. Each neuron is itself a very simple function that considers a weighted sum of incoming signals and then *fires* in a characteristic way by comparing the value of that sum against some threshold. Neurons are then organized in parallel into **layers**, and *deep* neural networks are those composed of multiple layers in sequence. The network is parametrized by the firing thresholds and the weighted connections between the neurons, and, to give a sense of the potential scale, current state-of-the-art neural networks can have over 100 billion parameters. A graph depicting the structure of a much more reasonably-sized neural network is shown in Figure 1.

For a moment, let's ignore all that structure and simply think of a neural network

as a parameterized function

$$f(x;\theta)\,, \tag{0.1}$$

where x is the input to the function and θ is a vector of a large number of **parameters** controlling the shape of the function. For such a function to be useful, we need to somehow tune the high-dimensional parameter vector θ. In practice, this is done in two steps:

- First, we *initialize* the network by randomly sampling the parameter vector θ from a computationally simple probability distribution,

  $$p(\theta)\,. \tag{0.2}$$

  We'll later discuss the theoretical reason why it is a good strategy to have an **initialization distribution** $p(\theta)$ but, more importantly, this corresponds to what is done in practice, and our approach in this book is to have our theoretical analysis correspond to realistic deep learning scenarios.

- Second, we adjust the parameter vector as $\theta \to \theta^\star$, such that the resulting *network function* $f(x;\theta^\star)$ is as close as possible to a desired *target function* $f(x)$:

  $$f(x;\theta^\star) \approx f(x)\,. \tag{0.3}$$

  This is called **function approximation**. To find these tunings $\theta^\star$, we fit the network function $f(x;\theta)$ to **training data**, consisting of many pairs of the form $\big(x, f(x)\big)$ observed from the desired – but only partially observable – target function $f(x)$. Overall, making these adjustments to the parameters is called **training**, and the particular procedure used to tune them is called a **learning algorithm**.

Our goal is to understand this *trained* network function:

$$f(x;\theta^\star)\,. \tag{0.4}$$

In particular, we'd like to understand the macroscopic behavior of this function from a first-principles microscopic description of the network in terms of these trained parameters $\theta^\star$. We'd also like to understand how the function approximation (0.3) works and evaluate how $f(x;\theta^\star)$ uses the training data $\big(x, f(x)\big)$ in its approximation of $f(x)$. Given the high dimensionality of the parameters θ and the degree of fine-tuning required for the approximation (0.3), this goal might seem naive and beyond the reach of any realistic theoretical approach.

One way to more directly see the kinds of technical problems that we'll encounter is to *Taylor expand* our trained network function $f(x;\theta^\star)$ around the initialized value of the parameters θ. Being schematic and ignoring for a moment that θ is a vector and that the derivatives of $f(x;\theta)$ are tensors, we see

$$f(x;\theta^\star) = f(x;\theta) + (\theta^\star - \theta)\,\frac{df}{d\theta} + \frac{1}{2}\,(\theta^\star - \theta)^2\,\frac{d^2 f}{d\theta^2} + \ldots\,, \tag{0.5}$$

where $f(x;\theta)$ and its derivatives on the right-hand side are all evaluated at initialized value of the parameters. This Taylor representation illustrates our three main problems:

Problem 1

In general, the series (0.5) contains an infinite number of terms

$$f\,, \quad \frac{df}{d\theta}\,, \quad \frac{d^2f}{d\theta^2}\,, \quad \frac{d^3f}{d\theta^3}\,, \quad \frac{d^4f}{d\theta^4}\,, \quad \ldots\,, \tag{0.6}$$

and to use this Taylor representation of the function (0.5), in principle we need to compute them all. More specifically, as the difference between the trained and initialized parameters, $(\theta^\star - \theta)$, becomes large, so too does the number of terms needed to get a good approximation of the trained network function $f(x;\theta^\star)$.

Problem 2

Since the parameters θ are randomly sampled from the initialization distribution, $p(\theta)$, each time we initialize our network we get a different function $f(x;\theta)$. This means that each term f, $df/d\theta$, $d^2f/d\theta^2$, $\ldots$, from (0.6) is really a *random function* of the input x. Thus, the initialization induces a distribution over the network function and its derivatives, and we need to determine the mapping,

$$p(\theta) \rightarrow p\left(f, \frac{df}{d\theta}, \frac{d^2f}{d\theta^2}, \ldots\right)\,, \tag{0.7}$$

that takes us from the distribution of initial parameters θ to the *joint* distribution of the network function, $f(x;\theta)$, its **gradient**, $df/d\theta$, its **Hessian**, $d^2f/d\theta^2$, and so on. This is a joint distribution comprised of an infinite number of random functions, and in general such functions will have an intricate statistical dependence. Even if we set aside this infinity of functions for a moment and consider just the marginal distribution of the network function *only*, $p(f)$, there's still no reason to expect that it's analytically tractable.

Problem 3

The learned value of the parameters, $\theta^\star$, is the result of a complicated training process. In general, $\theta^\star$ is not unique and can depend on *everything*:

$$\theta^\star \equiv [\theta^\star]\left(\theta,\, f,\, \frac{df}{d\theta},\, \frac{d^2f}{d\theta^2},\, \ldots;\, \text{learning algorithm};\, \text{training data}\right)\,. \tag{0.8}$$

In practice, the learning algorithm is *iterative*, accumulating changes over many steps, and the dynamics are nonlinear. Thus, the trained parameters $\theta^\star$ will depend in a very complicated way on all the quantities at initialization – such as the specific random sample of the parameters θ, the network function $f(x;\theta)$ and all of its derivatives, $df/d\theta$, $d^2f/d\theta^2$, $\ldots$ – as well as on the details of the learning algorithm and also on the particular pairs, $(x, f(x))$, that comprise the training data. Determining an *analytical* expression for $\theta^\star$ must involve taking all of this into account.

If we could solve all three of these problems, then we could in principle use the Taylor-series representation (0.5) to study the trained network function. More specifically, we'd find a *distribution* over trained network functions

$$p(f^\star) \equiv p\Big(f(x;\theta^\star)\Big|\,\text{learning algorithm; training data}\Big)\,, \tag{0.9}$$

now conditioned in a simple way on the learning algorithm and the data we used for training. Here, by *simple* we mean that it is easy to evaluate this distribution for different algorithms or choices of training data without having to solve a version of **Problem 3** each time. The development of a method for the analytical computation of (0.9) is a principle goal of this book.

Of course, solving our three problems for a general parameterized function $f(x;\theta)$ is not tractable. However, we are not trying to solve these problems in general; we only care about the functions that are deep neural networks. Necessarily, any solution to the above problems will thus have to make use of the particular *structure* of neural-network function. While specifics of how this works form the basis of the book, in the rest of this section we'll try to give intuition for how these complications can be resolved.

A Principle of Sparsity

To elaborate on the structure of neural networks, please scroll back a bit and look at Figure 1. Note that for the network depicted in this figure, each intermediate or *hidden* layer consists of five neurons, and the input x passes through three such hidden layers before the output is produced at the top after the final layer. In general, two essential aspects of a neural network *architecture* are its **width**, n, and its **depth**, L. For the network in Figure 1, we would have $n = 5$, counting the neurons per each hidden layer, and $L = 4$, counting the three hidden layers and the output layer at the top.

As we foreshadowed in §0.1, there are often simplifications to be found in the limit of a large number of components. However, it's not enough to consider any massive macroscopic system, and taking the right limit often requires some care. Regarding the neurons as the components of the network, there are essentially two primal ways that we can make a network grow in size: we can increase its width n holding its depth L fixed, or we can increase its depth L holding its width n fixed. In this case, it will actually turn out that the former limit will make everything really simple, while the latter limit will be hopelessly complicated and useless in practice.

So let's begin by formally taking the limit

$$\lim_{n\to\infty} p(f^\star)\,, \tag{0.10}$$

and studying an *idealized* neural network in this limit. This is known as the **infinite-width limit** of the network, and as a strict limit it's rather *unphysical* for a network: obviously you cannot directly program a function to have an infinite number of components on a finite computer. However, this extreme limit does massively simplify the distribution over trained networks $p(f^\star)$, rendering each of our three problems completely benign:

- Addressing **Problem 1**, all the higher derivative terms $d^k f / d\theta^k$ for $k \geq 2$ will effectively vanish, meaning we only need to keep track of two terms,

$$f\,, \quad \frac{df}{d\theta}\,. \tag{0.11}$$

- Addressing **Problem 2**, the distributions of these random functions will be independent,

$$\lim_{n\to\infty} p\left(f, \frac{df}{d\theta}, \frac{d^2 f}{d\theta^2}, \ldots\right) = p(f)\, p\left(\frac{df}{d\theta}\right)\,, \tag{0.12}$$

  with each marginal distribution factor taking a very simple form.

- Addressing **Problem 3**, the training dynamics become linear and completely independent of the details of the learning algorithm, letting us find a complete analytical solution for $\theta^\star$ in a *closed form*

$$\lim_{n\to\infty} \theta^\star = [\theta^\star]\left(\theta,\, f,\, \frac{df}{d\theta};\, \text{training data}\right)\,. \tag{0.13}$$

As a result, the trained distribution (0.10) is a simple **Gaussian distribution** with a nonzero mean, and we can easily analyze the functions that such networks are computing.

These simplifications are the consequence of a **principle of sparsity**. Even though it seems like we've made the network more complicated by growing it to have an infinite number of components, from the perspective of any particular neuron the input of an infinite number of signals is such that the leveling law of large numbers completely obscures much of the details in the signals. The result is that the *effective theory* of many such infinite-width networks leads to extreme sparsity in their description, e.g. enabling the truncation (0.11).

Unfortunately, the formal infinite-width limit, $n \to \infty$, leads to a poor model of *deep* neural networks: not only is infinite width an unphysical property for a network to possess, but the resulting trained distribution (0.10) also leads to a mismatch between theoretical description and practical observation for networks of more than one layer. In particular, it's empirically known that the distribution over such trained networks *does* depend on the properties of the learning algorithm used to train them. Additionally, we will show in detail that such infinite-width networks cannot learn representations of their inputs: for any input x, its transformations in the hidden layers, $s^{(1)}$, $s^{(2)}$, ..., will remain unchanged from initialization, leading to *random representations* and thus severely restricting the class of functions that such networks are capable of learning. Since nontrivial **representation learning** is an empirically demonstrated essential property of multilayer networks, this really underscores the breakdown of the correspondence between theory and reality in this strict infinite-width limit.

From the theoretical perspective, the problem with this limit is the washing out of the fine details at each neuron due to the consideration of an infinite number of incoming signals. In particular, such an infinite accumulation completely eliminates

the subtle correlations between neurons that get amplified over the course of training for representation learning. To make progress, we'll need to find a way to restore and then study the **interactions** between neurons that are present in realistic *finite-width* networks.

With that in mind, perhaps the infinite-width limit can be corrected in a way such that the corrections become small when the width n is large. To do so, we can use **perturbation theory** – just as we do in physics to analyze interacting systems – and study deep learning using a $1/n$ **expansion**, treating the inverse layer width, $\epsilon \equiv 1/n$, as our small parameter of expansion: $\epsilon \ll 1$. In other words, we're going to back off the strict infinite-width limit and compute the trained distribution (0.9) with the following expansion:

$$p(f^\star) \equiv p^{\{0\}}(f^\star) + \frac{p^{\{1\}}(f^\star)}{n} + \frac{p^{\{2\}}(f^\star)}{n^2} + \ldots \,, \tag{0.14}$$

where $p^{\{0\}}(f^\star) \equiv \lim_{n\to\infty} p(f^\star)$ is the infinite-width limit we discussed above, (0.10), and the $p^{\{k\}}(f^\star)$ for $k \geq 1$ give a series of corrections to this limit.

In this book, we'll in particular compute the first such correction, truncating the expansion as

$$p(f^\star) \equiv p^{\{0\}}(f^\star) + \frac{p^{\{1\}}(f^\star)}{n} + O\left(\frac{1}{n^2}\right) . \tag{0.15}$$

This *interacting theory* is still simple enough to make our three problems tractable:

- Addressing **Problem 1**, now all the higher derivative terms $d^k f/d\theta^k$ for $k \geq 4$ will effectively give contributions of the order $O(1/n^2)$ or smaller, meaning that to capture the leading contributions of order $1/n$, we only need to keep track of four terms:

  $$f \,, \quad \frac{df}{d\theta} \,, \quad \frac{d^2 f}{d\theta^2} \,, \quad \frac{d^3 f}{d\theta^3} \,. \tag{0.16}$$

  Thus, we see that the *principle of sparsity* will still limit the dual effective theory description, though not quite as extensively as in the infinite-width limit.

- Addressing **Problem 2**, the distribution of these random functions at initialization,

  $$p\left(f, \frac{df}{d\theta}, \frac{d^2 f}{d\theta^2}, \frac{d^3 f}{d\theta^3}\right) , \tag{0.17}$$

  will be *nearly* simple at order $1/n$, and we'll be able to work it out in full detail using perturbation theory.

- Addressing **Problem 3**, we'll be able to use a dynamical perturbation theory to tame the nonlinear training dynamics and find an analytic solution for $\theta^\star$ in a *closed form*:

  $$\theta^\star = [\theta^\star]\left(\theta, f, \frac{df}{d\theta} \,, \frac{d^2 f}{d\theta^2} \, \frac{d^3 f}{d\theta^3}; \text{ learning algorithm; training data}\right) . \tag{0.18}$$

In particular, this will make the dependence of the solution on the details of the learning algorithm transparent and manifest.

As a result, our description of the trained distribution at order $1/n$, (0.15), will be a **nearly-Gaussian distribution**.

In addition to being analytically tractable, this truncated description at order $1/n$ will satisfy our goal of computing and understanding the distribution over trained network functions $p(f^\star)$. As a consequence of incorporating the interactions between neurons, this description has a dependence on the details of the learning algorithm and, as we'll see, includes nontrivial representation learning. Thus, *qualitatively*, this effective theory at order $1/n$ corresponds much more closely to realistic neural networks than the infinite-width description, making it far more useful as a theoretically **minimal model** for understanding deep learning.

How about the quantitative correspondence? As there is a sequence of finer descriptions that we can get by computing higher-order terms in the expansion (0.14), do these terms also need to be included?

While the formalism we introduce in the book makes computing these additional terms in the $1/n$ expansion completely systematic – though perhaps somewhat tedious – an important byproduct of studying the leading correction is actually a deeper understanding of this truncation error. In particular, what we'll find is that the correct scale to compare with width n is the depth L. That is, we'll see that the relative magnitudes of the terms in the expansion (0.14) are given by the depth-to-width aspect ratio:

$$r \equiv L/n\,. \tag{0.19}$$

This lets us recast our understanding of infinite-width vs. finite-width and shallow vs. deep in the following way:

- In the strict limit $r \to 0$, the interactions between neurons turn off: the infinite-width limit (0.10) is actually a decent description. However, these networks are *not really deep*, as their relative depth is zero: $L/n = 0$.
- In the regime $0 < r \ll 1$, there are nontrivial interactions between neurons: the finite-width effective theory truncated at order $1/n$, (0.15), gives an accurate accounting of the trained network output. These networks are *effectively deep*.
- In the regime $r \gg 1$, the neurons are strongly coupled: networks will behave chaotically, and there is no effective description due to large fluctuations from instantiation to instantiation. These networks are *overly deep*.

As such, most networks of practical use actually have reasonably small depth-to-width ratios, and so our truncated description at order $1/n$, (0.15), will provide a great *quantitative* correspondence as well.[1]

[1] More precisely, there is an *optimal aspect ratio*, $r^\star$, that divides the effective regime $r \leq r^\star$ and the ineffective regime $r > r^\star$. In Appendix A, we'll estimate this optimal aspect ratio from an information-theoretic perspective. In Appendix B, we'll further show how *residual connections* can be introduced to shift the optimal aspect ratio $r^\star$ to larger values, making the formerly overly-deep networks more practically trainable as well as quantitatively describable by our effective theory approach.

From this, we see that to really describe the properties of *multilayer* neural networks, i.e. to understand *deep learning*, we need to study large-but-finite-width networks. In this way, we'll be able to find a macroscopic *effective theory* description of realistic deep neural networks.

Chapter 1

Pretraining

> *My strongest memory of the class is the very beginning, when he started, not with some deep principle of nature, or some experiment, but with a review of Gaussian integrals. Clearly, there was some calculating to be done.*
>
> Joe Polchinski, reminiscing about Richard Feynman's quantum mechanics class [5].

The goal of this book is to develop principles that enable a theoretical understanding of deep learning. Perhaps the most important principle is that wide and deep neural networks are governed by nearly-Gaussian distributions. Thus, to make it through the book, you will need to achieve mastery of Gaussian integration and perturbation theory. Our *pretraining* in this chapter consists of whirlwind introductions to these toolkits as well as a brief overview of some key concepts in statistics that we'll need. The only prerequisite is fluency in linear algebra, multivariable calculus, and rudimentary probability theory.

With that in mind, we begin in §1.1 with an extended discussion of Gaussian integrals. Our emphasis will be on calculational tools for computing averages of monomials against Gaussian distributions, culminating in a derivation of *Wick's theorem*.

Next, in §1.2, we begin by giving a general discussion of *expectation values* and *observables*. Thinking of observables as a way of learning about a probability distribution through repeated experiments, we're lead to the statistical concepts of moment and cumulant and the corresponding physicists' concepts of full M-point correlator and connected M-point correlator. A particular emphasis is placed on the connected correlators as they directly characterize a distribution's deviation from Gaussianity.

In §1.3, we introduce the negative log probability or *action* representation of a probability distribution and explain how the action lets us systematically deform Gaussian distributions in order to give a compact representation of non-Gaussian distributions. In particular, we specialize to nearly-Gaussian distributions, for which deviations from Gaussianity are implemented by small *couplings* in the action, and show how perturbation theory can be used to connect the non-Gaussian couplings to observables such as the connected correlators. By treating such couplings perturbatively, we can transform

any correlator of a nearly-Gaussian distribution into a sum of Gaussian integrals; each integral can then be evaluated by the tools we developed in §1.1. This will be one of our most important tricks, as the neural networks we'll study are all governed by nearly-Gaussian distributions, with non-Gaussian couplings that become perturbatively small as the networks become wide.

Since all these manipulations need to be on our fingertips, in this first chapter we've erred on the side of being verbose – in words and equations and examples – with the goal of making these materials as transparent and comprehensible as possible.

1.1 Gaussian Integrals

The goal of this section is to introduce Gaussian integrals and Gaussian probability distributions, and ultimately derive Wick's theorem (1.45). This theorem provides an operational formula for computing any moment of a multivariable Gaussian distribution, and will be used throughout the book.

Single-variable Gaussian integrals

Let's take it slow and start with the simplest single-variable Gaussian function,

$$e^{-\frac{z^2}{2}} \,. \tag{1.1}$$

The graph of this function depicts the famous *bell curve*, symmetric around the peak at $z = 0$ and quickly tapering off for large $|z| \gg 1$. By itself, (1.1) cannot serve as a probability distribution since it's not normalized. In order to find out the proper normalization, we need to perform the Gaussian integral

$$I_1 \equiv \int_{-\infty}^{\infty} dz \; e^{-\frac{z^2}{2}} \,. \tag{1.2}$$

As an ancient object, there exists a neat trick to evaluate such an integral. To begin, consider its square

$$I_1^2 = \left(\int_{-\infty}^{\infty} dz \; e^{-\frac{z^2}{2}} \right)^2 = \int_{-\infty}^{\infty} dx \; e^{-\frac{x^2}{2}} \int_{-\infty}^{\infty} dy \; e^{-\frac{y^2}{2}} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} dx dy \; e^{-\frac{1}{2}\left(x^2+y^2\right)} \,, \tag{1.3}$$

where in the middle we just changed the names of the dummy integration variables. Next, we change variables to polar coordinates $(x, y) = \Big(r\cos\left(\phi\right), r\sin\left(\phi\right) \Big)$, which transforms the integral measure as $dxdy = rdrd\phi$ and gives us two elementary integrals to compute:

$$\begin{aligned} I_1^2 = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} dx dy \; e^{-\frac{1}{2}\left(x^2+y^2\right)} &= \int_0^{\infty} r dr \int_0^{2\pi} d\phi \; e^{-\frac{r^2}{2}} \\ &= 2\pi \int_0^{\infty} dr \; r e^{-\frac{r^2}{2}} = 2\pi \left| -e^{-\frac{r^2}{2}} \right|_{r=0}^{r=\infty} = 2\pi \,. \end{aligned} \tag{1.4}$$

Finally, by taking a square root we can evaluate the Gaussian integral (1.2) as

$$I_1 = \int_{-\infty}^{\infty} dz\; e^{-\frac{z^2}{2}} = \sqrt{2\pi}\,. \tag{1.5}$$

Dividing the Gaussian function with this normalization factor, we define the **Gaussian probability distribution** with unit variance as

$$p(z) \equiv \frac{1}{\sqrt{2\pi}} e^{-\frac{z^2}{2}}\,, \tag{1.6}$$

which is now properly normalized, i.e., $\int_{-\infty}^{\infty} dz\, p(z) = 1$. Such a distribution with zero mean and unit variance is sometimes called the *standard normal distribution*.

Extending this result to a Gaussian distribution with **variance** $K > 0$ is super-easy. The corresponding normalization factor is given by

$$I_K \equiv \int_{-\infty}^{\infty} dz\; e^{-\frac{z^2}{2K}} = \sqrt{K} \int_{-\infty}^{\infty} du\; e^{-\frac{u^2}{2}} = \sqrt{2\pi K}\,, \tag{1.7}$$

where in the middle we rescaled the integration variable as $u = z/\sqrt{K}$. We can then define the Gaussian distribution with variance K as

$$p(z) \equiv \frac{1}{\sqrt{2\pi K}} e^{-\frac{z^2}{2K}}\,. \tag{1.8}$$

The graph of this distribution again depicts a bell curve symmetric around $z = 0$, but it's now equipped with a scale K characterizing its broadness, tapering off for $|z| \gg \sqrt{K}$. More generally, we can shift the center of the bell curve as

$$p(z) \equiv \frac{1}{\sqrt{2\pi K}} e^{-\frac{(z-s)^2}{2K}}\,, \tag{1.9}$$

so that it is now symmetric around $z = s$. This center value s is called the **mean** of the distribution, because it is:

$$\begin{aligned} \mathbb{E}\,[z] \equiv \int_{-\infty}^{\infty} dz\; p\,(z)\, z =& \frac{1}{\sqrt{2\pi K}} \int_{-\infty}^{\infty} dz\; e^{-\frac{(z-s)^2}{2K}} z \\ =& \frac{1}{I_K} \int_{-\infty}^{\infty} dw\; e^{-\frac{w^2}{2K}}\,(s+w) \\ =& \frac{s I_K}{I_K} + \frac{1}{I_K} \int_{-\infty}^{\infty} dw\; \left(e^{-\frac{w^2}{2K}} w\right) \\ =& s\,, \end{aligned} \tag{1.10}$$

where in the middle we shifted the variable as $w = z - s$ and in the very last step noticed that the integrand of the second term is odd with respect to the sign flip of the integration variable $w \leftrightarrow -w$ and hence integrates to zero.

Focusing on Gaussian distributions with zero mean, let's consider other **expectation values** for general functions $\mathcal{O}(z)$, i.e.,

$$\mathbb{E}\left[\mathcal{O}\left(z\right)\right] \equiv \int_{-\infty}^{\infty} dz\; p(z)\,\mathcal{O}(z) = \frac{1}{\sqrt{2\pi K}} \int_{-\infty}^{\infty} dz\; e^{-\frac{z^2}{2K}}\mathcal{O}(z)\;. \tag{1.11}$$

We'll often refer to such functions $\mathcal{O}(z)$ as **observables**, since they can correspond to measurement outcomes of experiments. A special class of expectation values are called **moments** and correspond to the insertion of z^M into the integrand for any integer M:

$$\mathbb{E}\left[z^M\right] = \frac{1}{\sqrt{2\pi K}} \int_{-\infty}^{\infty} dz\; e^{-\frac{z^2}{2K}} z^M\,. \tag{1.12}$$

Note that the integral vanishes for any odd exponent M, because then the integrand is odd with respect to the sign flip $z \leftrightarrow -z$. As for the even number $M = 2m$ of z insertions, we will need to evaluate integrals of the form

$$I_{K,m} \equiv \int_{-\infty}^{\infty} dz\; e^{-\frac{z^2}{2K}} z^{2m}\,. \tag{1.13}$$

As objects almost as ancient as (1.2), again there exists a neat trick to evaluate them:

$$\begin{aligned} I_{K,m} &= \int_{-\infty}^{\infty} dz\; e^{-\frac{z^2}{2K}} z^{2m} = \left(2K^2\frac{d}{dK}\right)^m \int_{-\infty}^{\infty} dz\; e^{-\frac{z^2}{2K}} = \left(2K^2\frac{d}{dK}\right)^m I_K \\ &= \left(2K^2\frac{d}{dK}\right)^m \sqrt{2\pi}K^{\frac{1}{2}} = \sqrt{2\pi}K^{\frac{2m+1}{2}}(2m-1)(2m-3)\cdots 1\,, \end{aligned} \tag{1.14}$$

where in going to the second line we substituted in our expression (1.7) for I_K. Therefore, we see that the even moments are given by the simple formula[1]

$$\mathbb{E}\left[z^{2m}\right] = \frac{I_{K,m}}{\sqrt{2\pi K}} = K^m\,(2m-1)!!\,, \tag{1.15}$$

where we have introduced the double factorial

$$(2m-1)!! \equiv (2m-1)(2m-3)\cdots 1 = \frac{(2m)!}{2^m m!}\,. \tag{1.16}$$

The result (1.15) is Wick's theorem for single-variable Gaussian distributions.

There's actually another nice way to derive (1.15), which can much more naturally be extended to multivariable Gaussian distributions. This derivation starts with the consideration of a Gaussian integral with a **source term** J, which we define as

$$Z_{K,J} \equiv \int_{-\infty}^{\infty} dz\; e^{-\frac{z^2}{2K}+Jz}\,. \tag{1.17}$$

Note that when setting the source to zero we recover the normalization of the Gaussian integral, giving the relationship $Z_{K,J=0} = I_K$. In the physics literature $Z_{K,J}$ is sometimes

[1] This equation with $2m = 2$ makes clear why we called K the variance, since for zero-mean Gaussian distributions with variance K we have $\mathrm{var}(z) \equiv \mathbb{E}\left[\left(z - \mathbb{E}\left[z\right]\right)^2\right] = \mathbb{E}\left[z^2\right] - \mathbb{E}\left[z\right]^2 = \mathbb{E}\left[z^2\right] = K$.

called a **partition function with source** and, as we will soon see, this integral serves as a **generating function** for the moments. We can evaluate $Z_{K,J}$ by completing the square in the exponent

$$-\frac{z^2}{2K}+Jz=-\frac{(z-JK)^2}{2K}+\frac{KJ^2}{2}\,, \tag{1.18}$$

which lets us rewrite the integral (1.17) as

$$Z_{K,J}=e^{\frac{KJ^2}{2}}\int_{-\infty}^{\infty}dz\;e^{-\frac{(z-JK)^2}{2K}}=e^{\frac{KJ^2}{2}}I_K=e^{\frac{KJ^2}{2}}\sqrt{2\pi K}\,, \tag{1.19}$$

where in the middle equality we noticed that the integrand is just a shifted Gaussian function with variance K.

We can now relate the Gaussian integral with a source $Z_{K,J}$ to the Gaussian integral with insertions $I_{K,m}$. By differentiating $Z_{K,J}$ with respect to the source J and *then* setting the source to zero, we observe that

$$I_{K,m}=\int_{-\infty}^{\infty}dz\;e^{-\frac{z^2}{2K}}z^{2m}=\left[\left(\frac{d}{dJ}\right)^{2m}\int_{-\infty}^{\infty}dz\;e^{-\frac{z^2}{2K}+Jz}\right]\Bigg|_{J=0}=\left[\left(\frac{d}{dJ}\right)^{2m}Z_{K,J}\right]\Bigg|_{J=0}. \tag{1.20}$$

In other words, the integrals $I_{K,m}$ are simply related to the even Taylor coefficients of the partition function $Z_{K,J}$ around $J=0$. For instance, for $2m=2$ we have

$$\mathbb{E}\left[z^2\right]=\frac{I_{K,1}}{\sqrt{2\pi K}}=\left[\left(\frac{d}{dJ}\right)^{2}e^{\frac{KJ^2}{2}}\right]\Bigg|_{J=0}=\left[e^{\frac{KJ^2}{2}}\left(K+K^2J^2\right)\right]\Bigg|_{J=0}=K\,, \tag{1.21}$$

and for $2m=4$ we have

$$\mathbb{E}\left[z^4\right]=\frac{I_{K,2}}{\sqrt{2\pi K}}=\left[\left(\frac{d}{dJ}\right)^{4}e^{\frac{KJ^2}{2}}\right]\Bigg|_{J=0}=\left[e^{\frac{KJ^2}{2}}\left(3K^2+6K^3J^2+K^4J^4\right)\right]\Bigg|_{J=0}=3K^2\,. \tag{1.22}$$

Notice that any terms with dangling sources J vanish upon setting $J=0$. This observation gives a simple way to evaluate correlators for general m: Taylor-expand the exponential $Z_{K,J}/I_K=\exp\left(\frac{KJ^2}{2}\right)$ and keep the term with the right amount of sources such that the expression doesn't vanish. Doing exactly that, we get

$$\mathbb{E}\left[z^{2m}\right]=\frac{I_{K,m}}{\sqrt{2\pi K}}=\left[\left(\frac{d}{dJ}\right)^{2m}e^{\frac{KJ^2}{2}}\right]\Bigg|_{J=0}=\left\{\left(\frac{d}{dJ}\right)^{2m}\left[\sum_{k=0}^{\infty}\frac{1}{k!}\left(\frac{K}{2}\right)^{k}J^{2k}\right]\right\}\Bigg|_{J=0} \tag{1.23}$$

$$=\left(\frac{d}{dJ}\right)^{2m}\left[\frac{1}{m!}\left(\frac{K}{2}\right)^{m}J^{2m}\right]=K^m\frac{(2m)!}{2^m m!}=K^m(2m-1)!!\,,$$

which completes our second derivation of Wick's theorem (1.15) for the single-variable Gaussian distribution. This derivation was much longer than the first neat derivation, but can be very naturally extended to the multivariable Gaussian distribution, which we turn to next.

Multivariable Gaussian integrals

Picking up speed, we are now ready to handle multivariable Gaussian integrals for an N-dimensional variable z_μ with $\mu = 1, \ldots, N$.[2] The multivariable Gaussian function is defined as

$$\exp\left[-\frac{1}{2}\sum_{\mu,\nu=1}^{N} z_\mu (K^{-1})_{\mu\nu}\, z_\nu\right], \tag{1.24}$$

where the variance or **covariance matrix** $K_{\mu\nu}$ is an N-by-N symmetric positive definite matrix, and its inverse $(K^{-1})_{\mu\nu}$ is defined so that their matrix product gives the N-by-N identity matrix

$$\sum_{\rho=1}^{N} (K^{-1})_{\mu\rho}\, K_{\rho\nu} = \delta_{\mu\nu}\,. \tag{1.25}$$

Here we have also introduced the **Kronecker delta** $\delta_{\mu\nu}$, which satisfies

$$\delta_{\mu\nu} \equiv \begin{cases} 1\,, & \mu = \nu\,, \\ 0\,, & \mu \neq \nu\,. \end{cases} \tag{1.26}$$

The Kronecker delta is just a convenient representation of the identity matrix.

Now, to construct a probability distribution from the Gaussian function (1.24), we again need to evaluate the normalization factor

$$\begin{aligned} I_K &\equiv \int d^N z \exp\left[-\frac{1}{2}\sum_{\mu,\nu=1}^{N} z_\mu (K^{-1})_{\mu\nu}\, z_\nu\right] \\ &= \int_{-\infty}^{\infty} dz_1 \int_{-\infty}^{\infty} dz_2 \cdots \int_{-\infty}^{\infty} dz_N \exp\left[-\frac{1}{2}\sum_{\mu,\nu=1}^{N} z_\mu (K^{-1})_{\mu\nu}\, z_\nu\right]. \end{aligned} \tag{1.27}$$

To compute this integral, first recall from linear algebra that, given an N-by-N symmetric matrix $K_{\mu\nu}$, there is always an orthogonal matrix[3] $O_{\mu\nu}$ that diagonalizes $K_{\mu\nu}$ as $(OKO^T)_{\mu\nu} = \lambda_\mu \delta_{\mu\nu}$ with eigenvalues $\lambda_{\mu=1,\ldots,N}$ and diagonalizes its inverse as $(OK^{-1}O^T)_{\mu\nu} = (1/\lambda_\mu)\,\delta_{\mu\nu}$. With this in mind, after twice inserting the identity matrix as $\delta_{\mu\nu} = (O^T O)_{\mu\nu}$, the sum in the exponent of the integral can be expressed in terms of the

[2] Throughout this book, we will explicitly write out the component indices of vectors, matrices, and tensors as much as possible, except on some occasions when it is clear enough from context.

[3] An *orthogonal matrix* $O_{\mu\nu}$ is a matrix whose transpose $\left(O^T\right)_{\mu\nu}$ equals its inverse, i.e., $(O^T O)_{\mu\nu} = \delta_{\mu\nu}$.

eigenvalues as

$$\begin{aligned}\sum_{\mu,\nu=1}^{N} z_\mu (K^{-1})_{\mu\nu} z_\nu &= \sum_{\mu,\rho,\sigma,\nu=1}^{N} z_\mu \, (O^T O)_{\mu\rho} (K^{-1})_{\rho\sigma} (O^T O)_{\sigma\nu} \, z_\nu \\ &= \sum_{\mu,\nu=1}^{N} (Oz)_\mu (O K^{-1} O^T)_{\mu\nu} (Oz)_\nu \\ &= \sum_{\mu=1}^{N} \frac{1}{\lambda_\mu} (Oz)_\mu^2 \,,\end{aligned} \tag{1.28}$$

where to reach the final line we used the diagonalization property of the inverse covariance matrix. Remembering that for a positive definite matrix $K_{\mu\nu}$ the eigenvalues are all positive $\lambda_\mu > 0$, we see that the λ_μ sets the scale of the falloff of the Gaussian function in each of the eigendirections. Next, recall from multivariable calculus that a change of variables $u_\mu \equiv (Oz)_\mu$ with an orthogonal matrix O leaves the integration measure invariant, i.e., $d^N z = d^N u$. All together, this lets us factorize the multivariable Gaussian integral (1.27) into a product of single-variable Gaussian integrals (1.7), yielding

$$\begin{aligned} I_K &= \int_{-\infty}^{\infty} du_1 \int_{-\infty}^{\infty} du_2 \cdots \int_{-\infty}^{\infty} du_N \; \exp\left(-\frac{u_1^2}{2\lambda_1} - \frac{u_2^2}{2\lambda_2} - \ldots - \frac{u_N^2}{2\lambda_N} \right) \\ &= \prod_{\mu=1}^{N} \left[\int_{-\infty}^{\infty} du_\mu \; \exp\left(-\frac{u_\mu^2}{2\lambda_\mu} \right) \right] = \prod_{\mu=1}^{N} \sqrt{2\pi\lambda_\mu} = \sqrt{\prod_{\mu=1}^{N} (2\pi\lambda_\mu)} \,. \end{aligned} \tag{1.29}$$

Finally, recall one last fact from linear algebra that the product of the eigenvalues of a matrix is equal to the matrix determinant. Thus, compactly, we can express the value of the multivariable Gaussian integral as

$$I_K = \int d^N z \; \exp\left[-\frac{1}{2} \sum_{\mu,\nu=1}^{N} z_\mu (K^{-1})_{\mu\nu} z_\nu \right] = \sqrt{|2\pi K|} \,, \tag{1.30}$$

where $|A|$ denotes the determinant of a square matrix A.

Having figured out the normalization factor, we can define the zero-mean **multivariable Gaussian probability distribution** with variance $K_{\mu\nu}$ as

$$p(z) = \frac{1}{\sqrt{|2\pi K|}} \exp\left[-\frac{1}{2} \sum_{\mu,\nu=1}^{N} z_\mu (K^{-1})_{\mu\nu} \, z_\nu \right] . \tag{1.31}$$

While we're at it, let us also introduce the conventions of suppressing the superscript "-1" for the inverse covariance $(K^{-1})_{\mu\nu}$, instead placing the component indices upstairs as

$$K^{\mu\nu} \equiv (K^{-1})_{\mu\nu} \,. \tag{1.32}$$

This way, we distinguish the covariance $K_{\mu\nu}$ and the inverse covariance $K^{\mu\nu}$ by whether or not component indices are lowered or raised. With this notation, inherited from

general relativity, the defining equation for the inverse covariance (1.25) is written instead as

$$\sum_{\rho=1}^{N} K^{\mu\rho} K_{\rho\nu} = \delta^{\mu}{}_{\nu}\,, \tag{1.33}$$

and the multivariable Gaussian distribution (1.31) is written as

$$p(z) = \frac{1}{\sqrt{|2\pi K|}} \exp\left(-\frac{1}{2}\sum_{\mu,\nu=1}^{N} z_\mu K^{\mu\nu} z_\nu\right). \tag{1.34}$$

Although it might take some getting used to, this notation saves us some space and saves you some handwriting pain.[4] Regardless of how it's written, the zero-mean mulitvariable Gaussian probability distribution (1.34) peaks at $z = 0$, and its falloff is direction-dependent, determined by the covariance matrix $K_{\mu\nu}$. More generally, we can shift the peak of the Gaussian distribution to s_μ

$$p(z) = \frac{1}{\sqrt{|2\pi K|}} \exp\left[-\frac{1}{2}\sum_{\mu,\nu=1}^{N} (z-s)_\mu K^{\mu\nu} (z-s)_\nu\right], \tag{1.35}$$

which defines a general multivariable Gaussian distribution with mean $\mathbb{E}[z_\mu] = s_\mu$ and covariance $K_{\mu\nu}$. This is the most general version of the Gaussian distribution.

Next, let's consider the moments of the mean-zero multivariable Gaussian distribution

$$\begin{aligned}\mathbb{E}[z_{\mu_1}\cdots z_{\mu_M}] &\equiv \int d^N z\; p(z)\, z_{\mu_1}\cdots z_{\mu_M} \\ &= \frac{1}{\sqrt{|2\pi K|}} \int d^N z\; \exp\left(-\frac{1}{2}\sum_{\mu,\nu=1}^{N} z_\mu K^{\mu\nu} z_\nu\right) z_{\mu_1}\cdots z_{\mu_M} = \frac{I_{K,(\mu_1,\ldots,\mu_M)}}{I_K}\,,\end{aligned} \tag{1.36}$$

where we introduced the mulit-variable Gaussian integrals with insertions

$$I_{K,(\mu_1,\ldots,\mu_M)} \equiv \int d^N z\; \exp\left(-\frac{1}{2}\sum_{\mu,\nu=1}^{N} z_\mu K^{\mu\nu} z_\nu\right) z_{\mu_1}\cdots z_{\mu_M}\,. \tag{1.37}$$

Following our approach in the single-variable case, let's construct the generating function for the integrals $I_{K,(\mu_1,\ldots,\mu_M)}$ by including a source term J^μ as

$$Z_{K,J} \equiv \int d^N z\; \exp\left(-\frac{1}{2}\sum_{\mu,\nu=1}^{N} z_\mu K^{\mu\nu} z_\nu + \sum_{\mu=1}^{N} J^\mu z_\mu\right). \tag{1.38}$$

[4] If you like, in your notes you can also go full general-relativistic mode and adopt *Einstein summation convention*, suppressing the summation symbol any time indices are repeated in upstair-downstair pairs. For instance, if we adopted this convention we would write the defining equation for inverse simply as $K^{\mu\rho}K_{\rho\nu} = \delta^{\mu}{}_{\nu}$ and the Gaussian function as $\exp\left(-\frac{1}{2} z_\mu K^{\mu\nu} z_\nu\right)$.

Specifically for neural networks, you might find the Einstein summation convention helpful for *sample* indices, but sometimes confusing for *neural* indices. For extra clarity, we won't adopt this convention in the text of the book, but we mention it now since we do often use such a convention to simplify our own calculations in private.

As the name suggests, differentiating the generating function $Z_{K,J}$ with respect to the source J^μ brings down a power of z_μ such that after M such differentiations we have

$$\begin{aligned}&\left[\frac{d}{dJ^{\mu_1}}\frac{d}{dJ^{\mu_2}}\cdots\frac{d}{dJ^{\mu_M}}Z_{K,J}\right]\Bigg|_{J=0} \\ &=\int d^N z\ \exp\left(-\frac{1}{2}\sum_{\mu,\nu=1}^{N} z_\mu K^{\mu\nu} z_\nu\right) z_{\mu_1}\cdots z_{\mu_M} = I_{K,(\mu_1,\ldots,\mu_M)}\,.\end{aligned} \tag{1.39}$$

So, as in the single-variable case, the Taylor coefficients of the partition function $Z_{K,J}$ expanded around $J^\mu = 0$ are simply related to the integrals with insertions $I_{K,(\mu_1,\ldots,\mu_M)}$. Therefore, if we knew a closed-form expression for $Z_{K,J}$, we could easily compute the values of the integrals $I_{K,(\mu_1,\ldots,\mu_M)}$.

To evaluate the generating function $Z_{K,J}$ in a closed form, again we follow the lead of the single-variable case and complete the square in the exponent of the integrand in (1.38) as

$$\begin{aligned}&-\frac{1}{2}\sum_{\mu,\nu=1}^{N} z_\mu K^{\mu\nu} z_\nu + \sum_{\mu=1}^{N} J^\mu z_\mu \\ &=-\frac{1}{2}\sum_{\mu,\nu=1}^{N}\left(z_\mu - \sum_{\rho=1}^{N} K_{\mu\rho}J^\rho\right)K^{\mu\nu}\left(z_\nu - \sum_{\lambda=1}^{N} K_{\nu\lambda}J^\lambda\right) + \frac{1}{2}\sum_{\mu,\nu=1}^{N} J^\mu K_{\mu\nu} J^\nu \\ &=-\frac{1}{2}\sum_{\mu,\nu=1}^{N} w_\mu K^{\mu\nu} w_\nu + \frac{1}{2}\sum_{\mu,\nu=1}^{N} J^\mu K_{\mu\nu} J^\nu\,,\end{aligned} \tag{1.40}$$

where we have introduced the shifted variable $w_\mu \equiv z_\mu - \sum_{\rho=1}^{N} K_{\mu\rho}J^\rho$. Using this substitution, the generating function can be evaluated explicitly

$$\begin{aligned}Z_{K,J} =&\exp\left(\frac{1}{2}\sum_{\mu,\nu=1}^{N} J^\mu K_{\mu\nu} J^\nu\right)\int d^N w\ \exp\left[-\frac{1}{2}\sum_{\mu,\nu=1}^{N} w_\mu K^{\mu\nu} w_\nu\right] \\ =&\sqrt{|2\pi K|}\exp\left(\frac{1}{2}\sum_{\mu,\nu=1}^{N} J^\mu K_{\mu\nu} J^\nu\right)\,,\end{aligned} \tag{1.41}$$

where at the end we used our formula for the multivariable integral I_K, (1.30). With our closed-form expression (1.41) for the generating function $Z_{K,J}$, we can compute the Gaussian integrals with insertions $I_{K,(\mu_1,\ldots,\mu_M)}$ by differentiating it, using (1.39). For an even number $M = 2m$ of insertions, we find a really nice formula

$$\begin{aligned}\mathbb{E}\left[z_{\mu_1}\cdots z_{\mu_{2m}}\right] =&\frac{I_{K,(\mu_1,\ldots,\mu_{2m})}}{I_K} = \frac{1}{I_K}\left[\frac{d}{dJ^{\mu_1}}\cdots\frac{d}{dJ^{\mu_{2m}}}Z_{K,J}\right]\Bigg|_{J=0} \\ =&\frac{1}{2^m m!}\frac{d}{dJ^{\mu_1}}\frac{d}{dJ^{\mu_2}}\cdots\frac{d}{dJ^{\mu_{2m}}}\left(\sum_{\mu,\nu=1}^{N} J^\mu K_{\mu\nu} J^\nu\right)^m\,.\end{aligned} \tag{1.42}$$

For an odd number $M = 2m + 1$ of insertions, there is dangling source upon setting $J = 0$, and so those integrals vanish. You can also see this by looking at the integrand for any odd moment and noticing that it is odd with respect to the sign flip of the integration variables $z_\mu \leftrightarrow -z_\mu$.

Now, let's take a few moments to evaluate a few moments using this formula. For $2m = 2$, we have

$$\mathbb{E}\left[z_{\mu_1} z_{\mu_2}\right] = \frac{1}{2} \frac{d}{dJ^{\mu_1}} \frac{d}{dJ^{\mu_2}} \left(\sum_{\mu,\nu=1}^{N} J^\mu K_{\mu\nu} J^\nu \right) = K_{\mu_1\mu_2}. \tag{1.43}$$

Here, there are $2! = 2$ ways to apply the product rule for derivatives and differentiate the two J's, both of which evaluate to the same expression due to the symmetry of the covariance, $K_{\mu_1\mu_2} = K_{\mu_2\mu_1}$. This expression (1.43) validates in the multivariable setting why we have been calling $K_{\mu\nu}$ the covariance, because we see explicitly that it is the covariance.

Next, for $2m = 4$ we get a more complicated expression

$$\begin{aligned} \mathbb{E}\left[z_{\mu_1} z_{\mu_2} z_{\mu_3} z_{\mu_4}\right] =& \frac{1}{2^2 2!} \frac{d}{dJ^{\mu_1}} \frac{d}{dJ^{\mu_2}} \frac{d}{dJ^{\mu_3}} \frac{d}{dJ^{\mu_4}} \left(\sum_{\mu,\nu=1}^{N} J^\mu K_{\mu\nu} J^\nu \right) \left(\sum_{\rho,\lambda=1}^{N} J^\rho K_{\rho\lambda} J^\lambda \right) \\ =& K_{\mu_1\mu_2} K_{\mu_3\mu_4} + K_{\mu_1\mu_3} K_{\mu_2\mu_4} + K_{\mu_1\mu_4} K_{\mu_2\mu_3} \,. \end{aligned} \tag{1.44}$$

Here we note that there are now $4! = 24$ ways to differentiate the four J's, though only three distinct ways to pair the four auxiliary indices $1, 2, 3, 4$ that sit under μ. This gives $24/3 = 8 = 2^2 2!$ equivalent terms for each of the three pairings, which cancels against the overall factor $1/(2^2 2!)$.

For general $2m$, there are $(2m)!$ ways to differentiate the sources, of which $2^m m!$ of those ways are equivalent. This gives $(2m)!/(2^m m!) = (2m-1)!!$ distinct terms, corresponding to the $(2m-1)!!$ distinct pairings of $2m$ auxiliary indices $1, \ldots, 2m$ that sit under μ. The factor of $1/(2^m m!)$ in the denominator of (1.42) ensures that the coefficient of each of these terms is normalized to unity. Thus, most generally, we can express the moments of the multivariable Gaussian with the following formula

$$\mathbb{E}\left[z_{\mu_1} \cdots z_{\mu_{2m}}\right] = \sum_{\text{all pairing}} K_{\mu_{k_1}\mu_{k_2}} \cdots K_{\mu_{k_{2m-1}}\mu_{k_{2m}}} \,, \tag{1.45}$$

where, to reiterate, the sum is over all the possible distinct pairings of the $2m$ auxiliary indices under μ such that the result has the $(2m-1)!!$ terms that we described above. Each factor of the covariance $K_{\mu\nu}$ in a term in sum is called a **Wick contraction**, corresponding to a particular pairing of auxiliary indices. Each term then is composed of m different Wick contractions, representing a distinct way of pairing up all the auxiliary indices. To make sure you understand how this pairing works, look back at the $2m = 2$ case (1.43) – with a single Wick contraction – and the $2m = 4$ case (1.44) – with three distinct ways of making two Wick contractions – and try to work out the $2m = 6$ case,

which yields $(6-1)!! = 15$ distinct ways of making three Wick contractions:

$$\begin{aligned}\mathbb{E}\left[z_{\mu_1}z_{\mu_2}z_{\mu_3}z_{\mu_4}z_{\mu_5}z_{\mu_6}\right] =& K_{\mu_1\mu_2}K_{\mu_3\mu_4}K_{\mu_5\mu_6} + K_{\mu_1\mu_3}K_{\mu_2\mu_4}K_{\mu_5\mu_6} + K_{\mu_1\mu_4}K_{\mu_2\mu_3}K_{\mu_5\mu_6} \\ &+K_{\mu_1\mu_2}K_{\mu_3\mu_5}K_{\mu_4\mu_6} + K_{\mu_1\mu_3}K_{\mu_2\mu_5}K_{\mu_4\mu_6} + K_{\mu_1\mu_5}K_{\mu_2\mu_3}K_{\mu_4\mu_6} \\ &+K_{\mu_1\mu_2}K_{\mu_5\mu_4}K_{\mu_3\mu_6} + K_{\mu_1\mu_5}K_{\mu_2\mu_4}K_{\mu_3\mu_6} + K_{\mu_1\mu_4}K_{\mu_2\mu_5}K_{\mu_3\mu_6} \\ &+K_{\mu_1\mu_5}K_{\mu_3\mu_4}K_{\mu_2\mu_6} + K_{\mu_1\mu_3}K_{\mu_5\mu_4}K_{\mu_2\mu_6} + K_{\mu_1\mu_4}K_{\mu_5\mu_3}K_{\mu_2\mu_6} \\ &+K_{\mu_5\mu_2}K_{\mu_3\mu_4}K_{\mu_1\mu_6} + K_{\mu_5\mu_3}K_{\mu_2\mu_4}K_{\mu_1\mu_6} + K_{\mu_5\mu_4}K_{\mu_2\mu_3}K_{\mu_1\mu_6}\,.\end{aligned}$$

The formula (1.45) is **Wick's theorem**. Put a box around it. Take a few moments for reflection.

...
...
...

Good. You are now a Gaussian sensei. Exhale, and then say as Neo would say, "I know Gaussian integrals."

Now that the moments have passed, it is an appropriate time to transition to the next section where you will learn about more general probability distributions.

1.2 Probability, Correlation and Statistics, and All That

In introducing the Gaussian distribution in the last section we briefly touched upon the concepts of expectation and moments. These are defined for non-Gaussian probability distributions too, so now let us reintroduce these concepts and expand on their definitions, with an eye towards understanding the nearly-Gaussian distributions that describe wide neural networks.

Given a **probability distribution** $p(z)$ of an N-dimensional random variable z_μ, we can learn about its statistics by measuring functions of z_μ. We'll refer to such measurable functions in a generic sense as **observables** and denote them as $\mathcal{O}(z)$. The **expectation value** of an observable

$$\mathbb{E}\left[\mathcal{O}(z)\right] \equiv \int d^N z\; p(z)\,\mathcal{O}(z) \tag{1.46}$$

characterizes the mean value of the random function $\mathcal{O}(z)$. Note that the observable $\mathcal{O}(z)$ needs not be a scalar-valued function, e.g. the second moment of a distribution is a matrix-valued observable given by $\mathcal{O}(z) = z_\mu z_\nu$.

Operationally, an observable is a quantity that we measure by conducting experiments in order to connect to a theoretical model for the underlying probability distribution describing z_μ. In particular, we repeatedly measure the observables that are naturally accessible to us as experimenters, collect their statistics, and then compare them with predictions for the expectation values of those observables computed from some theoretical model of $p(z)$.

With that in mind, it's very natural to ask: what kind of information can we learn about an underlying distribution $p(z)$ by measuring an observable $\mathcal{O}(z)$? For an a

priori unknown distribution, is there a set of observables that can serve as a sufficient probe of $p(z)$ such that we could use that information to predict the result of all future experiments involving z_μ?

Consider a class of observables that we've already encountered, the **moments** or ***M*-point correlators** of z_μ, given by the expectation[5]

$$\mathbb{E}\left[z_{\mu_1} z_{\mu_2} \cdots z_{\mu_M}\right] = \int d^N z \; p(z)\, z_{\mu_1} z_{\mu_2} \cdots z_{\mu_M} \,. \tag{1.47}$$

In principle, knowing the M-point correlators of a distribution lets us compute the expectation value of any analytic observable $\mathcal{O}(z)$ via Taylor expansion

$$\begin{aligned} \mathbb{E}\left[\mathcal{O}(z)\right] =& \mathbb{E}\left[\sum_{M=0}^{\infty} \frac{1}{M!} \sum_{\mu_1,\ldots,\mu_M=1}^{N} \left.\frac{\partial^M \mathcal{O}}{\partial z_{\mu_1} \cdots \partial z_{\mu_M}}\right|_{z=0} z_{\mu_1} z_{\mu_2} \cdots z_{\mu_M}\right] \\ =& \sum_{M=0}^{\infty} \frac{1}{M!} \sum_{\mu_1,\ldots,\mu_M=1}^{N} \left.\frac{\partial^M \mathcal{O}}{\partial z_{\mu_1} \cdots \partial z_{\mu_M}}\right|_{z=0} \mathbb{E}\left[z_{\mu_1} z_{\mu_2} \cdots z_{\mu_M}\right] \,, \end{aligned} \tag{1.48}$$

where on the last line we took the Taylor coefficients out of the expectation by using the linearity property of the expectation, inherited from the linearity property of the integral in (1.46). As such, it's clear that the collection of all the M-point correlators completely characterizes a probability distribution for all intents and purposes.[6]

However, this description in terms of all the correlators is somewhat cumbersome and operationally infeasible. To get a reliable estimate of the M-point correlator, we must simultaneously measure M components of a random variable for each draw and repeat such measurements many times. As M grows, this task quickly becomes impractical. In fact, if we could easily perform such measurements for all M, then our theoretical model of $p(z)$ would no longer be a useful abstraction; from (1.48) we would already know the outcome of all possible experiments that we could perform, leaving nothing for us to predict.

To that point, essentially all useful distributions can be effectively described in terms of a finite number of quantities, giving them a parsimonious representation. For instance, consider the zero-mean n-dimensional Gaussian distribution with the variance $K_{\mu\nu}$. The

[5] In the rest of this book, we'll often use the physics term *M-point correlator* rather than the statistics term *moment*, though they mean the same thing and can be used interchangeably.

[6] In fact, the moments offer a dual description of the probability distribution through either the Laplace transform or the Fourier transform. For instance, the Laplace transform of the probability distribution $p(z)$ is given by

$$Z_J \equiv \mathbb{E}\left[\exp\left(\sum_\mu J^\mu z_\mu\right)\right] = \int \left[\prod_\mu dz_\mu\right] \; p(z) \exp\left(\sum_\mu J^\mu z_\mu\right). \tag{1.49}$$

As in the Gaussian case, this integral gives a generating function for the M-point correlators of $p(z)$, which means that Z_J can be reconstructed from these correlators. The probability distribution can then be obtained through the inverse Laplace transform.

nonzero $2m$-point correlators are given by Wick's theorem (1.45) as

$$\mathbb{E}\left[z_{\mu_1} z_{\mu_2} \cdots z_{\mu_{2m}}\right] = \sum_{\text{all pairing}} K_{\mu_{k_1}\mu_{k_2}} \cdots K_{\mu_{k_{2m-1}}\mu_{k_{2m}}} , \tag{1.50}$$

and are determined entirely by the $N(N+1)/2$ independent components of the variance $K_{\mu\nu}$. The variance itself can be estimated by measuring the two-point correlator

$$\mathbb{E}\left[z_\mu z_\nu\right] = K_{\mu\nu} . \tag{1.51}$$

This is consistent with our description of the distribution itself as "the zero-mean N-dimensional Gaussian distribution with the variance $K_{\mu\nu}$" in which we only had to specify these same set of numbers, $K_{\mu\nu}$, to pick out the particular distribution we had in mind. For zero-mean Gaussian distributions, there's no reason to measure or keep track of any of the higher-point correlators as they are completely constrained by the variance through (1.50).

More generally, it would be nice if there were a systematic way for learning about non-Gaussian probability distributions without performing an infinite number of experiments. For nearly-Gaussian distributions, a useful set of observables is given by what statisticians call **cumulants** and physicists call **connected correlators**.[7] As the formal definition of these quantities is somewhat cumbersome and unintuitive, let's start with a few simple examples.

The first cumulant or the connected one-point correlator is the same as the full one-point correlator

$$\mathbb{E}\left[z_\mu\right]\big|_{\text{connected}} \equiv \mathbb{E}\left[z_\mu\right] . \tag{1.52}$$

This is just the *mean* of the distribution. The second cumulant or the connected two-point correlator is given by

$$\begin{aligned} \mathbb{E}\left[z_\mu z_\nu\right]\big|_{\text{connected}} \equiv& \mathbb{E}\left[z_\mu z_\nu\right] - \mathbb{E}\left[z_\mu\right]\mathbb{E}\left[z_\nu\right] \\ =& \mathbb{E}\left[\left(z_\mu - \mathbb{E}\left[z_\mu\right]\right)\left(z_\nu - \mathbb{E}\left[z_\nu\right]\right)\right] \equiv \text{Cov}[z_\mu,\, z_\nu] , \end{aligned} \tag{1.53}$$

which is also known as the *covariance* of the distribution. Note how the mean is subtracted from the random variable z_μ before taking the square in the connected version. The quantity $\widehat{\Delta z}_\mu \equiv z_\mu - \mathbb{E}\left[z_\mu\right]$ represents a **fluctuation** of the random variable around its mean. Intuitively, such fluctuations are equally likely to contribute positively as they are likely to contribute negatively, $\mathbb{E}\left[\widehat{\Delta z}_\mu\right] = \mathbb{E}\left[z_\mu\right] - \mathbb{E}\left[z_\mu\right] = 0$, so it's necessary to take the square in order to get an estimate of the magnitude of such fluctuations.

At this point, let us restrict our focus to distributions that are invariant under a sign-flip symmetry $z_\mu \to -z_\mu$, which holds for the zero-mean Gaussian distribution (1.34). Importantly, this *parity symmetry* will also hold for the nearly-Gaussian distributions

[7] Outside of this chapter, just as we'll often use the term M-point correlator rather than the term moment, we'll use the term M-point connected correlator rather than the term cumulant. When we want to refer to the moment and not the cumulant, we might sometimes say *full correlator* to contrast with *connected correlator*.

that we will study in order to describe neural networks. For all such even distributions with this symmetry, all odd moments and all odd-point connected correlators vanish.

With this restriction, the next simplest observable is the fourth cumulant or the connected four-point correlator, given by the formula

$$\begin{aligned} &\mathbb{E}\left[z_{\mu_1} z_{\mu_2} z_{\mu_3} z_{\mu_4}\right]\big|_{\text{connected}} \\ =&\mathbb{E}\left[z_{\mu_1} z_{\mu_2} z_{\mu_3} z_{\mu_4}\right] \\ &- \mathbb{E}\left[z_{\mu_1} z_{\mu_2}\right]\mathbb{E}\left[z_{\mu_3} z_{\mu_4}\right] - \mathbb{E}\left[z_{\mu_1} z_{\mu_3}\right]\mathbb{E}\left[z_{\mu_2} z_{\mu_4}\right] - \mathbb{E}\left[z_{\mu_1} z_{\mu_4}\right]\mathbb{E}\left[z_{\mu_2} z_{\mu_3}\right] . \end{aligned} \tag{1.54}$$

For the Gaussian distribution, recalling the Wick theorem (1.50), the last three terms precisely subtract off the three pairs of Wick contractions used to evaluate the first term, meaning

$$\mathbb{E}\left[z_{\mu_1} z_{\mu_2} z_{\mu_3} z_{\mu_4}\right]\big|_{\text{connected}} = 0. \tag{1.55}$$

Essentially by design, the connected four-point correlator vanishes for the Gaussian distribution, and a nonzero value signifies a deviation from Gaussian statistics.[8] In fact, the connected four-point correlator is perhaps the simplest measure of non-Gaussianity.

Now that we have a little intuition, we are as ready as we'll ever be to discuss the definition for the M-th cumulant or the M-point connected correlator. For completeness, we'll give the general definition, before restricting again to distributions that are symmetric under parity $z_\mu \to -z_\mu$. The definition is *inductive* and somewhat counterintuitive, expressing the M-th moment in terms of connected correlators from degree 1 to M:

$$\begin{aligned} &\mathbb{E}\left[z_{\mu_1} z_{\mu_2} \cdots z_{\mu_M}\right] \\ \equiv&\mathbb{E}\left[z_{\mu_1} z_{\mu_2} \cdots z_{\mu_M}\right]\big|_{\text{connected}} \\ &+ \sum_{\text{all subdivisions}} \mathbb{E}\left[z_{\mu_{k_1^{[1]}}} \cdots z_{\mu_{k_{\nu_1}^{[1]}}}\right]\Big|_{\text{connected}} \cdots \mathbb{E}\left[z_{\mu_{k_1^{[s]}}} \cdots z_{\mu_{k_{\nu_s}^{[s]}}}\right]\Big|_{\text{connected}} , \end{aligned} \tag{1.56}$$

where the sum is over all the possible subdivisions of M variables into $s > 1$ clusters of sizes $(\nu_1, \ldots, \nu_s)$ as $(k_1^{[1]}, \ldots, k_{\nu_1}^{[1]}), \ldots, (k_1^{[s]}, \ldots, k_{\nu_s}^{[s]})$. By decomposing the M-th moment into a sum of products of connected correlators of degree M and lower, we see that the connected M-point correlator corresponds to a *new* type of correlation that cannot be expressed by the connected correlators of a lower degree. We saw an example of this above when discussing the connected four-point correlator as a simple measure of non-Gaussianity.

To see how this abstract definition actually works, let's revisit the examples. First, we trivially recover the relation between the mean and the one-point connected correlator

$$\mathbb{E}\left[z_\mu\right]\big|_{\text{connected}} = \mathbb{E}\left[z_\mu\right] , \tag{1.57}$$

[8] In statistics, the connected four-point correlator for a single random variable z is called the *excess kurtosis* when normalized by the square of the variance. It is a natural measure of the tails of the distribution, as compared to a Gaussian distribution, and also serves as a measure of the potential for outliers. In particular, a positive value indicates fatter tails while a negative value indicates thinner tails.

as there is no subdivision of a $M = 1$ variable into any smaller pieces. For $M = 2$, the definition (1.56) gives

$$\begin{aligned}\mathbb{E}\left[z_{\mu_1} z_{\mu_2}\right] =& \mathbb{E}\left[z_{\mu_1} z_{\mu_2}\right]\big|_{\text{connected}} + \mathbb{E}\left[z_{\mu_1}\right]\big|_{\text{connected}} \mathbb{E}\left[z_{\mu_2}\right]\big|_{\text{connected}} \\ =& \mathbb{E}\left[z_{\mu_1} z_{\mu_2}\right]\big|_{\text{connected}} + \mathbb{E}\left[z_{\mu_1}\right]\mathbb{E}\left[z_{\mu_2}\right] \ .\end{aligned} \tag{1.58}$$

Rearranging to solve for the connected two-point function in terms of the moments, we see that this is equivalent to our previous definition for the covariance (1.53).

At this point, let us again restrict to parity-symmetric distributions invariant under $z_\mu \to -z_\mu$, remembering that this means that all the odd-point connected correlators will vanish. For such distributions, evaluating the definition (1.56) for $M = 4$ gives

$$\begin{aligned}\mathbb{E}\left[z_{\mu_1} z_{\mu_2} z_{\mu_3} z_{\mu_4}\right] =& \mathbb{E}\left[z_{\mu_1} z_{\mu_2} z_{\mu_3} z_{\mu_4}\right]\big|_{\text{connected}} \\ &+ \mathbb{E}\left[z_{\mu_1} z_{\mu_2}\right]\big|_{\text{connected}} \mathbb{E}\left[z_{\mu_3} z_{\mu_4}\right]\big|_{\text{connected}} \\ &+ \mathbb{E}\left[z_{\mu_1} z_{\mu_3}\right]\big|_{\text{connected}} \mathbb{E}\left[z_{\mu_2} z_{\mu_4}\right]\big|_{\text{connected}} \\ &+ \mathbb{E}\left[z_{\mu_1} z_{\mu_4}\right]\big|_{\text{connected}} \mathbb{E}\left[z_{\mu_2} z_{\mu_3}\right]\big|_{\text{connected}} \ .\end{aligned} \tag{1.59}$$

Since $\mathbb{E}\left[z_{\mu_1} z_{\mu_2}\right] = \mathbb{E}\left[z_{\mu_1} z_{\mu_2}\right]\big|_{\text{connected}}$ when the mean vanishes, this is also just a rearrangement of our previous expression (1.54) for the connected four-point correlator for such zero-mean distributions.

In order to see something new, let us carry on for $M = 6$:

$$\begin{aligned}\mathbb{E}\left[z_{\mu_1} z_{\mu_2} z_{\mu_3} z_{\mu_4} z_{\mu_5} z_{\mu_6}\right] =& \mathbb{E}\left[z_{\mu_1} z_{\mu_2} z_{\mu_3} z_{\mu_4} z_{\mu_5} z_{\mu_6}\right]\big|_{\text{connected}} \\ &+ \mathbb{E}\left[z_{\mu_1} z_{\mu_2}\right]\big|_{\text{connected}} \mathbb{E}\left[z_{\mu_3} z_{\mu_4}\right]\big|_{\text{connected}} \mathbb{E}\left[z_{\mu_5} z_{\mu_6}\right]\big|_{\text{connected}} \\ &+ [14 \text{ other } (2,2,2) \text{ subdivisions}] \\ &+ \mathbb{E}\left[z_{\mu_1} z_{\mu_2} z_{\mu_3} z_{\mu_4}\right]\big|_{\text{connected}} \mathbb{E}\left[z_{\mu_5} z_{\mu_6}\right]\big|_{\text{connected}} \\ &+ [14 \text{ other } (4,2) \text{ subdivisions}] \ ,\end{aligned} \tag{1.60}$$

in which we have expressed the full six-point correlator in terms of a sum of products of connected two-point, four-point, and six-point correlators. Rearranging the above expression and expressing the two-point and four-point connected correlators in terms of their definitions, (1.53) and (1.54), we obtain an expression for the connected six-point correlator:

$$\begin{aligned}&\mathbb{E}\left[z_{\mu_1} z_{\mu_2} z_{\mu_3} z_{\mu_4} z_{\mu_5} z_{\mu_6}\right]\big|_{\text{connected}} \\ =& \mathbb{E}\left[z_{\mu_1} z_{\mu_2} z_{\mu_3} z_{\mu_4} z_{\mu_5} z_{\mu_6}\right] \\ &- \left\{\mathbb{E}\left[z_{\mu_1} z_{\mu_2} z_{\mu_3} z_{\mu_4}\right]\mathbb{E}\left[z_{\mu_5} z_{\mu_6}\right] + [14 \text{ other } (4,2) \text{ subdivisions}]\right\} \\ &+ 2\left\{\mathbb{E}\left[z_{\mu_1} z_{\mu_2}\right]\mathbb{E}\left[z_{\mu_3} z_{\mu_4}\right]\mathbb{E}\left[z_{\mu_5} z_{\mu_6}\right] + [14 \text{ other } (2,2,2) \text{ subdivisions}]\right\} \ .\end{aligned} \tag{1.61}$$

The rearrangement is useful for computational purposes, in that it's simple to first compute the moments of a distribution and then organize the resulting expressions in order to evaluate the connected correlators.

Focusing back on (1.60), it's easy to see that the connected six-point correlator vanishes for Gaussian distributions. Remembering that the connected four-point correlator also vanishes for Gaussian distributions, we see that the fifteen $(2, 2, 2)$ subdivision terms are exactly equal to the fifteen terms generated by the Wick contractions resulting from evaluating the full correlator on the left-hand side of the equation. In fact, applying the general definition of connected correlators (1.56) to the zero-mean Gaussian distribution, we see inductively that all M-point connected correlators for $M > 2$ will vanish.[9] Thus, the connected correlators are a very natural measure of how a distribution deviates from Gaussianity.

With this in mind, we can finally define a **nearly-Gaussian distribution** as a distribution for which all the connected correlators for $M > 2$ are *small*.[10] In fact, the non-Gaussian distributions that describe neural networks generally have the property that, as the network becomes wide, the connected four-point correlator becomes small and the higher-point connected correlators become even smaller. For these nearly-Gaussian distributions, a few leading connected correlators give a concise and accurate description of the distribution, just as a few leading Taylor coefficients can give a good description of a function near the point of expansion.

1.3 Nearly-Gaussian Distributions

Now that we have defined nearly-Gaussian distributions in terms of measurable deviations from Gaussian statistics, i.e. via small but nonzero connected correlators, it's natural to ask how we can link these observables to the actual functional form of the distribution, $p(z)$. We can make this connection through the action.

The **action** $S(z)$ is a function that defines a probability distribution $p(z)$ through the relation

$$p(z) \propto e^{-S(z)} \,. \tag{1.62}$$

In the statistics literature, the action $S(z)$ is sometimes called the *negative log probability*, but we will again follow the physics literature and call it the action. In order for (1.62) to make sense as a probability distribution, $p(z)$ needs be normalizable so that we can satisfy

$$\int d^N z \; p(z) = 1 \,. \tag{1.63}$$

That's where the *normalization factor* or **partition function**

$$Z \equiv \int d^N z \; e^{-S(z)} \tag{1.64}$$

[9] To see this, note that if all the higher-point connected correlators vanish, then the definition (1.56) is equivalent to Wick's theorem (1.50), with nonzero terms in (1.56) – the subdivisions into clusters of sizes $(2, \ldots, 2)$ – corresponding exactly to the different pairings in (1.50).

[10] As we discussed in §1.1, the variance sets the scale of the Gaussian distribution. For nearly-Gaussian distributions, we require that all $2m$-point connected correlators be parametrically small when compared to an appropriate power of the variance, i.e., $|\mathbb{E}\left[z_{\mu_1} \cdots z_{\mu_{2m}}\right]|_{\text{connected}}| \ll |K_{\mu\nu}|^m$, schematically.

comes in. After computing the partition function, we can define a probability distribution for a particular action $S(z)$ as

$$p(z) \equiv \frac{e^{-S(z)}}{Z} \,. \tag{1.65}$$

Conversely, given a probability distribution we can associate an action, $S(z) = -\log[p(z)]$, up to an additive ambiguity: the ambiguity arises because a constant shift in the action can be offset by the multiplicative factor in the partition function.[11]

The action is a very convenient way to approximate certain types of statistical processes, particularly those with nearly-Gaussian statistics. To demonstrate this, we'll first start with the simplest action, which describes the Gaussian distribution, and then we'll show how to systematically perturb it in order to include various non-Gaussianities.

Quadratic action and the Gaussian distribution

Since we already know the functional form of the Gaussian distribution, it's simple to identify the action by reading it off from the exponent in (1.34)

$$S(z) = \frac{1}{2} \sum_{\mu,\nu=1}^{N} K^{\mu\nu} z_\mu z_\nu \,, \tag{1.66}$$

where, as a reminder, the matrix $K^{\mu\nu}$ is the inverse of the variance matrix $K_{\mu\nu}$. The partition function is given by the normalization integral (1.30) that we computed in §1.1

$$Z = \int d^N z \; e^{-S(z)} = I_K = \sqrt{|2\pi K|} \,. \tag{1.67}$$

This **quadratic action** is the simplest normalizable action and serves as a starting point for defining other distributions.

As we will show next, integrals against the Gaussian distribution are a primitive for evaluating expectations against nearly-Gaussian distributions. Therefore, in order to differentiate between a general expectation and an integral against the Gaussian distribution, let us introduce a special *bra-ket*, or $\langle\cdot\rangle$ notation for computing *Gaussian* expectation values. For an observable $\mathcal{O}(z)$, define the **Gaussian expectation** as

$$\langle \mathcal{O}(z) \rangle_K \equiv \frac{1}{\sqrt{|2\pi K|}} \int \left[\prod_{\mu=1}^{N} dz_\mu \right] \exp\left(-\frac{1}{2} \sum_{\mu,\nu=1}^{N} K^{\mu\nu} z_\mu z_\nu \right) \mathcal{O}(z) \,. \tag{1.68}$$

In particular, with this notation we can write Wick's theorem as

$$\langle z_{\mu_1} z_{\mu_2} \cdots z_{\mu_{2m}} \rangle_K = \sum_{\text{all pairing}} K_{\mu_{k_1}\mu_{k_2}} \cdots K_{\mu_{k_{2m-1}}\mu_{k_{2m}}} \,. \tag{1.69}$$

If we're talking about a Gaussian distribution with variance $K_{\mu\nu}$, then we can use the notation $\mathbb{E}[\,\cdot\,]$ and $\langle\cdot\rangle_K$ interchangeably. If instead we're talking about a nearly-Gaussian

[11] One convention is to pick the constant such that the action vanishes when evaluated at its global minimum.

distribution $p(z)$, then $\mathbb{E}[\,\cdot\,]$ indicates expectation with respect to $p(z)$, (1.46). However, in the evaluation of such an expectation, we'll often encounter Gaussian integrals, for which we'll use this bra-ket notation $\langle\cdot\rangle_K$ to simplify expressions.

Quartic action and perturbation theory

Now, let's find an action that represents a nearly-Gaussian distribution with a connected four-point correlator that is *small* but non-vanishing

$$\mathbb{E}[z_{\mu_1}z_{\mu_2}z_{\mu_3}z_{\mu_4}]\big|_{\text{connected}} = O(\epsilon)\ . \tag{1.70}$$

Here we have introduced a small parameter $\epsilon \ll 1$ and indicated that the correlator should be of order ϵ. For neural networks, we will later find that the role of the small parameter ϵ is played by 1/width.

We should be able to generate a small connected four-point correlator by *deforming* the Gaussian distribution through the addition of a small quartic term to the quadratic action (1.66), giving us a **quartic action**

$$S(z) = \frac{1}{2}\sum_{\mu,\nu=1}^{N} K^{\mu\nu}z_\mu z_\nu + \frac{\epsilon}{4!}\sum_{\mu,\nu,\rho,\lambda=1}^{N} V^{\mu\nu\rho\lambda}z_\mu z_\nu z_\rho z_\lambda\,, \tag{1.71}$$

where the **quartic coupling** $\epsilon V^{\mu\nu\rho\lambda}$ is an $(N\times N\times N\times N)$-dimensional **tensor** that is completely symmetric in all of its four indices. The factor of $1/4!$ is conventional in order to compensate for the overcounting in the sum due to the symmetry of the indices. While it's not a proof of the connection, note that the coupling $\epsilon V^{\mu\nu\rho\lambda}$ has the right number of components to faithfully reproduce the four-point connected correlator (1.70), which is also an $(N\times N\times N\times N)$-dimensional symmetric tensor. At least from this perspective we're off to a good start.

Let us now establish this correspondence between the quartic coupling and connected four-point correlator. Note that in general it is impossible to compute any expectation value in closed form with a non-Gaussian action – this includes even the partition function. Instead, in order to compute the connected four-point correlator we'll need to employ **perturbation theory** to expand everything to first order in the small parameter ϵ, each term of which can then be evaluated in a closed form. As this is easier done than said, let's get to the computations.

To start, let's evaluate the partition function:

$$\begin{aligned} Z &= \int\left[\prod_\mu dz_\mu\right]\ e^{-S(z)} \\ &= \int\left[\prod_\mu dz_\mu\right]\exp\left(-\frac{1}{2}\sum_{\mu,\nu}K^{\mu\nu}z_\mu z_\nu - \frac{\epsilon}{24}\sum_{\rho_1,\ldots,\rho_4}V^{\rho_1\rho_2\rho_3\rho_4}z_{\rho_1}z_{\rho_2}z_{\rho_3}z_{\rho_4}\right) \\ &= \sqrt{|2\pi K|}\left\langle\exp\left(-\frac{\epsilon}{24}\sum_{\rho_1,\ldots,\rho_4}V^{\rho_1\rho_2\rho_3\rho_4}z_{\rho_1}z_{\rho_2}z_{\rho_3}z_{\rho_4}\right)\right\rangle_K\ . \end{aligned} \tag{1.72}$$

In the second line we inserted our expression for the quartic action (1.71), and in the last line we used our bra-ket notation (1.68) for a Gaussian expectation with variance $K_{\mu\nu}$. As advertised, the Gaussian expectation in the final line cannot be evaluated in closed form. However, since our parameter ϵ is small, we can Taylor-expand the exponential to express the partition function as a sum of simple Gaussian expectations that can be evaluated using Wick's theorem (1.69):

$$\begin{aligned} Z =& \sqrt{|2\pi K|} \left\langle 1 - \frac{\epsilon}{24} \sum_{\rho_1,\ldots,\rho_4} V^{\rho_1\rho_2\rho_3\rho_4} z_{\rho_1} z_{\rho_2} z_{\rho_3} z_{\rho_4} + O\Big(\epsilon^2\Big) \right\rangle_K \\ =& \sqrt{|2\pi K|} \left[1 - \frac{\epsilon}{24} \sum_{\rho_1,\ldots,\rho_4} V^{\rho_1\rho_2\rho_3\rho_4} \left\langle z_{\rho_1} z_{\rho_2} z_{\rho_3} z_{\rho_4} \right\rangle_K + O\Big(\epsilon^2\Big) \right] \\ =& \sqrt{|2\pi K|} \left[1 - \frac{\epsilon}{24} \sum_{\rho_1,\ldots,\rho_4} V^{\rho_1\rho_2\rho_3\rho_4} \left(K_{\rho_1\rho_2} K_{\rho_3\rho_4} + K_{\rho_1\rho_3} K_{\rho_2\rho_4} + K_{\rho_1\rho_4} K_{\rho_2\rho_3} \right) + O\Big(\epsilon^2\Big) \right] \\ =& \sqrt{|2\pi K|} \left[1 - \frac{1}{8}\epsilon \sum_{\rho_1,\ldots,\rho_4} V^{\rho_1\rho_2\rho_3\rho_4} K_{\rho_1\rho_2} K_{\rho_3\rho_4} + O\Big(\epsilon^2\Big) \right] . \end{aligned} \tag{1.73}$$

In the final line, we were able to combine the three K^2 terms together by using the total symmetry of the quartic coupling and then relabeling some of the summed-over dummy indices.

Similarly, let's evaluate the two-point correlator:

$$\begin{aligned} \mathbb{E}\left[z_{\mu_1} z_{\mu_2}\right] =& \frac{1}{Z} \int \left[\prod_\mu dz_\mu \right] \; e^{-S(z)} \, z_{\mu_1} z_{\mu_2} \\ =& \frac{\sqrt{|2\pi K|}}{Z} \left\langle z_{\mu_1} z_{\mu_2} \exp\left(-\frac{\epsilon}{24} \sum_{\rho_1,\ldots,\rho_4} V^{\rho_1\rho_2\rho_3\rho_4} z_{\rho_1} z_{\rho_2} z_{\rho_3} z_{\rho_4} \right) \right\rangle_K \\ =& \frac{\sqrt{|2\pi K|}}{Z} \left[\left\langle z_{\mu_1} z_{\mu_2} \right\rangle_K - \frac{\epsilon}{24} \sum_{\rho_1,\ldots,\rho_4} V^{\rho_1\rho_2\rho_3\rho_4} \left\langle z_{\mu_1} z_{\mu_2} z_{\rho_1} z_{\rho_2} z_{\rho_3} z_{\rho_4} \right\rangle_K + O\Big(\epsilon^2\Big) \right] \\ =& \left[1 + \frac{1}{8}\epsilon \sum_{\rho_1,\ldots,\rho_4} V^{\rho_1\rho_2\rho_3\rho_4} K_{\rho_1\rho_2} K_{\rho_3\rho_4} \right] K_{\mu_1\mu_2} \\ & - \frac{\epsilon}{24} \sum_{\rho_1,\ldots,\rho_4} V^{\rho_1\rho_2\rho_3\rho_4} \left(3 K_{\mu_1\mu_2} K_{\rho_1\rho_2} K_{\rho_3\rho_4} + 12 K_{\mu_1\rho_1} K_{\mu_2\rho_2} K_{\rho_3\rho_4} \right) + O\Big(\epsilon^2\Big) \\ =& K_{\mu_1\mu_2} - \frac{\epsilon}{2} \sum_{\rho_1,\ldots,\rho_4} V^{\rho_1\rho_2\rho_3\rho_4} K_{\mu_1\rho_1} K_{\mu_2\rho_2} K_{\rho_3\rho_4} + O\Big(\epsilon^2\Big) . \end{aligned} \tag{1.74}$$

Here, to go from the first line to the second line we inserted our expression for the quartic action (1.71) and rewrote the integral as a Gaussian expectation. Then, after expanding in ϵ to first order, in the next step we substituted (1.73) in for the partition function Z in the denominator and expanded $1/Z$ to the first order in ϵ using the expansion $1/(1-x) = 1 + x + O(x^2)$. In that same step, we also noted that, of the fifteen terms coming from the Gaussian expectation $\left\langle z_{\mu_1} z_{\mu_2} z_{\rho_1} z_{\rho_2} z_{\rho_3} z_{\rho_4} \right\rangle_K$, there are three ways in

which z_{μ_1} and z_{μ_2} contract with each other but twelve ways in which they don't. Given again the symmetry of $V^{\rho_1\rho_2\rho_3\rho_4}$, this is the only distinction that matters.

At last, let's compute the full four-point correlator:

$$\begin{aligned}
&\mathbb{E}\left[z_{\mu_1}z_{\mu_2}z_{\mu_3}z_{\mu_4}\right] = \frac{1}{Z}\int\left[\prod_\mu dz_\mu\right]\; e^{-S(z)}\, z_{\mu_1}z_{\mu_2}z_{\mu_3}z_{\mu_4} \qquad (1.75)\\
=&\frac{\sqrt{|2\pi K|}}{Z}\left[\left\langle z_{\mu_1}z_{\mu_2}z_{\mu_3}z_{\mu_4}\right\rangle_K - \frac{\epsilon}{24}\sum_{\rho_1,\ldots,\rho_4} V^{\rho_1\rho_2\rho_3\rho_4}\left\langle z_{\mu_1}z_{\mu_2}z_{\mu_3}z_{\mu_4}z_{\rho_1}z_{\rho_2}z_{\rho_3}z_{\rho_4}\right\rangle_K + O\Big(\epsilon^2\Big)\right]\\
=&\left[1+\frac{1}{8}\epsilon\sum_{\rho_1,\ldots,\rho_4} V^{\rho_1\rho_2\rho_3\rho_4}K_{\rho_1\rho_2}K_{\rho_3\rho_4}\right]\left[K_{\mu_1\mu_2}K_{\mu_3\mu_4}+K_{\mu_1\mu_3}K_{\mu_2\mu_4}+K_{\mu_1\mu_4}K_{\mu_2\mu_3}\right]\\
&-\frac{\epsilon}{24}\sum_{\rho_1,\ldots,\rho_4} V^{\rho_1\rho_2\rho_3\rho_4}\\
&\times\Big(3K_{\mu_1\mu_2}K_{\mu_3\mu_4}K_{\rho_1\rho_2}K_{\rho_3\rho_4}+12K_{\mu_1\rho_1}K_{\mu_2\rho_2}K_{\mu_3\mu_4}K_{\rho_3\rho_4}+12K_{\mu_3\rho_1}K_{\mu_4\rho_2}K_{\mu_1\mu_2}K_{\rho_3\rho_4}\\
&\quad+3K_{\mu_1\mu_3}K_{\mu_2\mu_4}K_{\rho_1\rho_2}K_{\rho_3\rho_4}+12K_{\mu_1\rho_1}K_{\mu_3\rho_2}K_{\mu_2\mu_4}K_{\rho_3\rho_4}+12K_{\mu_2\rho_1}K_{\mu_4\rho_2}K_{\mu_1\mu_3}K_{\rho_3\rho_4}\\
&\quad+3K_{\mu_1\mu_4}K_{\mu_2\mu_3}K_{\rho_1\rho_2}K_{\rho_3\rho_4}+12K_{\mu_1\rho_1}K_{\mu_4\rho_2}K_{\mu_2\mu_3}K_{\rho_3\rho_4}+12K_{\mu_2\rho_1}K_{\mu_3\rho_2}K_{\mu_1\mu_4}K_{\rho_3\rho_4}\\
&\quad+24K_{\mu_1\rho_1}K_{\mu_2\rho_2}K_{\mu_3\rho_3}K_{\mu_4\rho_4}\Big)+O\Big(\epsilon^2\Big)\;.
\end{aligned}$$

To go from the first line to the second line we inserted our expression for the quartic action (1.71), expanded to first order in ϵ, and rewrote in the bra-ket notation (1.68). On the third line, we again substituted in the expression (1.73) for the partition function Z, expanded $1/Z$ to first order in ϵ, and then used Wick's theorem (1.69) to evaluate the fourth and eighth Gaussian moments. (Yes, we know that the evaluation of $\left\langle z_{\mu_1}z_{\mu_2}z_{\mu_3}z_{\mu_4}z_{\rho_1}z_{\rho_2}z_{\rho_3}z_{\rho_4}\right\rangle_K$ is not fun. The breakdown of the terms depends again on whether or not the μ-type indices are contracted with the ρ-type indices or not.) We can simplify this expression by noticing that some terms cancel due to $\frac{1}{8}-\frac{3}{24}=0$ and some other terms can be nicely regrouped once we notice through the expression for the two-point correlator (1.74) that

$$\begin{aligned}
&K_{\mu_1\mu_2}K_{\mu_3\mu_4}-\frac{\epsilon}{24}\sum_{\rho_1,\ldots,\rho_4} V^{\rho_1\rho_2\rho_3\rho_4}\left(12K_{\mu_1\rho_1}K_{\mu_2\rho_2}K_{\mu_3\mu_4}K_{\rho_3\rho_4}+12K_{\mu_3\rho_1}K_{\mu_4\rho_2}K_{\mu_1\mu_2}K_{\rho_3\rho_4}\right)\\
&=\mathbb{E}\left[z_{\mu_1}z_{\mu_2}\right]\mathbb{E}\left[z_{\mu_3}z_{\mu_4}\right]+O\Big(\epsilon^2\Big)\;, \qquad (1.76)
\end{aligned}$$

yielding in the end

$$\begin{aligned}
&\mathbb{E}\left[z_{\mu_1}z_{\mu_2}z_{\mu_3}z_{\mu_4}\right] \qquad (1.77)\\
=&\mathbb{E}\left[z_{\mu_1}z_{\mu_2}\right]\mathbb{E}\left[z_{\mu_3}z_{\mu_4}\right]+\mathbb{E}\left[z_{\mu_1}z_{\mu_3}\right]\mathbb{E}\left[z_{\mu_2}z_{\mu_4}\right]+\mathbb{E}\left[z_{\mu_1}z_{\mu_4}\right]\mathbb{E}\left[z_{\mu_2}z_{\mu_3}\right]\\
&-\epsilon\sum_{\rho_1,\ldots,\rho_4} V^{\rho_1\rho_2\rho_3\rho_4}K_{\mu_1\rho_1}K_{\mu_2\rho_2}K_{\mu_3\rho_3}K_{\mu_4\rho_4}+O\Big(\epsilon^2\Big)\;.
\end{aligned}$$

Given the full four-point correlator (1.75) and the two-point correlator (1.74), we can finally evaluate the connected four-point correlator (1.54) as

$$\mathbb{E}\left[z_{\mu_1} z_{\mu_2} z_{\mu_3} z_{\mu_4}\right]\big|_{\text{connected}} = -\epsilon \sum_{\rho_1,\ldots,\rho_4} V^{\rho_1\rho_2\rho_3\rho_4} K_{\mu_1\rho_1} K_{\mu_2\rho_2} K_{\mu_3\rho_3} K_{\mu_4\rho_4} + O\left(\epsilon^2\right) . \tag{1.78}$$

This makes explicit the relationship between the connected four-point correlator and the quartic coupling in the action, when both are small. We see that for the nearly-Gaussian distribution realized by the quartic action (1.71), the distribution is – as promised – *nearly* Gaussian: the strength of the coupling $\epsilon V^{\rho_1\rho_2\rho_3\rho_4}$ directly controls the distribution's deviation from Gaussian statistics, as measured by the connected four-point correlator. This also shows that the four-index tensor $V^{\rho_1\rho_2\rho_3\rho_4}$ creates nontrivial correlations between the components $z_{\rho_1} z_{\rho_2} z_{\rho_3} z_{\rho_4}$ that cannot otherwise be built up by the correlation $K_{\mu\nu}$ in any pair of random variables $z_\mu z_\nu$.

Finally, note that the connected two-point correlator (1.74) – i.e. the covariance of this nearly-Gaussian distribution – is also shifted from its Gaussian value of $K_{\mu_1\mu_2}$ by the quartic coupling $\epsilon V^{\rho_1\rho_2\rho_3\rho_4}$. Thus, the nearly-Gaussian deformation not only creates complicated patterns of four-point correlation as measured by the connected four-point correlator (1.78), it also can modify the details of the Gaussian two-point correlation.

Now that we see how to compute the statistics of a nearly-Gaussian distribution, let's take a step back and think about what made this possible. We can perform these perturbative calculations any time there exists in the problem a dimensionless parameter ϵ that is small $\epsilon \ll 1$, but nonzero $\epsilon > 0$. This makes perturbation theory an extremely powerful tool for theoretical analysis any time a problem has any extreme scales, small *or* large.

Importantly, this is directly relevant to theoretically understanding neural networks in practice. As we will explain in the following chapters, real networks have a parameter n – the number of neurons in a layer – that is typically large $n \gg 1$, but certainly not infinite $n < \infty$. This means that we can expand the distributions that describe such networks in the inverse of the large parameter as $\epsilon = 1/n$. Indeed, when the parameter n is large – as is typical in practice – the distributions that describe neural networks become nearly-Gaussian and thus theoretically tractable. This type of expansion is known as the **1/n expansion** or **large-n expansion** and will be one of our main tools for learning the principles of deep learning theory.

Aside: statistical independence and interactions

The quartic action (1.71) is one of the simplest models of an **interacting theory**. We showed this explicitly by connecting the quartic coupling to the non-Gaussian statistics of the non-vanishing connected four-point correlator. Here, let us try to offer an intuitive meaning of *interaction* by appealing to the notion of *statistical independence*.

Recall from the probability theory that two random variables x and y are statistically independent if their joint distribution factorizes as

$$p(x, y) = p(x)p(y) . \tag{1.79}$$

For the Gaussian distribution, if the variance matrix $K_{\mu\nu}$ is diagonal, there is no correlation at all between different components of z_μ; they are manifestly statistically independent from each other.

Even if $K_{\mu\nu}$ is not diagonal, we can still unwind the correlation of a Gaussian distribution by rotating to the right basis. As discussed in §1.1, there always exists an orthogonal matrix O that diagonalizes the covariance as $(OKO^T)_{\mu\nu} = \lambda_\mu \delta_{\mu\nu}$. In terms of the variables $u_\mu \equiv (Oz)_\mu$, the distribution looks like

$$p(z) = \frac{1}{\sqrt{|2\pi K|}} \exp\left(-\sum_{\mu=1}^{N} \frac{u_\mu^2}{2\lambda_\mu}\right) = \prod_{\mu=1}^{N} \left(\frac{e^{-\frac{u_\mu^2}{2\lambda_\mu}}}{\sqrt{2\pi\lambda_\mu}}\right) = p\,(u_1)\cdots p(u_N)\,. \tag{1.80}$$

Thus, we see that in the u-coordinate basis the original multivariable Gaussian distribution factorizes into N single-variable Gaussians that are statistically independent.

We also see that in terms of the action, statistical independence is characterized by the action breaking into a sum over separate terms. This unwinding of interaction between variables is generically impossible when there are nonzero non-Gaussian couplings. For instance, there are $\sim N^2$ components of an orthogonal matrix $O_{\mu\nu}$ to change basis, while there are $\sim N^4$ components of the quartic coupling $\epsilon V^{\mu\nu\rho\lambda}$ that correlate random variables, so it is generically impossible to re-express the quartic action as a sum of functions of N different variables. Since the action cannot be put into a sum over N separate terms, the joint distribution cannot factorize, and the components will not be independent from each other. Thus, it is impossible to factor the nearly-Gaussian distribution into the product of N statistically independent distributions. In this sense, what is meant by *interaction* is the breakdown of *statistical independence*.[12]

Nearly-Gaussian actions

Having given a concrete example in which we illustrated how to deform the quadratic action to realize the simplest nearly-Gaussian distribution, we now give a more general perspective on nearly-Gaussian distributions. In what follows, we will continue to require that our distributions are invariant under the parity symmetry that takes $z_\mu \to -z_\mu$. In the action representation, this corresponds to including only terms of even degree.[13]

[12] An astute reader might wonder if there is any interaction when we consider a single-variable distribution with $N = 1$, since there's no other variables to interact with. For nearly-Gaussian distributions, even if $N = 1$, we saw in (1.74) that the variance of the distribution is shifted from its Gaussian value, K, and depends on the quartic coupling ϵV. In physics, we say that this shift is due to the *self-interaction* induced by the quartic coupling ϵV, since it modifies the value of observables from the *free* Gaussian theory that we are comparing to, even though there's no notion of statistical independence to appeal to here.

Said another way, even though the action just involves one term, such a non-Gaussian distribution does not have a closed-form solution for its partition function or correlators; i.e. there's no trick that lets us compute integrals of the form $e^{-S(z)}$ exactly, when $S(z) = \frac{z^2}{2K} + \frac{1}{4!}\epsilon V z^4$. This means that we still have to make use of perturbation theory to analyze the self-interaction in such distributions.

[13] The imposition of such a parity symmetry, and thus the absence of odd-degree terms in the action, means that all of the odd moments and hence all of the odd-point connected correlators will vanish.

With that caveat in mind, though otherwise very generally, we can express a **non-Gaussian distribution** by *deforming* the Gaussian action as

$$S(z) = \frac{1}{2}\sum_{\mu,\nu=1}^{N} K^{\mu\nu} z_\mu z_\nu + \sum_{m=2}^{k} \frac{1}{(2m)!} \sum_{\mu_1,\ldots,\mu_{2m}=1}^{N} s^{\mu_1\cdots\mu_{2m}} z_{\mu_1}\cdots z_{\mu_{2m}} \,, \tag{1.81}$$

where the factor of $1/(2m)!$ is conventional in order to compensate for the overcounting in the sum due to the implied symmetry of the indices $\mu_1, \ldots, \mu_{2m}$ in the coefficients $s^{\mu_1\cdots\mu_{2m}}$, given the permutation symmetry of the product of variables $z_{\mu_1}\cdots z_{\mu_{2m}}$. The number of terms in the non-Gaussian part of the action is controlled by the integer k. If k were unbounded, then $S(z)$ would be an arbitrary even function, and $p(z)$ could be any parity-symmetric distribution. The action is most useful when the expanded polynomial $S(z)$ truncated to reasonably small degree k – like $k = 2$ for the quartic action – yields a good representation for the statistical process of interest.

The coefficients $s^{\mu_1\cdots\mu_{2m}}$ are generally known as **non-Gaussian couplings**, and they control the **interactions** of the z_μ.[14] In particular, there is a direct correspondence between the product of the specific components z_μ that appear together in the action and the presence of connected correlation between those variables, with the degree of the term in (1.81) directly contributing to connected correlators of that degree. We saw an example of this in (1.78), which connected the quartic term to the connected four-point correlator. In this way, the couplings give a very direct way of controlling the degree and pattern of non-Gaussian correlation, and the overall degree of the action offers a way of systematically including more and more complicated patterns of such correlations.

If you recall from §1.2, we defined nearly-Gaussian distributions as ones for which all these connected correlators are small. Equivalently, from the action perspective, a nearly-Gaussian distribution is a non-Gaussian distribution with an action of the form (1.81) for which all the couplings $s^{\mu_1\cdots\mu_{2m}}$ are parametrically small for all $1 \leq m \leq k$:

$$|s^{\mu_1\cdots\mu_{2m}}| \ll |K^{\mu\nu}|^m \,, \tag{1.82}$$

where this equation is somewhat schematic given the mismatch of the indices.[15] Importantly the comparison is with an appropriate power of the inverse variance or quadratic

[14] In the similar vein, the coefficient $K^{\mu\nu}$ in the action is sometimes called a *quadratic coupling* since the *coupling* of the component z_μ with the component z_ν in the quadratic action leads to a nontrivial *correlation*, i.e. $\mathrm{Cov}[z_\mu, z_\nu] = K_{\mu\nu}$.

[15] This schematic equation is, nonetheless, dimensionally consistent. To support that remark, let us give a brief introduction to *dimensional analysis*: let the random variable z_μ have dimension ζ, which we denote as $[z_\mu] = \zeta^1$. By *dimension*, you should have in mind something like a unit of length, so e.g. we read the expression $[z_\mu] = \zeta^1$ as "a component of z is measured in units of ζ." The particular units are arbitrary: e.g. for length, we can choose between `meters` or `inches` or `parsecs` as long as we use a unit of length but *not*, say, `meters`^{2}, which instead would be a unit of area. Importantly, we cannot add or equate quantities that have different units: it doesn't make any logical sense to add a length to an area. This is similar to the concept of *type safety* in computer science, e.g. we should not add a type `str` variable to a type `int` variable.

Now, since the action $S(z)$ is the argument of an exponential $p(z) \propto e^{-S(z)}$, it must be *dimensionless*; otherwise, the exponential $e^{-S} = 1 - S + \frac{S^2}{2} + \ldots$ would violate the addition rule that we just described. From this dimensionless requirement for the action, we surmise that the inverse of the covariance matrix

coupling $K^{\mu\nu}$ since, as we already explained, the variance sets the scale of the Gaussian distribution to which we are comparing these nearly-Gaussian distributions.

As we will see in §4, wide neural networks are described by nearly-Gaussian distributions. In particular, we will find that such networks are described by a special type of nearly-Gaussian distribution where the connected correlators are *hierarchically* small, scaling as

$$\mathbb{E}\left[z_{\mu_1}\cdots z_{\mu_{2m}}\right]\big|_{\text{connected}} = O(\epsilon^{m-1})\,, \tag{1.83}$$

with the same parameter ϵ controlling the different scalings for each of the $2m$-point connected correlators. Importantly, the non-Gaussianities coming from higher-point connected correlators become parametrically less important as ϵ becomes smaller.

This means that for a nearly-Gaussian distribution with hierarchical scalings (1.83), we can consistently approximate the distribution by *truncating* the action at some fixed order in ϵ. To be concrete, we can use an action of the form (1.81) to faithfully represent all the correlations up to order $O(\epsilon^{k-1})$, neglecting connected correlations of order $O(\epsilon^k)$ and higher. The resulting action offers a useful and effective description for the statistical process of interest, as long as ϵ is small enough and k is high enough that $O(\epsilon^k)$ is negligible.

In practice, a quartic action (1.71) truncated to $k = 2$ will let us model realistic finite-width neural networks. This quartic action captures the important *qualitative* difference between nearly-Gaussian distributions and the Gaussian distribution, incorporating nontrivial interactions between the different components of the random variable. In addition, the difference between the statistics (1.83) of a nearly-Gaussian distribution truncated to $O(\epsilon)$ versus one truncated to $O(\epsilon^2)$ is mostly *quantitative*: in both cases there are nontrivial non-Gaussian correlations, but the pattern of higher-order correlation differs only in a small way, with the difference suppressed as $O(\epsilon^2)$. In this way, the distribution represented by the quartic action is complex enough to capture the most salient non-Gaussian effects in neural networks while still being simple enough to be analytically tractable.

---

has dimension $[K^{\mu\nu}] = \zeta^{-2}$, and that the covariance itself has dimension $[K_{\mu\nu}] = \zeta^2$. Similarly, all the non-Gaussian couplings in (1.81) have dimensions $[s^{\mu_1\cdots\mu_{2m}}] = \zeta^{-2m}$. Thus, both sides of (1.82) have the same dimension, making this equation dimensionally consistent.

Even more concretely, consider the quartic action (1.71). If we let the tensorial part of the quartic coupling have dimensions $[V^{\mu\nu\rho\lambda}] = \zeta^{-4}$, then the parameter ϵ is dimensionless, as claimed. This means that we can consistently compare ϵ to unity, and its parametric smallness $\epsilon \ll 1$ means that the full quartic coupling $\epsilon V^{\mu\nu\rho\lambda}$ is much smaller than the square of the quadratic coupling, and that the connected four-point correlator (1.78) is much smaller than the square of the connected two-point correlator (1.74).

Chapter 2

Neural Networks

> *On being asked, "How is Perceptron performing today?" I am often tempted to respond, "Very well, thank you, and how are Neutron and Electron behaving?"*
>
> Frank Rosenblatt, inventor of the perceptron and also the Perceptron [6].

With our mathematical lessons concluded, we turn to an introductory overview of deep learning.

In §2.1, we introduce the basic components of neural network architectures – neurons, activations, biases, weights, and layers – in order to define the *multilayer perceptron* (MLP), a simple model that is iteratively composed of these basic components. Given that all deep networks are by definition iteratively composed of many structurally identical layers, MLPs will play the role of archetype network architecture for illustrating the principles of deep learning throughout the book. This class of neural-network models is rich enough to capture all the essential aspects of deep learning theory, while simple enough to maintain the pedagogical focus of the book. Nevertheless, we'll also briefly comment on how one could work out an effective theory for other network architectures.

In §2.2 we list some common activation functions that are often used in practice.

Finally, we discuss in §2.3 how MLPs are initialized. Here, we make a key conceptual shift from thinking about the weights and biases as the random variables to thinking about the induced distribution over the neural activities and network outputs. The expressions we derive here will provide a natural starting point for our analysis in §4 when we start developing our effective theory of MLPs with general activation functions.

2.1 Function Approximation

The subject of *artificial neural networks* has a rich history as cognitive science and neuroscience–inspired artificial intelligence.[1] Here, our starting point will be a discussion

[1]The *artificial neuron* was invented by McCulloch and Pitts in 1943 [7] as a model of the *biological neuron*. Their neuron was essentially a perceptron with a bias, but did not have learnable weights. The perceptron model, with learnable weights, was invented by Rosenblatt in 1958 [8]. Deep learning really

of the function, $f(x)$.

Some functions are really simple, easily described in terms of the elementary operations: addition, subtraction, multiplication, and division. For instance, consider either the identity function $f(x) = x$ or the exponential function $f(x) = e^x$. The former is the definition of trivial, involving no operations. The latter is a special function and can be defined in many ways, e.g. through its Taylor series

$$e^x \equiv \sum_{k=0}^{\infty} \frac{x^k}{k!} \,. \tag{2.1}$$

This definition constructs the exponential function in terms of elementary operations of addition, multiplication, and division: the numerator x^k represents the repeated multiplication of the variable x for k times, and the factorial $k!$ in the denominator represents the repeated multiplication of integers $k! = 1 \times 2 \times \cdots \times (k-1) \times k$. Although this description of the exponential function involves a sum of an infinite number of terms, the actual instructions (2.1) for computing this function in terms of these simple operations are so compact that they takes up only about one seventh of a line and, for many purposes, it only takes the first few terms in the sum to get a useful approximation of e^x.

Some functions are really complicated and their description in terms of elementary operations is unlikely to fit in the confines of any printed book. For instance, imagine a function $f(x)$ that takes as input an image x_i – represented as a vector of numbers corresponding to a black-and-white pixelated image – and outputs 1 if the image x_i depicts a cat and 0 otherwise. While such a classification function should exist since humans can recognize images of cats, it's not at all clear how to describe such a function in terms of simple operations like addition and multiplication. The subject of **artificial intelligence** (AI) is mostly concerned with functions of this sort: easy for humans to compute, but difficult for humans to describe in terms of elementary operations.

The conceptual leap needed to represent such hard-to-describe functions is to start with a flexible *set* of functions $\{f(x;\theta)\}$, constructed from simple components parametrized by a vector of adjustable **model parameters** θ_μ. One then tries to tune these model parameters θ_μ judiciously in order to approximate the original complicated function such that $f(x;\theta^\star) \approx f(x)$. The description of the set of functions $\{f(x;\theta)\}$ as well as the settings of the model parameters $\theta^\star_\mu$ then serve as a useful approximate description of a desired function $f(x)$. This is called **function approximation** and the procedure for adjusting the model parameters θ_μ is called a **learning algorithm**.

To be more concrete, let us represent the collection of inputs to our function $f(x)$ as a set $\mathcal{D}$ of n_0-dimensional vectors

$$\mathcal{D} = \{x_{i;\alpha}\}_{\alpha=1,\ldots,N_\mathcal{D}} \,, \tag{2.2}$$

called **input data**. Here, the **sample index** α labels each sample in the dataset of $N_\mathcal{D}$ elements, and the vector index $i = 1, \ldots, n_0$ labels the component of the input vector.

came into its own in 2012 [9] after the realization that the graphical processing unit (GPU) is well-suited for the parallel computations required to train and run neural networks.

In our motivating example above, each number $x_{i;\alpha}$ refers to the i-th pixel of the α-th image in the dataset $\mathcal{D}$ of $N_{\mathcal{D}}$ images, each of which might or might not depict a cat. By adjusting the model parameters θ_μ so that the function $f(x;\theta^\star)$ outputs the correct answer for as much input data as possible, we can try to approximate the elusive cat-or-not function in a way that no longer defies description. The overall idea of *training* such functions using a dataset $\mathcal{D}$ – rather than *programming* them – goes by the name **machine learning** and stands in contrast to the conventional von Neumann model of the digital computer.

While any set of parameterized functions can be used for function approximation,[2] our focus will be on a particular set of composable functions originally derived from a simplified model of the brain. Such functions were originally termed **artificial neural networks** and are now just referred to as **neural networks**. **Deep learning** is a branch of machine learning that uses neural networks as function approximators, with a particular emphasis on stacking *many layers* of structurally similar components. Let's see how this works in more detail.

The most basic component of the neural network is the **neuron**. Loosely inspired by the behavior of biological neurons, the artificial neuron essentially consists of two simple operations:

- The **preactivation** z_i of a neuron is a linear aggregation of incoming signals s_j where each signal is weighted by W_{ij} and biased by b_i

$$z_i(s) = b_i + \sum_{j=1}^{n_{\text{in}}} W_{ij}\, s_j \quad \text{for} \quad i = 1, \ldots, n_{\text{out}}\,. \tag{2.3}$$

- Each neuron then *fires* or not according to the weighted and biased evidence, i.e. according to the value of the preactivation z_i, and produces an **activation**

$$\sigma_i \equiv \sigma(z_i)\ . \tag{2.4}$$

  The scalar-valued function $\sigma(z)$ is called the **activation function** and acts independently on each component of the preactivation vector.

Taken together, these n_{out} neurons form a **layer**, which takes in the n_{in}-dimensional vector of signals s_j and outputs the n_{out}-dimensional vector of activations σ_i. With this collective perspective, a layer is parameterized by a vector of **biases** b_i and a matrix of **weights** W_{ij}, where $i = 1, \ldots, n_{\text{out}}$ and $j = 1, \ldots, n_{\text{in}}$, together with a fixed activation function $\sigma(z)$.

With these components, we can make an increasingly flexible set of functions by organizing many neurons into a layer and then iteratively stacking many such layers, so that the outgoing activations of the neurons in one layer become the input signals to the neurons in some other layer. The organization of the neurons and their pattern

[2] E.g. consider a sum of Gaussian functions, where the mean and variance of each Gaussian play the role of the adjustable parameters.

of connections is known as the neural network **architecture**. The archetypical neural network architecture based on this principle of stacking layers of many neurons is called the **multilayer perceptron** (MLP).[3]

The activation function is usually chosen to be a nonlinear function in order to increase the expressivity of the neural-network function $f(x;\theta)$. The simplest – and historically first – activation function either fires or does not fire: $\sigma(z) = 1$ for $z \geq 0$ and $\sigma(z) = 0$ for $z < 0$. In other words, each neuron fires if and only if the weighted accumulated evidence $\sum_j W_{ij}\, x_j$ exceeds the firing threshold $-b_i$. More generally, activation functions are not binary and can incorporate the strength of the evidence into their output. In §2.2 we'll describe many of the commonly-used activation functions in deep learning.

The MLP is recursively defined through the following iteration equations

$$z_i^{(1)}(x_\alpha) \equiv b_i^{(1)} + \sum_{j=1}^{n_0} W_{ij}^{(1)} x_{j;\alpha}\,, \quad \text{for} \quad i = 1, \ldots, n_1\,, \tag{2.5}$$

$$z_i^{(\ell+1)}(x_\alpha) \equiv b_i^{(\ell+1)} + \sum_{j=1}^{n_\ell} W_{ij}^{(\ell+1)} \sigma\Big(z_j^{(\ell)}(x_\alpha)\Big)\,, \quad \text{for} \quad i = 1, \ldots, n_{\ell+1}\,;\ \ell = 1, \ldots, L-1\,,$$

which describes a network with L layers of neurons, with each layer ℓ composed of n_ℓ neurons.[4] We depict an example MLP architecture in Figure 2.1. The number of layers L defines the **depth** of the network and the different number of neurons in each layer $n_{\ell=1,\ldots,L-1}$ define the **widths** of the layers. The depth and hidden-layer widths are variable **architectural hyperparameters** that define the shape of the network, while the values of n_0 and n_L are set by input and output dimensions of the function-approximation task, respectively. In particular, the final-layer preactivations computed by the network

$$f\,(x;\theta) = z^{(L)}(x), \tag{2.6}$$

serves as the function approximator, with its model parameters θ_μ being the union of the biases and weights from all the layers. Sometimes it will be convenient to think of this collection of model parameters as an explicit vector θ_μ whose components cover all the model parameters. In that case, the dimension of θ_μ and therefore the total number of the model parameters is given by

$$P = \sum_{\ell=1}^{L} \left(n_\ell + n_\ell n_{\ell-1}\right)\,, \tag{2.7}$$

[3] Here, the name "perceptron" was inherited from Rosenblatt's Perceptron architecture [8], which was originally envisioned for emulating *human perception*. The name perceptron is also used to refer to the original step-function activation function, cf. the first entry of §2.2.

[4] A more modern name for the MLP is the *fully-connected network* (FCN), highlighting the fact that each neuron in a given layer ℓ has a connection to every neuron in layer $\ell + 1$, as Figure 2.1 makes clear. Such a dense pattern of connections is computationally expensive in terms of the number of parameters required for the architecture and should be contrasted with the sparser architectures described at the end of this section. To place an emphasis on the *deepness* of networks rather than on the *density* of the connections, we'll mainly stick with the name *multilayer* perceptron over the name *fully-connected* network in this book.

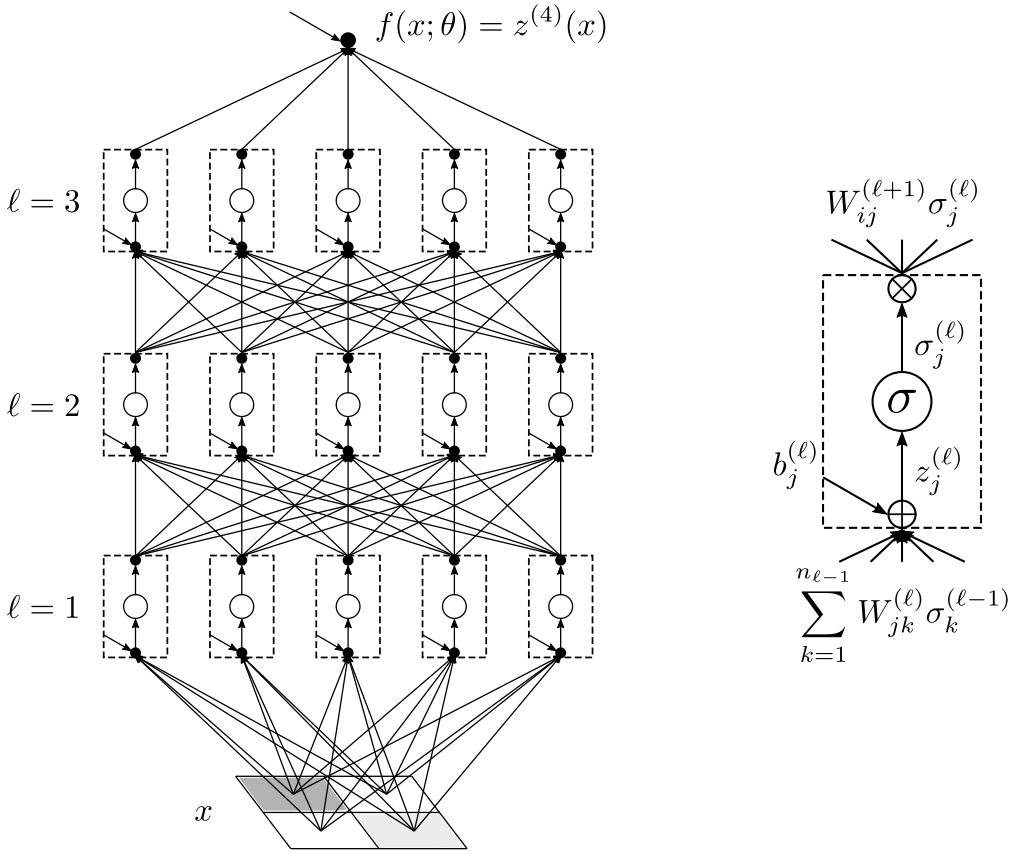

Figure 2.1: **Left:** depiction of the neurons and connections for an example multilayer perceptron (MLP) architecture. This particular MLP has $L = 4$ layers, defining a set of functions $f(x;\theta)$ with input dimension $n_0 = 4$ and output dimension $n_4 = 1$. The three hidden layers have five neurons each $n_1, n_2, n_3 = 5$, implying $P = 91$ total model parameters. The graph describing the connections between neurons is a *directed acyclic graph*, meaning that signals only propagate in one direction and do not loop inside the network. For this reason, MLPs are also sometimes called *feed-forward* networks. **Right:** the detailed structure of each neuron that *(i)* adds the bias and the weighted signals to produce the preactivation, *(ii)* generates the activation from the preactivation, and *(iii)* multiplies the activation by the next-layer weight.

which scales quadratically with the widths of the network and linearly with the depth.

The intermediate layers $\ell = 1, \ldots, L-1$ are referred to as **hidden layers**, since pre-activations and activations of the neurons from those layers are not part of the network's output. On the one hand, the variables $z^{(\ell)}(x)$ for $\ell < L$ are simply temporary variables introduced to construct an increasingly flexible set of functions, expressive enough to have a chance of approximating hard-to-describe functions. On the other hand, in analogy to the physical brain, these variables are thought to encode useful information about *how* the neural network is approximating; for example, a particular neuron might fire if it recognizes a tail, a whisker, or a pattern representing fur – all potentially useful *features* for determining whether an image contains a cat or not.

Moving beyond MLPs, the choice of neural network architecture is often motivated by the nature of the function we are trying to approximate. For instance, the properties of the dataset $\mathcal{D}$, when known and articulated, can be used to build **inductive biases** into the architecture so that the resulting set of functions may better represent the underlying function.[5] Let's look at a few examples.

- For **computer vision** (CV) applications, **convolutional neural networks** (CNN) or conv-nets [9–13] are used to take advantage of the fact that information in images is organized in a spatially local manner, often respecting *translational invariance*.[6]
- For **natural language processing** (NLP) applications, the **transformer** architecture (no acronym yet) is used to process sequential input – such as a paragraph of text or an amino acid sequence coding a protein – in a way that encourages correlations to develop between any of the elements in the sequence [14]. This property of the model is aptly called *attention*.

An important property of these inductive biases is that they induce constraints or relationships between the weights. For instance, we can think of the convolutional layer as a particular type of MLP layer, where many weights are set to zero and the values of remaining weights are further shared among several different neurons. This property is known as *weight tying*. That means that convolutional layers are actually within the class of functions describable by using MLP layers, but they are very unlikely to be found via training unless the constraints are explicitly enforced. As long as the inductive bias

[5] We'll discuss the inductive bias of MLPs from various different perspectives in §6, §11, and Epilogue ε.

[6] For a two-dimensional convolutional layer, the iteration equation (2.5) for MLPs is replaced by

$$z^{(\ell+1)}_{i,(c,d)}(x_\alpha) \equiv b^{(\ell+1)}_i + \sum_{j=1}^{n_\ell} \sum_{c'=-k}^{k} \sum_{d'=-k}^{k} W^{(\ell+1)}_{ij} \sigma\left(z^{(\ell)}_{j,(c+c',d+d')}(x_\alpha)\right) , \tag{2.8}$$

where in $z^{(\ell)}_{i,(c,d)}$, the first index i is an auxiliary *channel* index and the paired index (c,d) is a two-dimensional spatial index, and the number k is a fixed constant for each layer, determining the size of the convolutional window. In particular, the same weights are used on different spatial locations of the input, which promotes the inductive bias that image data are often translationally invariant. In other words, a cat is still a cat regardless of its location in an image. At the time of writing, the convolutional layer is an essential part of many modern deep learning architectures, but this situation may change in the future. Please pay *attention*.

of spatial locality and translational invariance is well founded, the convolution layer has obvious computational advantages by heavily curtailing the number of weights to be trained and stored.

Regardless of these specific inductive biases ingrained into modern neural network architectures used in deep learning, the common thread to all is the idea of constructing a flexible set of functions by organizing neural components into many iterated layers. MLPs are the simplest of these neural network architectures that hinges on this stacking idea, and thus provide a *minimal model* for an **effective theory of deep learning**. Specifically, we expect that (*a*) the *principles of deep learning theory* that we uncover to be general and valid across the large variety of architectures that are based on the idea of stacking many layers of neural components and (*b*) the resulting effective theory formalism can be specialized to specific architectures of interest as needed, using this book as a guide for how to work out such a theory. In particular, one can study other architectures in our formalism simply by swapping out the MLP iteration equation (2.5) – e.g. for the convolution layer iteration equation (2.8) – in the appropriate place. We'll provide pointers on where to make such substitutions when we begin working out our effective theory in §4.

Finally, in Appendix B we'll study neural networks with *residual connections*, known as **residual networks**. These architectures are specially modified to enable the training of deeper and deeper networks. In the final section of that appendix, we'll also explain how our effective theory approach can be extended to *general* residual networks, including the *residual convolution network* or **ResNet** and the *transformer* architecture.

2.2 Activation Functions

In this section, we discuss some of the most common activation functions. This list is non-exhaustive, so hopefully you won't find this section exhausting. To make it easier for you, we've plotted all these activation functions together in Figure 2.2. In §5, we'll use our effective theory to evaluate the relative usefulness of these activation functions in allowing input signals to effectively pass through a deep network.

Perceptron

The `perceptron` was the original activation function [7]. It is just a step function

$$\sigma(z) = \begin{cases} 1\,, & z \geq 0\,, \\ 0\,, & z < 0\,, \end{cases} \tag{2.9}$$

corresponding to a computer scientist's notion of simplicity: the neuron either *fires* and outputs 1 or *doesn't fire* and outputs 0.[7]

Despite the logical simplicity, this turns out to be a poor choice. As we will see, in order to both effectively pass signals through networks (§5) as well as train them (§9),

[7] Alternatively, the `perceptron` may be shifted and scaled such that $\sigma(z) = \text{sign}(z)$.

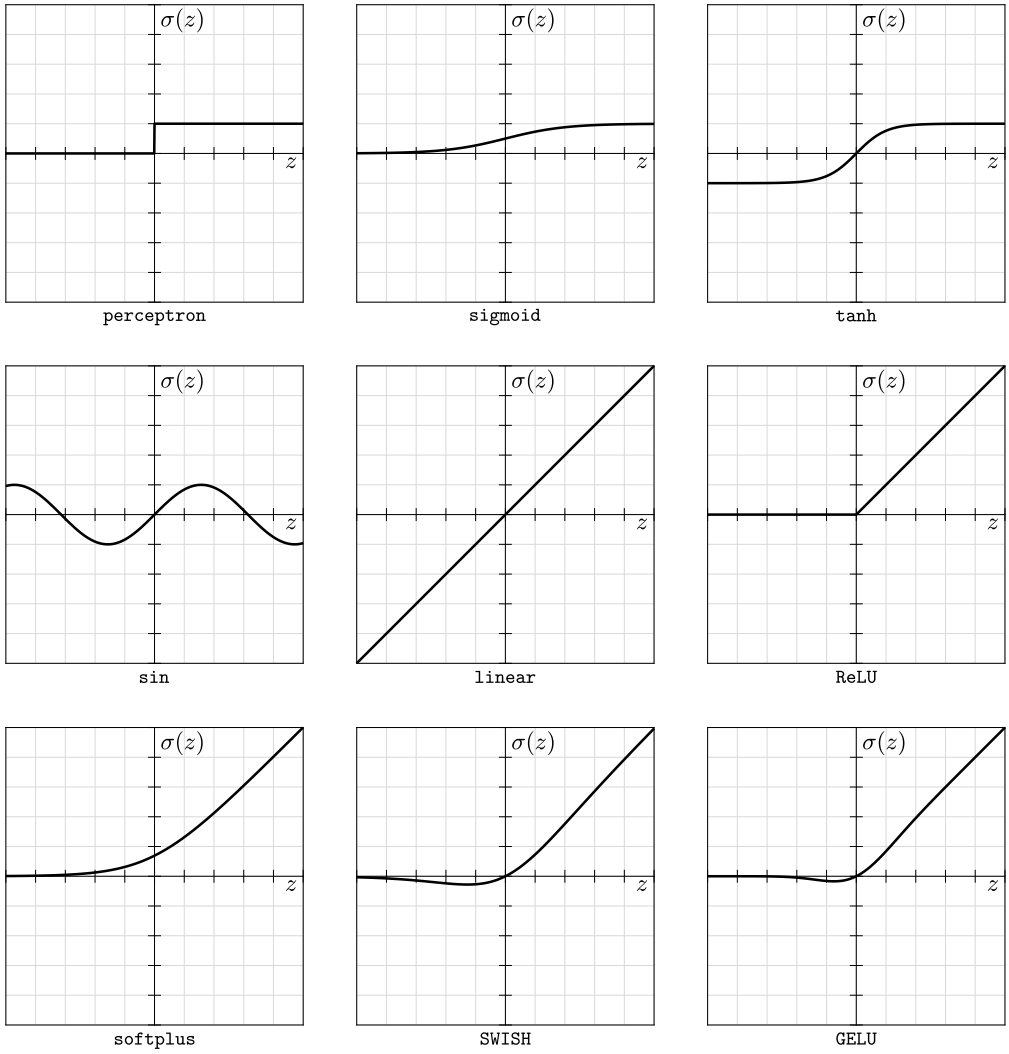

Figure 2.2: Commonly-used activation functions $\sigma(z)$. Grids are in units of one for both the preactivation z and activation σ. (The `leaky ReLU` is not shown.)

it's helpful to propagate more than one bit of information about the preactivation z. The `perceptron` has historical significance, but is never used in deep neural networks.

Sigmoid

The `sigmoid` activation function is a logistic function

$$\sigma(z) = \frac{1}{1+e^{-z}} = \frac{1}{2} + \frac{1}{2}\tanh\left(\frac{z}{2}\right) , \tag{2.10}$$

which is a smoothed version of the `perceptron`. Not only is it continuous, but also it preserves information about the magnitude of preactivation, albeit mostly in the range near $z = 0$ where the function is nearly linear. Outside of this range, the `sigmoid` heavily compresses such information as it becomes more and more `perceptron`-like, *saturating* as $\sigma(z) = 1$ when $z \to \infty$ and as $\sigma(z) = 0$ when $z \to -\infty$.

As a mapping from the domain of $(-\infty, \infty)$ to the range $[0, 1]$, the `sigmoid` also has a natural interpretation of converting log-odds to a probability, which is its main application in machine learning. For deep learning, the differentiability of the `sigmoid` was essential in the development of a learning algorithm – backpropagation – for training neural networks with hidden layers [15]. Nevertheless, the `sigmoid` activation function is still a poor choice in *deep* neural networks: as we'll see in §5, a problem arises from the fact that it doesn't pass through the origin.

Tanh

The hyperbolic tangent or `tanh` activation function

$$\sigma(z) = \tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} = \frac{e^{2z} - 1}{e^{2z} + 1} , \tag{2.11}$$

is a scaled (both before and after the activation) and shifted `sigmoid`, as is clear from (2.10). Of particular importance is the fact that it's shifted such that $\sigma(0) = 0$ [16].

The `tanh` is probably the most popular choice of activation function aside from the `ReLU` or `ReLU`-like activation functions to be discussed shortly, and arguably `tanh` is the most popular smooth activation function. As an exemplary smooth activation function, the `tanh` will be of significant interest for us in this book.

Sin

The `sin` activation function is just what it sounds like:

$$\sigma(z) = \sin(z) , \tag{2.12}$$

i.e. one of the three standard trigonometric function. Periodic nonlinearities have been cycling in and out of popularity for a long while now, see e.g. [17], though they have never really achieved true popularity.

Scale-invariant: linear, ReLU, and leaky ReLU

A scale-invariant activation function is any activation function that satisfies

$$\sigma(\lambda z) = \lambda \sigma(z)\,, \tag{2.13}$$

for any positive rescaling λ. We call these activation functions scale-invariant because any scaling of the preactivation $z \to \lambda z$ can be undone by an inverse scaling of the activation $\sigma(z) \to \lambda^{-1}\sigma(z)$. This condition is met by – and only by[8] – activation functions of the form

$$\sigma(z) = \begin{cases} a_+ z\,, & z \geq 0\,, \\ a_- z\,, & z < 0\,. \end{cases} \tag{2.14}$$

The class of scale-invariant activation functions includes `linear` ($a_+ = a_- = a$), Rectified Linear Unit or `ReLU` ($a_+ = 1$, $a_- = 0$) [18, 19], and `leaky ReLU` ($a_+ = 1$, $a_- = a$) [20] activation functions. The `ReLU` is the most popular of the activation functions used in deep neural networks and therefore will be of substantial interest for us in this book.

In order to deepen our understanding of scale invariance, let's consider how other activation functions can break it. For instance, consider the `tanh` activation function $\sigma(z) = \tanh(z)$. Mathematically, `tanh` violates scale invariance because $\tanh(\lambda z) \neq \lambda \tanh \sigma(z)$ unless $\lambda = 1$. In particular, while the activation function is approximately linear for small preactivations, i.e. $\tanh(z) \approx z$ for $|z| \ll 1$, it saturates for large preactivations, i.e. $|\tanh(z)| \approx 1$ for $|z| \gg 1$. Thus, `tanh` comes with an intrinsic crossover scale $|z| \sim 1$ that separates the two regimes. We can see this visually in Figure 2.2: if we zoom out, all the non-scale-invariant activation functions – e.g. `perceptron`, `sigmoid`, and `tanh` – will look squashed, while the scale-invariant activation functions – e.g. `ReLU` and `linear` – will look the same at any scale.

Finally, note that all the scale-invariant activation functions – except the aptly-named `linear` activation – create a nonlinear relationship between the network inputs and outputs due to the kink at the origin $z = 0$. Stacking up many layers of neurons with these nonlinear activation functions accumulates the nonlinearity, allowing such deep neural networks to express highly nonlinear functions.

ReLU-like: softplus, SWISH, and GELU

Despite the popularity of the `ReLU`, there's an uneasiness about the fact that it's not smooth. In an attempt to rectify the situation, a variety of smoothed-out `ReLU`-like activations have been proposed and achieved semi-popularity, of which we will consider the following three:

[8] In order to prove this necessity statement, first take the derivative of the scale-invariance equation (2.13) with respect to z, which gives $\sigma'(\lambda z) = \sigma'(z)$ for any $\lambda > 0$. Then note that this enforces a constant derivative, a_+, for $z > 0$ and another constant derivative, a_-, for $z < 0$. Finally, to satisfy (2.13) we also must have $\lim_{z \to \pm 0} \sigma(z) = 0$. Quantum Electrodynamics.

- The `softplus` activation function [21]

$$\sigma(z) = \log\left(1 + e^z\right) , \tag{2.15}$$

behaves linearly $\sigma(z) \approx z$ for a large argument $z \gg 1$ and vanishes exponentially for a negative argument, $\sigma(z) \approx e^{-|z|}$ for $z < 0$. Importantly the `softplus` does not pass through the origin: $\sigma(0) = \log(2)$.

- The `SWISH` activation function [22] is defined as

$$\sigma(z) = \frac{z}{1 + e^{-z}} , \tag{2.16}$$

which is a logistic function (2.10) multiplied by the preactivation z. The logistic function behaves as a continuous on/off switch, and so the `SWISH` approximates the `ReLU`, which we recall was defined as a discrete on/off switch multiplied by the preactivation z. In particular, for $z > 0$ the `SWISH` behaves as $\sigma(z) \approx z$, but for $z < 0$ it behaves as $\sigma(z) \approx 0$. Also, the multiplication by z ensures that the `SWISH` passes through the origin, $\sigma(0) = 0$.

- The Gaussian Error Linear Unit (`GELU`) activation function [23] is a lot like the `SWISH`. It's given by the expression

$$\sigma(z) = \left[\frac{1}{2} + \frac{1}{2}\text{erf}\left(\frac{z}{\sqrt{2}}\right)\right] \times z , \tag{2.17}$$

where the error function $\text{erf}(z)$ is given by

$$\text{erf}(z) \equiv \frac{2}{\sqrt{\pi}} \int_0^z dt\, e^{-t^2} , \tag{2.18}$$

which is a partial integration of the Gaussian function. In particular, the graph of $\text{erf}(z)$ looks very similar to graph of $\tanh(z)$, and so the graph of the scaled and shifted version used in the definition of the `GELU`, $\frac{1}{2} + \frac{1}{2}\text{erf}\left(\frac{z}{\sqrt{2}}\right)$, looks very similar to the graph of the logistic function (2.10). Like the `SWISH`, it crosses the origin and behaves more like the `ReLU` the further we go away from 0 in either direction.

In smoothing the `ReLU`, all three of these activation functions introduce an intrinsic scale and violate the scale-invariance condition (2.13).

2.3 Ensembles

As we discussed in §2.1, neural networks are *trained* rather than *programmed*. Practically speaking, to begin training a neural network for function approximation, we need to set initial values of the biases $b_i^{(\ell)}$ and weights $W_{ij}^{(\ell)}$. Since the learned values of these model parameters are almost always iteratively built up from their initial values, the

initialization strategy can have a major impact on the success or failure of the function approximation.

Perhaps the simplest strategy would be to set all the biases and weights to zero, $b_i^{(\ell)} = W_{ij}^{(\ell)} = 0$. However, this initialization fails to break the permutation symmetry among the n_ℓ different neurons in a hidden layer ℓ. If this symmetry isn't broken, then we cannot distinguish between the different neurons in a layer as all these neurons perform exactly the same computation. In effect, the network would behave as if it only had single neuron $n_\ell = 1$ in each hidden layer. Thus, in order to leverage all the different components of the biases and weights in a wider network, we need to somehow break the permutation symmetry.

Perhaps the simplest strategy that breaks this permutation symmetry is to sample each bias and weight independently from some *probability distribution*. Theoretically speaking, we should pick this **initialization distribution** so that the resulting **ensemble** of networks are well behaved with respect to the function-approximation task. This section initializes ourselves for analyzing such an ensemble.

Initialization distribution of biases and weights

Among the many potential reasonable choices for the initialization distribution, the obvious choice is the Gaussian distribution.[9] As we discussed, Gaussian distributions are defined solely in terms of their mean and variance, so they're easy to specify and work with in theory. They're also extremely easy to sample from, which is also an essential consideration when picking a sampling distribution in practice.

In particular, to initialize MLPs, we'll independently sample each bias and each weight from zero-mean Gaussian distributions with variances given by

$$\mathbb{E}\left[b_{i_1}^{(\ell)} b_{i_2}^{(\ell)}\right] = \delta_{i_1 i_2} C_b^{(\ell)} \,, \tag{2.19}$$

$$\mathbb{E}\left[W_{i_1 j_1}^{(\ell)} W_{i_2 j_2}^{(\ell)}\right] = \delta_{i_1 i_2}\delta_{j_1 j_2} \frac{C_W^{(\ell)}}{n_{\ell-1}} \,, \tag{2.20}$$

respectively. Here the Kronecker deltas indicate that each bias and each weight are all drawn independently from the others. Explicitly, the functional forms of these Gaussian

[9] Two other choices seen in the wild for the initialization distribution are the uniform distribution and the truncated normal distribution. For the weights, the difference between the Gaussian distribution and any other distribution – when the means are set zero and the variances are set equal – turns out to be suppressed by 1/width for wide networks. That is, due to the central limit theorem, ultimately only the first and second moment – i.e. the mean and variance – for the weight initialization distribution is of any real consequence. Thus, we might as well just use a Gaussian distribution. For the biases, the difference between the Gaussian distribution and any other distribution is mostly moot in practice because we shall find that the bias variance $C_b^{(\ell)}$ should be set to zero for all good activation functions.

distributions are given by

$$p\left(b_i^{(\ell)}\right) = \frac{1}{\sqrt{2\pi C_b^{(\ell)}}} \exp\left[-\frac{1}{2C_b^{(\ell)}}\left(b_i^{(\ell)}\right)^2\right], \tag{2.21}$$

$$p\left(W_{ij}^{(\ell)}\right) = \sqrt{\frac{n_{\ell-1}}{2\pi C_W^{(\ell)}}} \exp\left[-\frac{n_{\ell-1}}{2C_W^{(\ell)}}\left(W_{ij}^{(\ell)}\right)^2\right]. \tag{2.22}$$

Here the normalization of weight variances by $1/n_{\ell-1}$ is purely conventional but, as we will show explicitly in §3 and §4, it is necessary for wide neural networks and natural for comparing the behavior of networks with different widths.[10] Also note that we allow the bias variance $C_b^{(\ell)}$ and rescaled weight variance $C_W^{(\ell)}$ to potentially vary from layer to layer. Together, the set of bias variances $\left\{C_b^{(1)}, \ldots, C_b^{(L)}\right\}$ and the set of rescaled weight variances $\left\{C_W^{(1)}, \ldots, C_W^{(L)}\right\}$ are called **initialization hyperparameters**. One practical result of our effective theory approach will be prescriptions for setting these initialization hyperparameters so that the output of the neural network is well behaved.

Induced distributions

Given a dataset $\mathcal{D} = \{x_{i;\alpha}\}$ consisting of $N_{\mathcal{D}}$ input data, an MLP with model parameters $\theta_\mu = \left\{b_i^{(\ell)}, W_{ij}^{(\ell)}\right\}$ evaluated on $\mathcal{D}$ outputs an array of $n_L \times N_{\mathcal{D}}$ numbers

$$f_i(x_\alpha; \theta) = z_i^{(L)}(x_\alpha) \equiv z_{i;\alpha}^{(L)}, \tag{2.23}$$

indexed by both **neural indices** $i = 1, \ldots, n_L$ and **sample indices** $\alpha = 1, \ldots, N_{\mathcal{D}}$. Each time we instantiate MLPs by drawing model parameters θ_μ from the initialization distribution $p(\theta)$, we get a different initial set of outputs $z_{i;\alpha}^{(L)}$. It follows that since the biases $b_i^{(\ell)}$ and weights $W_{ij}^{(\ell)}$ are random variables at initialization, then so must be the network outputs $z_{i;\alpha}^{(L)}$. In this way, the initialization distribution *induces* a distribution on the network outputs.

This **output distribution** $p\left(z^{(L)}\middle|\mathcal{D}\right)$ controls the statistics of network outputs at the point of initialization. In practice, the properties of this distribution are directly related to how hard it is for a network to approximate its target function through iterated adjustments of its model parameters. As such, having control over this distribution is of significant interest from a practitioner's perspective. From a theorist's perspective, even though the initialization distribution for model parameters is simple by design, the

[10] We can trace this convention to the MLP iteration equation (2.5). To compute the preactivation $z_{i;\alpha}^{(\ell)} = b_i^{(\ell)} + \sum_{j=1}^{n_{\ell-1}} W_{ij}^{(\ell)} \sigma_{j;\alpha}^{(\ell-1)}$, we essentially add together $n_{\ell-1}$ random weights. For large $n_{\ell-1}$, the normalization factor of $1/n_{\ell-1}$ in the variance – which is tantamount to normalizing each weight by $1/\sqrt{n_{\ell-1}}$ – essentially counteracts this summation of many zero-mean random numbers. Since there is no such summation for the biases, there is no need for such a normalization factor.

induced output distribution is not. In theory, we need to calculate the following gigantic integral over all the model parameters

$$p\Big(z^{(L)}\Big|\mathcal{D}\Big) = \int \left[\prod_{\mu=1}^{P} d\theta_\mu\right] p\Big(z^{(L)}\Big|\theta,\mathcal{D}\Big)\, p(\theta)\,. \tag{2.24}$$

Before performing this heroic integration, notice that the conditional distribution $p\Big(z^{(L)}\Big|\theta,\mathcal{D}\Big)$ in the integrand (2.24) is actually *deterministic*. In other words, if we know the set of inputs $\mathcal{D}$ and the settings of all the model parameters θ_μ, then we know how to compute the network outputs: we just use the iteration equation (2.5) that defines the MLP. What we don't yet know is how to express this determinism as a distribution.

Deterministic distributions and the Dirac delta function

What kind of a probability distribution is *deterministic*? Let's abstractly denote such a distribution as $p(z|s) = \delta(z|s)$, which intend to encode the deterministic relationship $z = s$. What properties should this distribution have? First, the mean of z should be s

$$\mathbb{E}\,[z] = \int dz\; \delta(z|s)\, z \equiv s\,. \tag{2.25}$$

Second, the variance should vanish, since this is a deterministic relationship. In other words,

$$\mathbb{E}[z^2] - (\mathbb{E}\,[z])^2 = \left[\int dz\; \delta(z|s)\, z^2\right] - s^2 \equiv 0\,, \tag{2.26}$$

or, equivalently,

$$\int dz\; \delta(z|s)\, z^2 \equiv s^2. \tag{2.27}$$

In fact, this determinism implies an even stronger condition. In particular, the expectation of any function $f(z)$ of z, should evaluate to $f(s)$:

$$\mathbb{E}\,[f(z)] = \int dz\; \delta(z|s)\, f(z) \equiv f(s)\,, \tag{2.28}$$

which includes the properties (2.25) and (2.27) as special cases when $f(z) = z$ and $f(z) = z^2$, respectively, as well as the probability normalization condition

$$\int dz\; \delta(z|s) = 1\,, \tag{2.29}$$

when $f(z) = 1$.[11] In fact, (2.28) is the defining property of the **Dirac delta function**.[12]

[11] A random variable that obeys $\mathbb{E}\,[f(z)] = f\,(\mathbb{E}\,[z])$ is said to *self-average*, meaning that we can exchange the order of the expectation with the function evaluation. The condition (2.28) is equivalent to saying that the distribution $\delta(z|s)$ is self-averaging.

[12] The Dirac delta function is really a generalization of the Kronecker delta (1.26) for continuous variables. In this footnote we also include the obligatory disclaimer that – despite its name – the Dirac delta function is a *distribution* and not a *function*, as should have been clear from our discussion. Despite this, we will stick with common convention and continue to refer to it as the *Dirac delta function*.

As a representation though, (2.28) is a little too abstract, even for us. However, our discussion above paves the way for a much more concrete representation. Since the Dirac delta function is a normalized distribution (2.29) with mean s (2.25) and zero variance (2.26), let's consider a normalized Gaussian distribution with mean s (1.9) and take the limit as the variance K goes to zero:

$$\delta(z|s) \equiv \lim_{K\to+0} \frac{1}{\sqrt{2\pi K}} e^{-\frac{(z-s)^2}{2K}} \,. \tag{2.30}$$

This distribution is infinitely peaked at $z = s$ while vanishing everywhere else, so any function $f(z)$ integrated against (2.30) will give $f(s)$ after taking the limit. In other words, it satisfies the defining property of the Dirac delta function (2.28).

The limit in (2.30) should always be taken after integrating the distribution against some function. Having said that, perhaps this representation still makes you a little bit uncomfortable as it is still a very singular limit. Let's try to fix this and find a yet even better representation. Here's a magic trick: starting from (2.30), let's insert "1" on the right hand side as

$$\begin{aligned}\delta(z|s) &= \lim_{K\to+0} \frac{1}{\sqrt{2\pi K}} e^{-\frac{(z-s)^2}{2K}} \left\{ \frac{1}{\sqrt{2\pi/K}} \int_{-\infty}^{\infty} d\Lambda \, \exp\left[-\frac{K}{2} \left(\Lambda - \frac{i(z-s)}{K} \right)^2 \right] \right\} \\ &= \lim_{K\to+0} \frac{1}{2\pi} \int_{-\infty}^{\infty} d\Lambda \, \exp\left[-\frac{1}{2} K\Lambda^2 + i\Lambda(z-s) \right] ,\end{aligned} \tag{2.31}$$

where in the curly brackets we inserted an integral over a dummy variable Λ of a normalized Gaussian with variance $1/K$ and imaginary mean $i(z-s)/K$, and on the second line we simply combined the exponentials. Now we can easily take the limit $K \to +0$ to find an *integral representation* of the Dirac delta function

$$\delta(z|s) = \frac{1}{2\pi} \int_{-\infty}^{\infty} d\Lambda \, e^{i\Lambda(z-s)} \equiv \delta(z-s) \,. \tag{2.32}$$

In this final expression, we noted that the function depends only on the difference $z - s$. This integral representation will come in handy in §4.

Induced distributions, redux

Now that we are familiar with the Dirac delta function, we can use it to express the output distribution (2.24) more concretely. To start, for a one-layer network of depth $L = 1$, the distribution of the first layer output (2.5) is given by

$$\begin{aligned} p\Big(z^{(1)}\Big|\mathcal{D}\Big) = \int \left[\prod_{i=1}^{n_1} db_i^{(1)} \, p\Big(b_i^{(1)}\Big) \right] \left[\prod_{i=1}^{n_1} \prod_{j=1}^{n_0} dW_{ij}^{(1)} \, p\Big(W_{ij}^{(1)}\Big) \right] \\ \times \left[\prod_{i=1}^{n_1} \prod_{\alpha\in\mathcal{D}} \delta\left(z_{i;\alpha}^{(1)} - b_i^{(1)} - \sum_{j=1}^{n_0} W_{ij}^{(1)} x_{j;\alpha} \right) \right] . \end{aligned} \tag{2.33}$$

Here, we needed $n_1 \times N_{\mathcal{D}}$ Dirac delta functions, one for each component of $z^{(1)}_{i;\alpha}$. In §4.1 we will explicitly evaluate the above integrals, though you should feel free to do so now on your own, if you're impatient. In passing, let us also introduce a cousin of (2.33)

$$p\Big(z^{(\ell+1)}\Big|z^{(\ell)}\Big) = \int \left[\prod_{i=1}^{n_{\ell+1}} db_i^{(\ell+1)}\; p\Big(b_i^{(\ell+1)}\Big)\right] \left[\prod_{i=1}^{n_{\ell+1}}\prod_{j=1}^{n_\ell} dW_{ij}^{(\ell+1)}\; p\Big(W_{ij}^{(\ell+1)}\Big)\right] \times \left[\prod_{i=1}^{n_{\ell+1}}\prod_{\alpha\in\mathcal{D}} \delta\left(z_{i;\alpha}^{(\ell+1)} - b_i^{(\ell+1)} - \sum_{j=1}^{n_\ell} W_{ij}^{(\ell+1)}\sigma\Big(z_{j;\alpha}^{(\ell)}\Big)\right)\right], \tag{2.34}$$

which determines the distribution of the preactivations in the $(\ell+1)$-th layer, conditioned on the preactivations in the ℓ-th layer, after integrating out the model parameters.

More generally, for any parameterized model with output $z^{\text{out}}_{i;\alpha} \equiv f_i(x_\alpha;\theta)$ for $i = 1,\ldots,n_{\text{out}}$ and with the model parameters θ_μ distributed according to $p(\theta)$, the output distribution (2.24) can be written using the Dirac delta function as

$$p\Big(z^{\text{out}}\Big|\mathcal{D}\Big) = \int \left[\prod_{\mu=1}^{P} d\theta_\mu\right] p(\theta) \left[\prod_{i=1}^{n_{\text{out}}}\prod_{\alpha\in\mathcal{D}} \delta\Big(z^{\text{out}}_{i;\alpha} - f_i(x_\alpha;\theta)\Big)\right]. \tag{2.35}$$

Our pretraining was designed precisely to prepare ourselves for performing this integral for MLPs.

Chapter 3

Effective Theory of Deep Linear Networks at Initialization

> *. . . a system which has spherical symmetry . . . certainly cannot result in an organism such as a horse, which is not spherically symmetrical.*
>
> Alan Turing, on the limitations of toy models [24].

In this final warm-up chapter, we introduce and then solve a toy model of deep learning, the **deep linear network**.[1] As will be explained in §3.1, the deep linear network is simply an MLP with `linear` activation functions. In particular, such a network can only compute linear transformations of its inputs and certainly cannot result in a function such as a human, which is empirically known to be nonlinear. Nonetheless, the study of deep linear networks will serve as a useful blueprint for an *effective theory of deep learning* that we will develop more generally over the subsequent chapters. Specifically, the exercises in this chapter illustrate how layer-to-layer recursions control the statistics of deep neural networks in a very intuitive way, without getting bogged down by all the technical details.

To that end, in §3.2 we obtain and then exactly solve a layer-to-layer recursion for the two-point correlator of preactivations in deep linear networks. The result highlights that the statistics of the network sensitively depend on the setting of the *initialization hyperparameters*, with the sensitivity increasing exponentially with depth. This leads to the important concept of *criticality*, which we will explore in §5 in greater depth and sensitivity. In short, we learn that for networks to be well behaved, these hyperparameters need to be finely tuned.

Next, in §3.3 we obtain and then solve a layer-to-layer recursion for the four-point correlator, albeit for a single input to further simplify the algebra. This showcases the way in which the behavior of the network can depend on the *architecture hyperparameters*, particularly the width and depth of the network. In addition, we interpret the

[1] For physicists, we give an analogy: the deep linear network is to deep learning as the simple harmonic oscillator is to quantum mechanics.

four-point *connected* correlator as a measurement of the *fluctuation* of the network function from draw to draw of the model parameters. Such fluctuations can interfere with the tuning of the initialization hyperparameters and need to be controlled so that networks behave reliably for typical draws. The scale of the fluctuations are set by the depth-to-width ratio of the network, highlighting this important *emergent scale* in the analysis of MLPs, and we'll see that the fluctuations can be kept under control by keeping the depth-to-width ratio of the network sufficiently small.

Finally, in §3.4 we obtain a recursion for an arbitrary M-point correlator for a deep linear network evaluated on a single input. Such recursions are all exactly solvable at any width n and depth L, meaning we can fully determine the statistics of these networks at initialization.[2] Given these nonperturbative solutions, we take the limit of large width, with fixed depth, and the limit of large depth, with fixed width, and show explicitly that these two limits do not commute. We also construct an interpolating solution with both large width and large depth, but fixed depth-to-width ratio L/n, and see how this *scale* serves as a perturbative parameter that controls all the interactions in the network and controls the validity of the perturbative analysis.

3.1 Deep Linear Networks

A deep linear network iteratively transforms an input $x_{i;\alpha}$ through a sequence of simple linear transformations

$$z_{i;\alpha}^{(\ell+1)} = b_i^{(\ell+1)} + \sum_{j=1}^{n_\ell} W_{ij}^{(\ell+1)} z_{j;\alpha}^{(\ell)} \,, \tag{3.1}$$

with $z_{i;\alpha}^{(0)} \equiv x_{i;\alpha}$ and $z_{i;\alpha}^{(\ell)} \equiv z_i^{(\ell)}(x_\alpha)$. Since the `linear` activation function is the identity function, $\sigma(z) = z$, there's no distinction here between preactivations and activations.

In this chapter, we'll simplify matters a bit by turning off all the biases, $b_i^{(\ell)} = 0$, so that the preactivations in layer ℓ are simply given by a repeated matrix multiplication of weight matrices as

$$z_{i;\alpha}^{(\ell)} = \sum_{j_0=1}^{n_0} \sum_{j_1=1}^{n_1} \cdots \sum_{j_{\ell-1}=1}^{n_{\ell-1}} W_{ij_{\ell-1}}^{(\ell)} W_{j_{\ell-1}j_{\ell-2}}^{(\ell-1)} \cdots W_{j_1 j_0}^{(1)} x_{j_0;\alpha} \equiv \sum_{j=1}^{n_0} \mathcal{W}_{ij}^{(\ell)} x_{j;\alpha} \,. \tag{3.2}$$

Here we have introduced an n_ℓ-by-n_0 matrix

$$\mathcal{W}_{ij}^{(\ell)} = \sum_{j_1=1}^{n_1} \cdots \sum_{j_{\ell-1}=1}^{n_{\ell-1}} W_{ij_{\ell-1}}^{(\ell)} W_{j_{\ell-1}j_{\ell-2}}^{(\ell-1)} \cdots W_{j_1 j}^{(1)} \,, \tag{3.3}$$

which highlights the fact that the preactivation at the ℓ-th layer is simply a linear transformation of the input. Additionally, let us set $C_W^{(\ell)} \equiv C_W$ so that the order-one

[2] This notion of *solve* should not be confused with the solving of the training dynamics for a particular learning algorithm. In the context of deep linear networks, the dynamics of gradient descent were analyzed in [25]. In §10 and §∞, we will solve the training dynamics of gradient descent for MLPs with general activation functions in the context of our effective theory formalism.

part of the weight variance is layer independent. All together, this means that the initialization distribution over the weights is characterized by the following expectations

$$\mathbb{E}\left[W_{ij}^{(\ell)}\right] = 0\,, \qquad \mathbb{E}\left[W_{i_1 j_1}^{(\ell)} W_{i_2 j_2}^{(\ell)}\right] = \delta_{i_1 i_2}\delta_{j_1 j_2}\frac{C_W}{n_{\ell-1}}\,. \tag{3.4}$$

Somewhat counterintuitively, deep linear networks generically represent a smaller set of functions than fully general linear transformations, a.k.a. one-layer networks of the same input-output dimensions.[3] As an extreme example, let's take a two-layer deep linear network in which the first hidden layer consists of a single neuron $n_1 = 1$ and consider the network output in the second layer $\ell = 2$. In this case, all the information in the input is compressed through a *bottleneck* into a single number in the first layer before being converted into an n_2-dimensional vector in the output layer. Surely, such a deep linear network represents a tinier subspace of linear transformations than those given by all the possible n_2-by-n_0 matrices, so long as $n_0, n_2 > 1$.

More importantly, we will show that the statistics of deep linear networks at initialization are also very different from those of one-layer networks. In particular, while the statistics of each $W_{ij}^{(\ell)}$ are given by a simple Gaussian distribution, the statistics of their product $\mathcal{W}_{ij}^{(\ell)}$ are non-Gaussian, depending in a complicated way on the depth ℓ and widths $n_1, \ldots, n_\ell$ of the network.

The goal of the rest of this chapter is to exactly work out this dependence. Concretely, we are going to compute the nontrivial distribution

$$p\Big(z^{(\ell)}\Big|\mathcal{D}\Big) \equiv p\Big(z^{(\ell)}\left(x_1\right), \ldots, z^{(\ell)}\left(x_{N_\mathcal{D}}\right)\Big)\,, \tag{3.5}$$

of the preactivations $z_{i;\alpha}^{(\ell)} \equiv z_i^{(\ell)}(x_\alpha)$ implied by the iterated multiplication (3.2) when evaluated on the entire dataset $\mathcal{D}$. As mentioned in §1.2, a distribution is completely determined by the set of all its M-point correlators, and so our method for determining $p\left(z^{(\ell)}\middle|\mathcal{D}\right)$ will be to directly compute these correlators.

Before moving onto the next section, let's consider the simplest observable, the mean of the preactivation $z_{i;\alpha}^{(\ell)}$. Taking an expectation of the defining equation (3.2), it's easy to see that the mean preactivation must vanish at any layer:

$$\begin{aligned}\mathbb{E}\left[z_{i;\alpha}^{(\ell)}\right] &= \sum_{j_0=1}^{n_0}\sum_{j_1=1}^{n_1}\cdots\sum_{j_{\ell-1}=1}^{n_{\ell-1}} \mathbb{E}\left[W_{ij_{\ell-1}}^{(\ell)} W_{j_{\ell-1}j_{\ell-2}}^{(\ell-1)}\cdots W_{j_1 j_0}^{(1)} x_{j_0;\alpha}\right] \\ &= \sum_{j_0=1}^{n_0}\sum_{j_1=1}^{n_1}\cdots\sum_{j_{\ell-1}=1}^{n_{\ell-1}} \mathbb{E}\left[W_{ij_{\ell-1}}^{(\ell)}\right]\mathbb{E}\left[W_{j_{\ell-1}j_{\ell-2}}^{(\ell-1)}\right]\cdots\mathbb{E}\left[W_{j_1 j_0}^{(1)}\right] x_{j_0;\alpha} = 0\,,\end{aligned} \tag{3.6}$$

[3]This is not necessarily a bad thing, since there are often both computational and representational advantages to focusing on a specialized class of functions. For instance, we saw that convolutional networks represent a much smaller set of functions than MLPs, and yet they are known to perform better on computer vision tasks due to their translational-invariance-respecting *inductive bias* as well as the fact that they require significantly less computation due to their sparse pattern of connections. Having said that, it's not obvious if deep linear networks have a useful inductive bias when compared to general linear transformations.

since the weight matrices are mutually independent – and independent of the input – and have zero mean (3.4). By a similar argument, it's easy to see that any odd-point correlator of preactivations will vanish as well. Thus, going forward, we will only have to concern ourselves with the even-point correlators.

3.2 Criticality

Since the mean is trivial, the next simplest candidate for an interesting observable is the two-point correlator $\mathbb{E}\left[z^{(\ell)}_{i_1;\alpha_1} z^{(\ell)}_{i_2;\alpha_2}\right]$, which quantifies the typical magnitudes of the preactivations. We'll first go through the math, and then we'll discuss the physics.

Math: recursion for the two-point correlator

Let's start slowly by first considering the two-point correlator in the first layer. Using the defining equation (3.2) to express the first-layer preactivations in terms of the inputs as

$$z^{(1)}_{i;\alpha} = \sum_{j}^{n_0} W^{(1)}_{ij} x_{j;\alpha} \,, \tag{3.7}$$

we can express the two-point correlator as

$$\begin{aligned} \mathbb{E}\left[z^{(1)}_{i_1;\alpha_1} z^{(1)}_{i_2;\alpha_2}\right] &= \sum_{j_1,j_2=1}^{n_0} \mathbb{E}\left[W^{(1)}_{i_1 j_1} x_{j_1;\alpha_1} W^{(1)}_{i_2 j_2} x_{j_2;\alpha_2}\right] \\ &= \sum_{j_1,j_2=1}^{n_0} \mathbb{E}\left[W^{(1)}_{i_1 j_1} W^{(1)}_{i_2 j_2}\right] x_{j_1;\alpha_1} x_{j_2;\alpha_2} \\ &= \sum_{j_1,j_2=1}^{n_0} \frac{C_W}{n_0} \delta_{i_1 i_2} \delta_{j_1 j_2} x_{j_1;\alpha_1} x_{j_2;\alpha_2} = \delta_{i_1 i_2} C_W \frac{1}{n_0} \sum_{j=1}^{n_0} x_{j;\alpha_1} x_{j;\alpha_2} \,, \end{aligned} \tag{3.8}$$

where to go from the second line to the third line we Wick-contracted the two weights and inserted the variance (3.4). Additionally, let us introduce the notation

$$G^{(0)}_{\alpha_1\alpha_2} \equiv \frac{1}{n_0} \sum_{i=1}^{n_0} x_{i;\alpha_1} x_{i;\alpha_2} \,, \tag{3.9}$$

for the inner product of the two inputs, normalized by the input dimension n_0. In terms of this object, we can rewrite the first-layer two-point correlator (3.8) as

$$\mathbb{E}\left[z^{(1)}_{i_1;\alpha_1} z^{(1)}_{i_2;\alpha_2}\right] = \delta_{i_1 i_2} C_W G^{(0)}_{\alpha_1\alpha_2} \,. \tag{3.10}$$

Next, we could mindlessly repeat the same exercise to get the two-point correlator in any arbitrary layer, using the defining equation (3.2) to express $z^{(\ell)}_{i;\alpha}$ in terms of the input. Instead, in order to practice our recursive approach, let's evaluate the two-point correlator recursively. To do so, we inductively assume that the two-point correlator at

the ℓ-th layer is known and then derive the two-point correlator at the $(\ell+1)$-th layer. Using the iteration equation (3.1) with the bias set to zero, we find

$$\begin{aligned}\mathbb{E}\left[z^{(\ell+1)}_{i_1;\alpha_1} z^{(\ell+1)}_{i_2;\alpha_2}\right] &= \sum_{j_1,j_2=1}^{n_\ell} \mathbb{E}\left[W^{(\ell+1)}_{i_1 j_1} W^{(\ell+1)}_{i_2 j_2} z^{(\ell)}_{j_1;\alpha_1} z^{(\ell)}_{j_2;\alpha_2}\right] \\ &= \sum_{j_1,j_2=1}^{n_\ell} \mathbb{E}\left[W^{(\ell+1)}_{i_1 j_1} W^{(\ell+1)}_{i_2 j_2}\right] \mathbb{E}\left[z^{(\ell)}_{j_1;\alpha_1} z^{(\ell)}_{j_2;\alpha_2}\right] \\ &= \delta_{i_1 i_2} C_W \frac{1}{n_\ell} \sum_{j=1}^{n_\ell} \mathbb{E}\left[z^{(\ell)}_{j;\alpha_1} z^{(\ell)}_{j;\alpha_2}\right] ,\end{aligned} \tag{3.11}$$

where to go from the first line to the second line we used the fact that the weights $W^{(\ell+1)}$ of the $(\ell+1)$-th layer are statistically independent from the preactivations $z^{(\ell)}$ in the ℓ-th layer, and to go from the second line to the third line we Wick-contracted the two weights and substituted in the variance (3.4). Notice that at *any* layer, the two-point correlator is proportional to the Kronecker delta $\delta_{i_1 i_2}$, vanishing unless the neural indices i_1 and i_2 are the same. With that in mind, let us decompose the two-point correlator as

$$\mathbb{E}\left[z^{(\ell)}_{i_1;\alpha_1} z^{(\ell)}_{i_2;\alpha_2}\right] \equiv \delta_{i_1 i_2} G^{(\ell)}_{\alpha_1\alpha_2} , \tag{3.12}$$

and introduce a generalization of the above notation (3.9) for an arbitrary layer ℓ. Multiplying this equation by $\delta_{i_1 i_2}$, summing over $i_1, i_2 = 1, \ldots, n_\ell$ and dividing it by n_ℓ, the quantity $G^{(\ell)}_{\alpha_1\alpha_2}$ can also be expressed as

$$G^{(\ell)}_{\alpha_1\alpha_2} = \frac{1}{n_\ell} \sum_{j=1}^{n_\ell} \mathbb{E}\left[z^{(\ell)}_{j;\alpha_1} z^{(\ell)}_{j;\alpha_2}\right] , \tag{3.13}$$

and can thus be thought of as the average inner-product of preactivations in the ℓ-th layer, divided by the number of neurons in the layer n_ℓ. This inner product depends on sample indices *only* and lets us interpret $G^{(\ell)}_{\alpha_1\alpha_2} \equiv G^{(\ell)}\left(x_{\alpha_1}, x_{\alpha_2}\right)$ as the covariance of the two inputs, x_{α_1} and x_{α_2}, after passing through an ℓ-layer deep linear network.

With all this notation introduced and fully interpreted, it's easy to see that the above recursion (3.11) can be compactly summarized by

$$G^{(\ell+1)}_{\alpha_1\alpha_2} = C_W G^{(\ell)}_{\alpha_1\alpha_2} , \tag{3.14}$$

which describes how the covariance $G^{(\ell)}_{\alpha_1\alpha_2}$ evolves from layer to layer. Apparently, to transform the covariance from layer ℓ to layer $\ell+1$, we simply multiply by the constant C_W. The initial condition $G^{(0)}_{\alpha_1\alpha_2}$ is given by the inner product of the two inputs (3.9), and the solution is an exponential

$$G^{(\ell)}_{\alpha_1\alpha_2} = \left(C_W\right)^\ell G^{(0)}_{\alpha_1\alpha_2} , \tag{3.15}$$

as is typical for a repeated application of matrix multiplication. Note that the factor of the width n_ℓ in the variance of the weights (3.4) nicely dropped out, indicating that this was in fact the proper way to scale the variance.

Physics: criticality

Already at this point our analysis illustrates an interesting and very general phenomenon. Considering the solution (3.15), generically one of two things happens. If $C_W > 1$, the covariance blows up exponentially, quickly being driven to a fixed point $G^{\star}_{\alpha_1\alpha_2} = \infty$ for all pairs of inputs and leading to a divergent network output. If $C_W < 1$, the covariance exponentially decays to a fixed point $G^{\star}_{\alpha_1\alpha_2} = 0$ for all pairs of inputs, quickly curtailing any data dependence in the network output. Any time an observable approaches a value exponentially quickly, we'll refer to the limiting value as a **trivial fixed point**. The value $G^{\star}_{\alpha_1\alpha_2} = \infty$ associated with $C_W > 1$ and the value $G^{\star}_{\alpha_1\alpha_2} = 0$ associated with $C_W < 1$ are prime examples of a trivial fixed point.

Exploring this further, first note that the diagonal part of the covariance at the output layer L estimates the typical magnitude of the output for a given input $x_{i;\alpha}$

$$G^{(L)}_{\alpha\alpha} = \mathbb{E}\left[\frac{1}{n_L}\sum_{j=1}^{n_L}\left(z^{(L)}_{j;\alpha}\right)^2\right] . \tag{3.16}$$

With this observable in mind, the aforementioned exponential behavior should immediately set off alarm bells, signaling either some sort of numerical instability ($C_W > 1$) or loss of information ($C_W < 1$). In addition, note that the target values for the different components of the network output are typically $O(1)$ numbers, neither exponentially large nor small. Such exponential behavior of the network should thus make it extremely difficult to learn to approximate the desired function. In this way, this *exploding and vanishing covariance problem* is a baby version of the infamous exploding and vanishing gradient problem – a well-known obstacle to gradient-based training of deep networks – which we shall make more precise in §9.

However, we were actually a little too quick in our analysis before: what happens if we tune the weight variance C_W so that it's precisely equal to 1? This is clearly a special point in the hyperparameter space of initialization, separating the exponentially growing solution from the exponentially decaying solution. Going back to the recursion (3.14), we see that if $C_W = 1$ then the covariance is fixed $G^{(\ell)}_{\alpha_1\alpha_2} = G^{(0)}_{\alpha_1\alpha_2} \equiv G^{\star}_{\alpha_1\alpha_2}$, manifestly preserving the full covariance of the input data even after passing through many layers of the deep linear network. This is a bona fide **nontrivial fixed point**, as it doesn't exponentially trivialize the structure of input data. Thus, at least at this heuristic level of analysis, choosing $C_W = 1$ appears to be essential for preserving the structure of the input data in a numerically stable manner. More generally, flowing to a nontrivial fixed point seems to be a necessary condition for deep networks to do anything useful.

When we fine-tune the initialization hyperparameters of a network so that the covariance avoids exponential behavior, we'll call them **critical initialization hyperparameters**.[4] For deep linear networks, the critical initialization hyperparameter $C_W = 1$ separates two regimes, one with an exponentially growing covariance for $C_W > 1$, and

[4]This word choice is motivated by the analogy to critical phenomena in statistical physics. For instance, consider the prototypical example: a magnet made of iron. At high temperature, the magnetic moments – or spins – of the iron atoms point in random directions, leading to a paramagnetic phase

the other with an exponentially decaying covariance for $C_W < 1$. When the weight variance is tuned to criticality $C_W = 1$, the network has a perfect self-similarity of the covariance, preserving it exactly through the evolution from layer to layer.

In §5, we will extend our analysis of **criticality** to MLPs that use any particular activation function. And, as shall be seen further on in §10 and §∞, tuning a network to criticality is *critical* for any *deep* network to be well behaved and perform useful tasks – at least without otherwise employing ad-hoc tricks to ensure that signals can propagate stably.

3.3 Fluctuations

Recall from §1 that if a distribution is Gaussian and has a zero mean, then the covariance completely specifies the distribution. If the preactivation distribution $p\left(z^{(\ell)}\middle|\mathcal{D}\right)$ were Gaussian, this would mean that the critical tuning of the one initialization hyperparameter $C_W = 1$ would be sufficient to ensure that any observable is well behaved. However, if the distribution $p\left(z^{(\ell)}\middle|\mathcal{D}\right)$ is not Gaussian, then it's not clear a priori whether observables depending on higher-point connected correlators will be well behaved with the same tuning. In principle, such observables could require other tunings of C_W that are incompatible with the critical setting $C_W = 1$ for the covariance $G^{(\ell)}_{\alpha_1\alpha_2}$. To settle this question, let's look at the next simplest observable, the connected four-point correlator. As before, we'll go through the math first and discuss the physics second.

In this section and the next, to simplify the algebra we'll focus on correlators of preactivations that are evaluated only on a single input $x_\alpha = x$. This is sufficient to qualitatively highlight the importance of the higher-point correlators while letting us avoid the interference of some annoying technical manipulations. Accordingly, in these sections we will drop the sample indices on preactivations and denote the covariance as

$$G_2^{(\ell)} \equiv G^{(\ell)}_{\alpha\alpha} = G^{(\ell)}(x, x)\,. \tag{3.17}$$

In the next chapter, we'll consider the fully general case.

Math: recursion for the four-point correlator

As we did for the two-point correlator in the previous section, we'll begin by working out the four-point correlator in the first layer and then next derive and solve a recursion for the correlator in the deeper layers. First for the first layer, using the defining

without any coherent magnetic field. By contrast, at low temperature, the spins instead try to collectively orient in the same direction, leading to a ferromagnetic phase with coherent magnetic field – think of the ∩-shaped cartoon magnet that children play with. A critical temperature separates these two phases of magnetism, and the magnet set to the critical temperature will exhibit very special behavior that is neither paramagnetism nor ferromagnetism but known as self-similarity.

equation (3.7) with the sample index omitted, we have for the *full* four-point correlator

$$\begin{aligned}
&\mathbb{E}\left[z_{i_1}^{(1)} z_{i_2}^{(1)} z_{i_3}^{(1)} z_{i_4}^{(1)}\right] \qquad (3.18)\\
=&\sum_{j_1,j_2,j_3,j_4=1}^{n_0} \mathbb{E}\left[W_{i_1 j_1}^{(1)} W_{i_2 j_2}^{(1)} W_{i_3 j_3}^{(1)} W_{i_4 j_4}^{(1)}\right] x_{j_1} x_{j_2} x_{j_3} x_{j_4}\\
=&\frac{C_W^2}{n_0^2} \sum_{j_1,j_2,j_3,j_4=1}^{n_0} \left(\delta_{i_1 i_2}\delta_{j_1 j_2}\delta_{i_3 i_4}\delta_{j_3 j_4} + \delta_{i_1 i_3}\delta_{j_1 j_3}\delta_{i_2 i_4}\delta_{j_2 j_4} + \delta_{i_1 i_4}\delta_{j_1 j_4}\delta_{i_2 i_3}\delta_{j_2 j_3}\right) x_{j_1} x_{j_2} x_{j_3} x_{j_4}\\
=&C_W^2 \left(\delta_{i_1 i_2}\delta_{i_3 i_4} + \delta_{i_1 i_3}\delta_{i_2 i_4} + \delta_{i_1 i_4}\delta_{i_2 i_3}\right) \left(G_2^{(0)}\right)^2 .
\end{aligned}$$

where to go from line two to line three, we made three distinct pairings for the two Wick contractions of the four weights, and then used the weight variance (3.4) to evaluate each contraction. To get to the final line, we evaluated the sums over the j indices and then substituted using our definition of the inner product (3.9), which for a single input simply reads

$$G_2^{(0)} = \frac{1}{n_0} \sum_{j=1}^{n_0} x_j x_j \,. \qquad (3.19)$$

Comparing this result (3.18) with the two-point correlator in the first layer (3.10), we note that this answer is precisely what we'd expect for the full four-point correlator if the preactivation distribution were exactly Gaussian. Thus, deep linear networks appear to be simply Gaussian after a single layer, at least at the four-point correlator level of analysis.[5]

This Gaussianity does *not* hold in deeper layers. To see that, let's derive and solve a recursion for the four-point correlator. Beginning with the iteration equation (3.1) with zero bias, we find

$$\begin{aligned}
&\mathbb{E}\left[z_{i_1}^{(\ell+1)} z_{i_2}^{(\ell+1)} z_{i_3}^{(\ell+1)} z_{i_4}^{(\ell+1)}\right] \qquad (3.20)\\
=&\sum_{j_1,j_2,j_3,j_4=1}^{n_\ell} \mathbb{E}\left[W_{i_1 j_1}^{(\ell+1)} W_{i_2 j_2}^{(\ell+1)} W_{i_3 j_3}^{(\ell+1)} W_{i_4 j_4}^{(\ell+1)}\right] \mathbb{E}\left[z_{j_1}^{(\ell)} z_{j_2}^{(\ell)} z_{j_3}^{(\ell)} z_{j_4}^{(\ell)}\right]\\
=&\frac{C_W^2}{n_\ell^2} \sum_{j_1,j_2,j_3,j_4=1}^{n_\ell} \left(\delta_{i_1 i_2}\delta_{j_1 j_2}\delta_{i_3 i_4}\delta_{j_3 j_4} + \delta_{i_1 i_3}\delta_{j_1 j_3}\delta_{i_2 i_4}\delta_{j_2 j_4} + \delta_{i_1 i_4}\delta_{j_1 j_4}\delta_{i_2 i_3}\delta_{j_2 j_3}\right)\\
&\qquad\qquad \times \mathbb{E}\left[z_{j_1}^{(\ell)} z_{j_2}^{(\ell)} z_{j_3}^{(\ell)} z_{j_4}^{(\ell)}\right]\\
=&C_W^2 \left(\delta_{i_1 i_2}\delta_{i_3 i_4} + \delta_{i_1 i_3}\delta_{i_2 i_4} + \delta_{i_1 i_4}\delta_{i_2 i_3}\right) \frac{1}{n_\ell^2} \sum_{j,k=1}^{n_\ell} \mathbb{E}\left[z_j^{(\ell)} z_j^{(\ell)} z_k^{(\ell)} z_k^{(\ell)}\right] ,
\end{aligned}$$

where on the second line we used the independence of the $(\ell+1)$-th-layer weights from the ℓ-th-layer preactivations, on the third line we again made three distinct pairings for

[5] In the next chapter, we'll show very generally that the preactivation distribution is always Gaussian in the first layer.

the two pairs of Wick contractions of the four weights, and on the last line we made judicious use of the Kronecker deltas to collapse the sums.

Now, we see from this recursion that at *any* layer the full four-point correlator is proportional to the factor $(\delta_{i_1 i_2}\delta_{i_3 i_4} + \delta_{i_1 i_3}\delta_{i_2 i_4} + \delta_{i_1 i_4}\delta_{i_2 i_3})$, a fixed tensor structure that specifies the neural-index dependence of the correlator. Thus by decomposing the full four-point correlator as

$$\mathbb{E}\left[z_{i_1}^{(\ell)} z_{i_2}^{(\ell)} z_{i_3}^{(\ell)} z_{i_4}^{(\ell)}\right] \equiv (\delta_{i_1 i_2}\delta_{i_3 i_4} + \delta_{i_1 i_3}\delta_{i_2 i_4} + \delta_{i_1 i_4}\delta_{i_2 i_3})\, G_4^{(\ell)}\,, \tag{3.21}$$

we can put all of the layer dependence into this simpler object $G_4^{(\ell)}$ and not worry about neural indices in our recursion. In terms of this decomposition, the result (3.18) for the correlator in the first layer becomes

$$G_4^{(1)} = C_W^2 \left(G_2^{(0)}\right)^2\,, \tag{3.22}$$

and the final factor in the above recursion (3.20) can be rewritten as

$$\frac{1}{n_\ell^2}\sum_{j,k=1}^{n_\ell} \mathbb{E}\left[z_j^{(\ell)} z_j^{(\ell)} z_k^{(\ell)} z_k^{(\ell)}\right] = \frac{1}{n_\ell^2}\sum_{j,k=1}^{n_\ell} (\delta_{jj}\delta_{kk} + \delta_{jk}\delta_{jk} + \delta_{jk}\delta_{kj})\, G_4^{(\ell)} = \left(1 + \frac{2}{n_\ell}\right) G_4^{(\ell)}\,. \tag{3.23}$$

Using this, the entire recursion above (3.20) can be rewritten simply as a recursion for $G_4^{(\ell)}$ as

$$G_4^{(\ell+1)} = C_W^2 \left(1 + \frac{2}{n_\ell}\right) G_4^{(\ell)}\,. \tag{3.24}$$

This recursion, with the initial condition set by (3.22), has a simple solution

$$\begin{aligned} G_4^{(\ell)} =& C_W^{2\ell}\left[\prod_{\ell'=1}^{\ell-1}\left(1 + \frac{2}{n_{\ell'}}\right)\right]\left(G_2^{(0)}\right)^2 \\ =& \left[\prod_{\ell'=1}^{\ell-1}\left(1 + \frac{2}{n_{\ell'}}\right)\right]\left(G_2^{(\ell)}\right)^2\,, \end{aligned} \tag{3.25}$$

where in the final line we substituted in the solution (3.15) for the covariance. Now let's extract some physics from this compact formula.

Physics: large-n expansion, non-Gaussianities, interactions, and fluctuations

To start, we note that the four-point correlator (3.25) drastically simplifies in the limit of an infinite number of neurons per hidden layer ($n_\ell \to \infty$). In such a limit, the solution (3.25) degenerates to

$$G_4^{(\ell)} = \left(G_2^{(\ell)}\right)^2\,, \tag{3.26}$$

and the full four-point correlator (3.21) becomes

$$\mathbb{E}\left[z_{i_1}^{(\ell)} z_{i_2}^{(\ell)} z_{i_3}^{(\ell)} z_{i_4}^{(\ell)}\right] = (\delta_{i_1 i_2}\delta_{i_3 i_4} + \delta_{i_1 i_3}\delta_{i_2 i_4} + \delta_{i_1 i_4}\delta_{i_2 i_3}) \left(G_2^{(\ell)}\right)^2\,. \tag{3.27}$$

This is exactly what we'd find if the preactivation distribution were Gaussian: the four-point correlator is determined entirely by the two-point correlator, with the tensor structure determined by Wick's theorem. In fact, as we will show in the next chapter, for any MLP with any particular choice of a nonlinear activation function, the preactivation distribution is governed by Gaussian statistics in this infinite-width limit, implying no interactions between the neurons in such a limit. However, despite the rather large computational resources that *big tech* can throw at machine-learning problems, realistic MLPs simply do not have an infinite number of neurons per layer. To understand such realistic MLPs, we'll have to back off this infinite-width limit.

To illustrate this most clearly, let's set all the hidden layer widths to be equal $n_1 = n_2 = \ldots = n_{L-1} \equiv n$. Then, evaluating (3.25), the deviation from the infinite-width limit at the level of four-point correlator statistics is encoded by the difference

$$\begin{aligned} G_4^{(\ell)} - \left(G_2^{(\ell)}\right)^2 &= \left[\left(1+\frac{2}{n}\right)^{\ell-1} - 1\right]\left(G_2^{(\ell)}\right)^2 \\ &= \frac{2(\ell-1)}{n}\left(G_2^{(\ell)}\right)^2 + O\left(\frac{1}{n^2}\right) , \end{aligned} \tag{3.28}$$

where in the last line we expanded in $1/n$ and kept the leading correction to the infinite-width limit.[6] In particular, at criticality where $G_2^{(\ell)}$ is constant, this leading correction (3.28) scales inversely proportionally with the width and proportionally with the depth. Thus, the deviation from infinite width is proportional to the depth-to-width ratio of the network, our first encounter with this important **emergent scale**. There are multiple ways to think about this finite-width correction.

First, the connected four-point correlator (1.54) is given by

$$\mathbb{E}\left[z_{i_1}^{(\ell)} z_{i_2}^{(\ell)} z_{i_3}^{(\ell)} z_{i_4}^{(\ell)}\right]\Big|_{\text{connected}} = \left(\delta_{i_1 i_2}\delta_{i_3 i_4} + \delta_{i_1 i_3}\delta_{i_2 i_4} + \delta_{i_1 i_4}\delta_{i_2 i_3}\right)\left[G_4^{(\ell)} - \left(G_2^{(\ell)}\right)^2\right] , \tag{3.29}$$

which directly connects the difference (3.28) to our measure of non-Gaussianity for the distribution. We see that the non-Gaussianity grows as the network deepens, and the preactivation statistics in layer ℓ are *nearly-Gaussian* so long as the emergent scale, the depth-to-width-ratio, remains perturbatively small. From the action perspective, this means that the quartic coupling changes – or **runs** – as the layer at which we consider the preactivation distribution changes, with the coupling growing in proportion with layer ℓ.

Second, in §1.3 we gave another interpretation for a nonzero connected four-point correlator as measuring interactions – i.e. the breakdown of statistical independence – between the different components of the random vector. To be very specific, let us look at a particular entry of the connected four-point correlator tensor with $i_1 = i_2 = j$ and $i_3 = i_4 = k$ for $j \neq k$. This entry can be expressed as

$$\mathbb{E}\left[\left(z_j^{(\ell)} z_j^{(\ell)} - G_2^{(\ell)}\right)\left(z_k^{(\ell)} z_k^{(\ell)} - G_2^{(\ell)}\right)\right] = G_4^{(\ell)} - \left(G_2^{(\ell)}\right)^2 , \quad \text{for } j \neq k . \tag{3.30}$$

[6] This approximation is valid so long as the depth of the network doesn't grow too large. Stay tuned for the analysis in the next section where we will discuss how this limit breaks down.

This shows that the deviation of $z_j^{(\ell)} z_j^{(\ell)}$ from its mean value $\mathbb{E}\left[z_j^{(\ell)} z_j^{(\ell)}\right] = G_2^{(\ell)}$ on a particular neuron j is correlated with the same deviation from the mean on a different neuron k. We can thus interpret the finite-width difference (3.28) as controlling intralayer interactions between distinct neurons, with the strength of the interactions growing with depth.

Third, we can see that some observables that are deterministic in the infinite-width limit start to fluctuate at finite width. To this end, let us consider the simple observable

$$\mathcal{O}^{(\ell)} \equiv \mathcal{O}\left(z^{(\ell)}\right) \equiv \frac{1}{n} \sum_{j=1}^{n} z_j^{(\ell)} z_j^{(\ell)} , \quad \text{for } \ell < L , \tag{3.31}$$

which captures the average magnitude of the preactivations over all the different neurons in a hidden layer ℓ for a given instantiation of the network weights. Its mean over different realizations of the weights is given by the expectation

$$\mathbb{E}\left[\mathcal{O}^{(\ell)}\right] = \frac{1}{n} \sum_{j=1}^{n} \mathbb{E}\left[z_j^{(\ell)} z_j^{(\ell)}\right] = G_2^{(\ell)} , \tag{3.32}$$

and the magnitude of this observable's fluctuation from instantiation to instantiation is measured by its variance

$$\begin{aligned} \mathbb{E}\left[\left(\mathcal{O}^{(\ell)} - \mathbb{E}\left[\mathcal{O}^{(\ell)}\right]\right)^2\right] =& \frac{1}{n^2} \sum_{j,k=1}^{n} \mathbb{E}\left[z_j^{(\ell)} z_j^{(\ell)} z_k^{(\ell)} z_k^{(\ell)}\right] - \left(G_2^{(\ell)}\right)^2 \\ =& \frac{1}{n^2} \sum_{j,k=1}^{n} \left(\delta_{jj}\delta_{kk} + \delta_{jk}\delta_{jk} + \delta_{jk}\delta_{kj}\right) G_4^{(\ell)} - \left(G_2^{(\ell)}\right)^2 \\ =& \frac{2}{n} G_4^{(\ell)} + \left[G_4^{(\ell)} - \left(G_2^{(\ell)}\right)^2\right] \\ =& \frac{2\ell}{n} \left(G_2^{(\ell)}\right)^2 + O\left(\frac{1}{n^2}\right) , \end{aligned} \tag{3.33}$$

where in the last step we recalled the expansion (3.28) for the finite-width difference. As promised, $\mathcal{O}^{(\ell)}$ is deterministic at infinite width, since this variance is suppressed by $1/n$ and vanishes identically in the infinite-width limit. However, as we back off the infinite-width limit, the variance grows linearly with depth at criticality due to the finite-width correction (3.28). As such depth increases, the fluctuation becomes larger, meaning that the *typical* magnitude of the preactivations $\mathcal{O}^{(\ell)}$ measured on any given realization of the deep linear network may deviate more from the mean value $\mathbb{E}\left[\mathcal{O}^{(\ell)}\right] = G_2^{(\ell)}$.

All these finite-width effects – be they non-Gaussianities, intralayer interactions, or finite-width fluctuations – are proportional to the depth-to-width ratio of the network. This is perhaps the most important recurring theme of the book: the leading finite-width contributions at criticality grow linearly with depth, despite being suppressed by the inverse of the layer widths. Since the depths of a real networks with at least one hidden layer are bounded from below as $L \geq 2$ – that is, at minimum such networks

have one hidden layer and one output layer – in practice, networks of any finite size will express some amount of finite-width effects in their output distribution proportional to their aspect ratio L/n. As we will see later in §5, this emergent scaling will hold very generally at criticality for networks with any particular activation function.

Thus, the deeper a network is, the less the infinite-width Gaussian description will apply, due to accumulation of finite-width fluctuations. This is actually a good thing because, as we shall emphasize more in §11, infinite-width networks do not have correlations among neurons within a layer and cannot learn nontrivial representations from input data. Real useful deep learning systems that are used in practice do both of these things, and our later analysis will show that deeper networks have the capacity to do more of these things.

Depth, however, is a double-edged sword. As the overall depth L of a network becomes comparable to its hidden-layer width, fluctuations can begin to dominate. In particular, such extremely deep networks will have a huge variation in observables from instantiation to instantiation. Thus, even if we choose the critical initialization hyperparameter $C_W = 1$, in some instantiations signals blow up, in other instantiations signals decay, and rarely do they stay tamed to be of order one. From a practical point of view, these networks are pretty useless.

This set of circumstances is actually very fortuitous from a theorist's vantage point: our *effective theory of deep learning* is most accurate when the aspect ratio L/n of the network is small but nonzero – due to the applicability of the perturbative large-width expansion – and this is exactly the setting of these architecture hyperparameters where networks work best in practice. In fact, one could expect that balancing the utility of nonzero depth for learning features against the cost of growing fluctuations could result in some optimal aspect ratio L/n for MLPs of a particular activation, just as we saw that there is a correct tuning for the initialization hyperparameter C_W for deep linear networks. We will return to this question of tuning L/n when we discuss inductive bias in §6 after first redoing our analysis of criticality and fluctuations for arbitrary activation functions in the following two chapters, §4 and §5. In particular, in these chapters we will understand how the statistics of the preactivations run with depth, and see the emergence of the depth-to-width ratio as a scale that controls the validity of the perturbative $1/n$ expansion, as was the case here for deep linear networks.

Quite generally, in the regime where perturbation theory works, the finite-width corrections grow linearly – *not* exponentially – with depth and the network remains well behaved. By contrast, when the depth-to-width ratio becomes large, perturbation theory breaks down, making it very difficult to analyze such networks. However, in the special case of deep linear networks a nonperturbative analysis is possible. In the next section we'll illustrate explicitly what happens to deep linear networks when the depth-to-width ratio grows very large in order to paint an intuitive picture of the way networks behave in this regime.

3.4 Chaos

In the last two sections we used the method of Wick contractions to derive recursions for the two-point and four-point correlators of deep linear networks, which we then easily solved. Now, we will use this same method to compute all the higher-order correlators in order to complete our goal of determining the full distribution $p\left(z^{(\ell)}\middle|\mathcal{D}\right)$. So that we may first simplify the algebra and then focus on the interesting properties of this distribution, we'll again evaluate the correlators only on a single input $x_\alpha = x$ and drop the sample indices in all of the following equations. Math then physics.

Math: recursions for six-point and higher-point correlators

Starting with the first layer, let's compute a general $2m$-point *full* correlator. As this involves many Wick contractions, it might be helpful to remind yourself of the formal statement of Wick's theorem by flipping back to §1.1 and consulting (1.45).... Good.

Now, using the defining equation (3.7) to express the first-layer preactivations in terms of the input, we get

$$\begin{aligned}
&\mathbb{E}\left[z_{i_1}^{(1)} z_{i_2}^{(1)} \cdots z_{i_{2m-1}}^{(1)} z_{i_{2m}}^{(1)}\right] \\
&= \sum_{j_1,\ldots,j_{2m}=1}^{n_0} \mathbb{E}\left[W_{i_1 j_1}^{(1)} W_{i_2 j_2}^{(1)} \cdots W_{i_{2m-1} j_{2m-1}}^{(1)} W_{i_{2m} j_{2m}}^{(1)}\right] x_{j_1} x_{j_2} \cdots x_{j_{2m-1}} x_{j_{2m}} \\
&= \left(\sum_{\text{all parings}} \delta_{i_{k_1} i_{k_2}} \cdots \delta_{i_{k_{2m-1}} i_{k_{2m}}}\right) C_W^m \left(G_2^{(0)}\right)^m \\
&= \left(\sum_{\text{all parings}} \delta_{i_{k_1} i_{k_2}} \cdots \delta_{i_{k_{2m-1}} i_{k_{2m}}}\right) \left(G_2^{(1)}\right)^m ,
\end{aligned} \tag{3.34}$$

where, as before we used Wick's theorem to determine the Wick contractions and then evaluated each contraction by substituting in (3.4) for the variance. Here, the sum is over all the possible pairing of the $2m$ auxiliary indices, $k_1, \ldots, k_{2m}$, resulting in $(2m-1)!!$ distinct terms, and on the final line we substituted in the solution (3.15) for the first-layer covariance.

The result (3.34) confirms what we suspected in the last section, that the preactivation distribution for the first layer is completely Gaussian. If this isn't clear by inspection, it's easy to check directly – via basically the same application of Wick's theorem – that the correlators (3.34) are precisely the $2m$-point correlators of a Gaussian distribution with zero mean and variance $\delta_{i_1 i_2} G_2^{(1)}$. In other words, the preactivation distribution in the first layer is governed by the quadratic action

$$S\left(z^{(1)}\right) = \frac{1}{2G_2^{(1)}} \sum_{i=1}^{n_1} z_i^{(1)} z_i^{(1)} . \tag{3.35}$$

Before presenting a recursion for general $2m$-point correlators, let us work out the recursion for the six-point correlator in detail. Beginning with the iteration equation (3.1) with the bias set to zero, we find

$$
\begin{aligned}
&\mathbb{E}\left[z_{i_1}^{(\ell+1)} z_{i_2}^{(\ell+1)} z_{i_3}^{(\ell+1)} z_{i_4}^{(\ell+1)} z_{i_5}^{(\ell+1)} z_{i_6}^{(\ell+1)}\right] \qquad (3.36)\\
&= \sum_{j_1,j_2,j_3,j_4,j_5,j_6=1}^{n_\ell} \mathbb{E}\left[W_{i_1 j_1}^{(\ell+1)} W_{i_2 j_2}^{(\ell+1)} W_{i_3 j_3}^{(\ell+1)} W_{i_4 j_4}^{(\ell+1)} W_{i_5 j_5}^{(\ell+1)} W_{i_6 j_6}^{(\ell+1)}\right] \mathbb{E}\left[z_{j_1}^{(\ell)} z_{j_2}^{(\ell)} z_{j_3}^{(\ell)} z_{j_4}^{(\ell)} z_{j_5}^{(\ell)} z_{j_6}^{(\ell)}\right] \\
&= C_W^3 \Big(\delta_{i_1 i_2}\delta_{i_3 i_4}\delta_{i_5 i_6} + \delta_{i_1 i_3}\delta_{i_2 i_4}\delta_{i_5 i_6} + \delta_{i_1 i_4}\delta_{i_2 i_3}\delta_{i_5 i_6} \\
&\quad + \delta_{i_1 i_2}\delta_{i_3 i_5}\delta_{i_4 i_6} + \delta_{i_1 i_3}\delta_{i_2 i_5}\delta_{i_4 i_6} + \delta_{i_1 i_5}\delta_{i_2 i_3}\delta_{i_4 i_6} \\
&\quad + \delta_{i_1 i_2}\delta_{i_5 i_4}\delta_{i_3 i_6} + \delta_{i_1 i_5}\delta_{i_2 i_4}\delta_{i_3 i_6} + \delta_{i_1 i_4}\delta_{i_2 i_5}\delta_{i_3 i_6} \\
&\quad + \delta_{i_1 i_5}\delta_{i_3 i_4}\delta_{i_2 i_6} + \delta_{i_1 i_3}\delta_{i_5 i_4}\delta_{i_2 i_6} + \delta_{i_1 i_4}\delta_{i_5 i_3}\delta_{i_2 i_6} \\
&\quad + \delta_{i_5 i_2}\delta_{i_3 i_4}\delta_{i_1 i_6} + \delta_{i_5 i_3}\delta_{i_2 i_4}\delta_{i_1 i_6} + \delta_{i_5 i_4}\delta_{i_2 i_3}\delta_{i_1 i_6} \Big) \frac{1}{n_\ell^3} \sum_{i,j,k=1}^{n_\ell} \mathbb{E}\left[z_i^{(\ell)} z_i^{(\ell)} z_j^{(\ell)} z_j^{(\ell)} z_k^{(\ell)} z_k^{(\ell)}\right] ,
\end{aligned}
$$

noting again the independence of the $(\ell+1)$-th layer weights from the ℓ-th layer preactivations. On the final line, we see that there were fifteen distinct ways to make the three Wick contractions of six weights.

As we saw for the four-point correlator, the structure of neural indices for the full six-point correlator is the same for *any* layer and proportional to a constant tensor, given by the object in the parenthesis above with all those Kronecker deltas. This suggests a decomposition of the six-point correlator as

$$
\mathbb{E}\left[z_{i_1}^{(\ell)} z_{i_2}^{(\ell)} z_{i_3}^{(\ell)} z_{i_4}^{(\ell)} z_{i_5}^{(\ell)} z_{i_6}^{(\ell)}\right] \equiv \left(\delta_{i_1 i_2}\delta_{i_3 i_4}\delta_{i_5 i_6} + \ldots + \delta_{i_5 i_4}\delta_{i_2 i_3}\delta_{i_1 i_6}\right) G_6^{(\ell)} , \qquad (3.37)
$$

with the neural-dependence encapsulated by that complicated sum-over-products of Kronecker deltas and the layer dependence captured solely by $G_6^{(\ell)}$.

Now, to find a recursion for $G_6^{(\ell)}$, we need to perform the sum

$$
\frac{1}{n_\ell^3} \sum_{i,j,k=1}^{n_\ell} \mathbb{E}\left[z_i^{(\ell)} z_i^{(\ell)} z_j^{(\ell)} z_j^{(\ell)} z_k^{(\ell)} z_k^{(\ell)}\right] \qquad (3.38)
$$

after substituting in the decomposition (3.37). With the given pattern of neural indices, there are really only three types of terms in the sum. In particular, there is one term that looks like this

$$
\frac{1}{n_\ell^3} \sum_{i,j,k=1}^{n_\ell} \delta_{ii}\delta_{jj}\delta_{kk} = 1 , \qquad (3.39)
$$

six terms that look like this

$$
\frac{1}{n_\ell^3} \sum_{i,j,k=1}^{n_\ell} \delta_{ij}\delta_{ji}\delta_{kk} = \frac{1}{n_\ell} , \qquad (3.40)
$$

and eight terms that look like this

$$\frac{1}{n_\ell^3}\sum_{i,j,k=1}^{n_\ell}\delta_{ij}\delta_{jk}\delta_{ki}=\frac{1}{n_\ell^2}\,. \tag{3.41}$$

Putting all these terms together, we find a recursion for the layer-dependence of the full six-point correlator

$$G_6^{(\ell+1)}=C_W^3\left(1+\frac{6}{n_\ell}+\frac{8}{n_\ell^2}\right)G_6^{(\ell)}\,, \tag{3.42}$$

which has a simple solution

$$\begin{aligned}G_6^{(\ell)}=&C_W^{3\ell}\left[\prod_{\ell'=1}^{\ell-1}\left(1+\frac{6}{n_{\ell'}}+\frac{8}{n_{\ell'}^2}\right)\right]\left(G_2^{(0)}\right)^3\\ =&\left[\prod_{\ell'=1}^{\ell-1}\left(1+\frac{6}{n_{\ell'}}+\frac{8}{n_{\ell'}^2}\right)\right]\left(G_2^{(\ell)}\right)^3\,.\end{aligned} \tag{3.43}$$

Here, we used the initial condition (3.34) $G_6^{(0)}=\left(G_2^{(0)}\right)^3$, and on the final line we substituted in our solution for the variance of a single input (3.15).

Similarly, we can decompose an arbitrary $2m$-point correlator as

$$\mathbb{E}\left[z_{i_1}^{(\ell)}z_{i_2}^{(\ell)}\cdots z_{i_{2m-1}}^{(\ell)}z_{i_{2m}}^{(\ell)}\right]=\left(\sum_{\text{all parings}}\delta_{i_{k_1}i_{k_2}}\cdots\delta_{i_{k_{2m-1}}i_{k_{2m}}}\right)G_{2m}^{(\ell)}\,, \tag{3.44}$$

and use a similar set of manipulations to show that the layer dependence $G_{2m}^{(\ell)}$ obeys a recursion

$$G_{2m}^{(\ell+1)}=c_{2m}(n_\ell)\,C_W^m G_{2m}^{(\ell)}\,, \tag{3.45}$$

with the combinatorial factor $c_{2m}(n)$ given by

$$c_{2m}(n)=\left(1+\frac{2}{n}\right)\left(1+\frac{4}{n}\right)\cdots\left(1+\frac{2m-2}{n}\right)=\frac{(\frac{n}{2}-1+m)!}{(\frac{n}{2}-1)!}\left(\frac{2}{n}\right)^m\,. \tag{3.46}$$

We included the explicit form of this factor only for completeness. If you insist on checking this factor, note that it reproduces the right combinatorial factors for $2m=2,4,6$, though we strongly suggest that you do not explicitly write out all of the terms for any other particular value of m. Overall, this recursion is still just a simple sequence of multiplications, with a simple solution

$$G_{2m}^{(\ell)}=\left[\prod_{\ell'=1}^{\ell-1}c_{2m}(n'_\ell)\right]\left(G_2^{(\ell)}\right)^m\,. \tag{3.47}$$

Enough with the math, time for the physics.

Physics: breakdown of perturbation theory and the emergence of chaos

Let's play with this formula (3.47) a bit by taking various limits. For simplicity, let's set all the hidden layer widths to be equal $n_1 = n_2 = \ldots = n_{L-1} \equiv n$, and also focus only on output distribution $p\Big(z^{(L)}\Big|x\Big)$.

- On the one hand, if we send the network width to infinity, $n \to \infty$, while keeping the depth L fixed, then all the combinatorial factors (3.46) become unity:

$$\lim_{n\to\infty} c_{2m}(n) = 1\,. \tag{3.48}$$

  In this infinite-width limit, all the correlators (3.47) are given by their Gaussian values

$$G^{(L)}_{2m} = \Big(G^{(L)}_2\Big)^m\,, \tag{3.49}$$

  and the output distribution $p\Big(z^{(L)}\Big|x\Big)$ is precisely Gaussian. More generally, even for multiple inputs the output distribution $p\Big(z^{(L)}\Big|\mathcal{D}\Big)$ remains Gaussian, with covariance $G^{(L)}_{\alpha_1\alpha_2} = C_W^L\left(\frac{1}{n_0}\sum_{i=1}^{n_0} x_{i;\alpha_1}x_{i;\alpha_2}\right)$. As this distribution is equivalent to that of one-layer networks initialized with weight variance C_W^L, we see that such networks are not really deep after all.

- On the other hand, if we send the depth to infinity, $L \to \infty$, while keeping the width n fixed, then all the combinatorial factors are fixed and greater than one, $c_{2m} > 1$. This means that the higher-point correlators for $2m > 2$ will all blow up exponentially as

$$G^{(L)}_{2m} = \Big[c_{2m}(n)\Big]^{L-1}\Big(G^{(L)}_2\Big)^m\,. \tag{3.50}$$

  Note that this behavior persists *even if* we tune the network to criticality by setting $C_W = 1$ so that the two-point correlator is fixed $G^{(\ell)}_2 = G^{(0)}_2$. This shows explicitly how our large-width analysis from the last section can break down if the network depth becomes too large. Furthermore, the distribution implied by these correlators is extremely non-Gaussian, to say the least, and in practice the outputs of these networks will fluctuate chaotically from instantiation to instantiation. Such networks are entirely unusable.

- Clearly these limits do not commute, i.e.,

$$\lim_{n\to\infty}\lim_{L\to\infty} G^{(L)}_{2m} \neq \lim_{L\to\infty}\lim_{n\to\infty} G^{(L)}_{2m}\,. \tag{3.51}$$

  However, we can construct an *interpolating solution* by sending both the width and depth to infinity, $n, L \to \infty$, while keeping their ratio fixed:

$$r \equiv \frac{L}{n}\,. \tag{3.52}$$

Noting that we can expand the combinatorial factors as

$$c_{2m}(n) = 1 + \frac{1}{n}\left(\sum_{s=1}^{m-1} 2s\right) + O\Big(\frac{1}{n^2}\Big) = 1 + \frac{m(m-1)}{n} + O\Big(\frac{1}{n^2}\Big)\,, \tag{3.53}$$

and then using the well-known formula for the exponential

$$\lim_{L\to\infty}\left[1 + \frac{a}{L} + O\Big(\frac{1}{L^2}\Big)\right]^L = e^a\,, \tag{3.54}$$

we can construct a limiting value for any correlator at a given value of m and fixed aspect ratio r:

$$G_{2m}^{(L)} \to e^{m(m-1)r}\left(G_2^{(L)}\right)^m\,. \tag{3.55}$$

This solution interpolates between the two extreme limits: by sending $r \to 0$ we recover the Gaussian limit (3.49), and by sending $r \to \infty$ we recover the chaotic limit (3.50) that demonstrates the breakdown of criticality.[7]

Let us play a tiny bit more with the last interpolating formula (3.55) at criticality where $G_2^{(L)} = G_2^{(0)}$. Here, the finite-width difference (3.28) that governs the connected four-point correlator (3.29) becomes

$$\begin{aligned} G_4^{(L)} - \left(G_2^{(L)}\right)^2 &= \left(e^{2r} - 1\right)\left(G_2^{(0)}\right)^2 \\ &= 2r\left(G_2^{(0)}\right)^2 + O\Big(r^2\Big)\,. \end{aligned} \tag{3.56}$$

This reproduces the *running* of the quartic coupling with the depth-to-width ratio (3.28). Similarly, the corresponding quantity governing the layer dependence of the connected six-point correlator (1.61) is given by

$$\begin{aligned} G_6^{(L)} - 3G_2^{(L)}G_4^{(L)} + 2\left(G_2^{(L)}\right)^3 &= \left(e^{6r} - 3e^{2r} + 2\right)\left(G_2^{(0)}\right)^3 \\ &= 12r^2\left(G_2^{(0)}\right)^3 + O\Big(r^3\Big)\,, \end{aligned} \tag{3.57}$$

which scales like the depth-to-width ratio *squared*. Therefore, the connected six-point correlator is even more suppressed than the connected four-point correlator for large networks with sufficiently small depth-to-width ratio r. This is in accord with the comments we made in §1.3: neural networks obey *nearly-Gaussian* statistics, and the

[7] This *double-scaling limit* corresponds to neglecting terms that scale like $\frac{L}{n^2}$, $\frac{L^3}{n^5}$, $\frac{L^{120}}{n^{157}}$, etc., which are all subleading when the depth and the width are large, $n, L \to \infty$, but their ratio r is fixed.

Furthermore, there is a very subtle point in using this interpolating solution – albeit a theoretical subtlety – when we consider not just a particular correlator at a given $2m$, but the set of *all* the correlators. Namely, for any *finite* n, L – no matter how big – there always exist higher-point correlators for which the exponential approximation (3.55) is invalid because the factor of $m(m-1)$ becomes too big. That is, since we constructed this interpolating solution assuming fixed m, such a solution can break down if m is large enough.

connected correlators have a hierarchical structure. In particular, we see here that the scaling of the correlators is controlled by the same small parameter r, with the higher-point connected correlators suppressed by a higher power of that parameter. This means that for small r, we should be able to consistently truncate our distribution and only compute up to a fixed order in r.

Chapter 4

RG Flow of Preactivations

"You can hide a lot in a large-N matrix." – Steve Shenker – John McGreevy [26].

At the end of the last chapter, we computed the statistics of preactivations for deep linear networks at initialization and saw them *run* as a function of the network depth. For that toy model, using a handful of Wick contractions and the recursive structure of the network architecture, we were able to fully understand the effects of the network's hyperparameters – its initialization scheme, width, and depth – on preactivation correlators. This exercise in particular highlighted the importance of *critical initialization hyperparameters* and sufficiently small *depth-to-width ratio* in order for the network outputs to be well-behaved, theoretically and practically. To extend these insights beyond deep linear networks, we need to develop an effective theory of deep learning for networks with any activation function.

While ultimately the goal of our effective theory is to explain how a *particular* neural network learns from a given dataset, our immediate goal in §4 and §5 will be to understand how an *ensemble* of neural networks at initialization behaves as a function of data. In §10, §11, and §∞, we'll find that these goals are closely tied together: through the judicious study of the ensemble, we can systematically evaluate the *typical* behavior of trained networks as well as how any particular network may *fluctuate* away from typicality. Our starting point will thus be a study of the statistics of neural-network preactivations with Gaussian-initialized biases and weights. All in all, the formalism developed in this chapter for analyzing the ensemble of networks at initialization will be the key to a principled understanding of deep learning.

As stressed in the introduction, §0, our focus will always be on describing real finite-width networks, since a lot is lost in idealized infinite-width networks. One salient phenomenon lost in the infinite-width limit is the increasing non-Gaussianity in the preactivation distributions of deeper layers. Such non-Gaussianity makes the behavior of finite-width networks much richer but more complicated to analyze. In order to tame these complications, we'll need to borrow some tools from theoretical physics. In particu-

lar, physicists have a long tradition of finding simple descriptions of complicated systems in the limit of a large number of degrees of freedom, while keeping in mind the true goal of modeling real systems. In our context, this hints at tractability and simplification in the regime where networks become very wide, though not infinitely so. To make this precise, in this chapter we introduce the *large-n* expansion or $1/n$ expansion in order to perform perturbative expansions when hidden-layer width n becomes parametrically big. With this tool, we'll be able to systematically study the preactivation distributions of finite neural networks to arbitrary precision.[1]

As we did for deep linear networks, we will proceed recursively, investigating how the distribution of preactivations changes from layer to layer by following the transformation of inputs via the iterative MLP forward-pass equation. We start in §4.1 by computing the distribution of preactivations in the first layer, integrating out the first set of weights and biases. This procedure recovers a well-known result that the distribution of the first-layer preactivations is Gaussian. Since this calculation is so central to the rest of the chapter, we'll present two different derivations: a combinatorial derivation in terms of Wick contractions and an algebraic derivation using the Hubbard-Stratonovich transformation.

Next, in §4.2, we'll consider the distribution of preactivations in the second layer and see the emergence of non-Gaussianity in four-point and higher-point connected correlators. The magnitude of these correlators is suppressed when the network is very wide, vanishing in the strict infinite-width limit. This suppression for wide networks in turn enables us to write down an action describing the preactivation distribution, building on the correspondence explored in §1 between such connected correlators and the couplings in the action. In particular, the large-n expansion lets us start with the quadratic action describing the Gaussian distribution in the infinite-with limit and then perturbatively expand around it in a series of the inverse width, $1/n$, to arbitrary desired precision. Given the importance of this result, we again provide two derivations, one based on Wick contractions and the other based on expanding the stochastic metric.

Finally, in §4.3, we'll analyze the distribution of preactivations at any depth. At this point we can simply repurpose the calculations from the preceding sections to see how the distribution of preactivations recursively transforms from the ℓ-th layer to the $(\ell+1)$-th layer. In particular, keeping the leading finite-width $1/n$ corrections, we'll obtain recursion equations for the two-point and four-point correlators, encoding how these observables evolve with increasing depth. We'll see that the preactivation distribution

[1] Back in 1996, Neal introduced the *infinite-width limit* in a seminal work [27], focusing on single-hidden-layer networks. Much later, this program was continued in [28], extending the infinite-width limit to deeper networks, and then was extended further by Yaida in [29] to *finite-width networks*. A large part of this chapter is focused on reproducing the recursions first derived in [29].

However, our perspective here is different than the one taken in this prior work. In particular, our main motivation is in computing the distribution of preactivations at initialization, with an eye towards ultimately understanding gradient-based training (§10, §11, §∞), rather than providing a starting point for Bayesian inference. (We will give our own perspective on Bayesian learning for deep learning in §6.) Additionally, in contrast to [29], our results here are derived by first focusing on the couplings in the *action*, rather than directly on the correlators of the distribution. This method is more intuitive and can be more easily extended.

of the $(\ell+1)$-th layer contains a non-Gaussian piece inherited from the ℓ-th layer as well as an additional non-Gaussianity generated in the transition from the ℓ-th to $(\ell+1)$-th layer. In the next chapter, §5, we'll see in detail how the non-Gaussianity accumulates with depth by explicitly solving these recursions and analyzing their solutions, which extends the notion of criticality and emergence of the depth-to-width ratio to networks with general activation functions.

After a short clarifying section on some implications of marginalization (§4.4) and a section on subleading corrections (§4.5), we take a step back in §4.6 in order to draw a parallel between our formalism and the *renormalization group* in theoretical physics. Renormalization group is a powerful recursive method for understanding complicated interacting systems, capturing how the effective interactions between the constituents of a system change when the scale at which they are measured changes from microscopic to macroscopic. Specifically, renormalization marginalizes over the microscopic degrees of freedom in the system to yield an effective *coarse-grained* description at long distances. This is analogous to the way we recursively marginalize over preactivations in previous layers to obtain an effective description of a *representation* at the current layer, in our case capturing how the interactions between neurons change with depth. In both cases the flow of the distributions is created by the marginalization of fine-grained information. Given the complete parallel, we will call our flow *representation group (RG) flow*.

If this sounds like a popular heuristic explanation for what deep neural networks do – transforming fine-grained information at the input level into coarser information at the feature levels and finally into fully coarse-grained representation at the output level – that's because our formalism makes this heuristic picture of representation coarse-graining concrete.[2] Our formalism will further let us directly probe the effect of the *deep* in *deep learning* by tracking the change in preactivation distributions as we increase the number of layers. Thus, it is the starting point for an effective theory of deep learning, which we will continue to develop throughout the book.

4.1 First Layer: Good-Old Gaussian

Given a dataset

$$\mathcal{D} = \{x_{i;\alpha}\}_{i=1,\ldots,n_0;\,\alpha=1,\ldots,N_{\mathcal{D}}} \tag{4.1}$$

containing $N_{\mathcal{D}}$ inputs of n_0-dimensional vectors, the preactivations in the first layer are given by

$$z^{(1)}_{i;\alpha} \equiv z^{(1)}_i(x_\alpha) = b^{(1)}_i + \sum_{j=1}^{n_0} W^{(1)}_{ij} x_{j;\alpha}\,, \quad \text{for} \quad i = 1, \ldots, n_1\,. \tag{4.2}$$

[2]There have been many formal and informal comments on the connection between renormalization and deep learning, but the relationship has never before been made precise.

At initialization the biases $b^{(1)}$ and weights $W^{(1)}$ are independently distributed according to mean-zero Gaussian distributions with variances

$$\mathbb{E}\left[b_i^{(1)} b_j^{(1)}\right] = \delta_{ij} C_b^{(1)} \,, \tag{4.3}$$

$$\mathbb{E}\left[W_{i_1 j_1}^{(1)} W_{i_2 j_2}^{(1)}\right] = \delta_{i_1 i_2} \delta_{j_1 j_2} \frac{C_W^{(1)}}{n_0} \,. \tag{4.4}$$

The first-layer preactivations $z^{(1)} = z_{i;\alpha}^{(1)}$ form an $(n_1 N_{\mathcal{D}})$-dimensional vector, and we are interested in its distribution at initialization,

$$p\Big(z^{(1)}\Big|\mathcal{D}\Big) = p\Big(z^{(1)}\left(x_1\right), \ldots, z^{(1)}\left(x_{N_{\mathcal{D}}}\right)\Big) \,. \tag{4.5}$$

Note how this distribution depends conditionally on the input data, representing the fact that the preactivations are functions of the input.

Now, let us compute the distribution of the first-layer preactivations at initialization. Since this will be so important, we give two derivations, one combinatorial and one algebraic.

Wick this way: combinatorial derivation via correlators

The first derivation involves direct application of Wick contractions to compute correlators of the first-layer distribution (4.5). Starting with the one-point correlator, simply inserting the definition of the first-layer preactivations (4.2) gives

$$\mathbb{E}\left[z_{i;\alpha}^{(1)}\right] = \mathbb{E}\left[b_i^{(1)} + \sum_{j=1}^{n_0} W_{ij}^{(1)} x_{j;\alpha_1}\right] = 0 \,, \tag{4.6}$$

since $\mathbb{E}\left[b_i^{(1)}\right] = \mathbb{E}\left[W_{ij}^{(1)}\right] = 0$. In fact, it's easy to see that all the odd-point correlators of $p\left(z^{(1)}\middle|\mathcal{D}\right)$ vanish because there always is an odd number of either biases $b^{(1)}$ or weights $W^{(1)}$ left unpaired under Wick contractions.

Next for the two-point correlator, again inserting the definition (4.2), we see

$$\begin{aligned}\mathbb{E}\left[z_{i_1;\alpha_1}^{(1)} z_{i_2;\alpha_2}^{(1)}\right] &= \mathbb{E}\left[\left(b_{i_1}^{(1)} + \sum_{j_1=1}^{n_0} W_{i_1 j_1}^{(1)} x_{j_1;\alpha_1}\right)\left(b_{i_2}^{(1)} + \sum_{j_2=1}^{n_0} W_{i_2 j_2}^{(1)} x_{j_2;\alpha_2}\right)\right] \\ &= \delta_{i_1 i_2}\left(C_b^{(1)} + C_W^{(1)} \frac{1}{n_0} \sum_{j=1}^{n_0} x_{j;\alpha_1} x_{j;\alpha_2}\right) = \delta_{i_1 i_2} G_{\alpha_1 \alpha_2}^{(1)} \,,\end{aligned} \tag{4.7}$$

where to get to the second line we Wick-contracted the biases and weights using (4.3) and (4.4). We also introduced the first-layer **metric**

$$G_{\alpha_1 \alpha_2}^{(1)} \equiv C_b^{(1)} + C_W^{(1)} \frac{1}{n_0} \sum_{j=1}^{n_0} x_{j;\alpha_1} x_{j;\alpha_2} \,, \tag{4.8}$$

which is a function of the two samples, $G^{(1)}_{\alpha_1\alpha_2} = G^{(1)}(x_{\alpha_1}, x_{\alpha_2})$, and represents the two-point correlation of preactivations in the first layer between different samples.

The higher-point correlators can be obtained similarly. For instance, the full four-point correlation can be obtained by inserting the definition (4.2) four times and Wick-contracting the biases and weights, yielding

$$\begin{aligned}
&\mathbb{E}\left[z^{(1)}_{i_1;\alpha_1} z^{(1)}_{i_2;\alpha_2} z^{(1)}_{i_3;\alpha_3} z^{(1)}_{i_4;\alpha_4}\right] \\
=&\delta_{i_1 i_2}\delta_{i_3 i_4} G^{(1)}_{\alpha_1\alpha_2} G^{(1)}_{\alpha_3\alpha_4} + \delta_{i_1 i_3}\delta_{i_2 i_4} G^{(1)}_{\alpha_1\alpha_3} G^{(1)}_{\alpha_2\alpha_4} + \delta_{i_1 i_4}\delta_{i_2 i_3} G^{(1)}_{\alpha_1\alpha_4} G^{(1)}_{\alpha_2\alpha_3} \\
=&\mathbb{E}\left[z^{(1)}_{i_1;\alpha_1} z^{(1)}_{i_2;\alpha_2}\right]\mathbb{E}\left[z^{(1)}_{i_3;\alpha_3} z^{(1)}_{i_4;\alpha_4}\right] + \mathbb{E}\left[z^{(1)}_{i_1;\alpha_1} z^{(1)}_{i_3;\alpha_3}\right]\mathbb{E}\left[z^{(1)}_{i_2;\alpha_2} z^{(1)}_{i_4;\alpha_4}\right] \\
&+ \mathbb{E}\left[z^{(1)}_{i_1;\alpha_1} z^{(1)}_{i_4;\alpha_4}\right]\mathbb{E}\left[z^{(1)}_{i_2;\alpha_2} z^{(1)}_{i_3;\alpha_3}\right] .
\end{aligned} \tag{4.9}$$

Note that the end result is same as Wick-contracting $z^{(1)}$'s with the variance given by (4.7). As we recall from §1, this can compactly be summarized by saying that the *connected* four-point correlator vanishes,

$$\mathbb{E}\left[z^{(1)}_{i_1;\alpha_1} z^{(1)}_{i_2;\alpha_2} z^{(1)}_{i_3;\alpha_3} z^{(1)}_{i_4;\alpha_4}\right]\Big|_{\text{connected}} = 0\,. \tag{4.10}$$

Similar Wick combinatorics shows that all the full higher-point correlators can be obtained simply by Wick-contracting $z^{(1)}$'s with the variance given by (4.7), and hence all the connected higher-point correlators vanish. This means that all correlators can be generated from a Gaussian distribution with zero mean and the variance (4.7).

Then, in order to write down the first-layer action, all we need is the inverse of this variance, given by a matrix $\delta_{i_1 i_2} G^{\alpha_1\alpha_2}_{(1)}$ that satisfies

$$\sum_{j=1}^{n_1}\sum_{\beta\in\mathcal{D}}\left(\delta_{i_1 j} G^{\alpha_1\beta}_{(1)}\right)\left(\delta_{j i_2} G^{(1)}_{\beta\alpha_2}\right) = \delta_{i_1 i_2}\delta^{\alpha_1}_{\ \alpha_2}\,, \tag{4.11}$$

with the inverse of the first-layer metric $G^{(1)}_{\alpha_1\alpha_2}$ denoted as $G^{\alpha_1\alpha_2}_{(1)}$ and defined by

$$\sum_{\beta\in\mathcal{D}} G^{\alpha_1\beta}_{(1)} G^{(1)}_{\beta\alpha_2} = \delta^{\alpha_1}_{\ \alpha_2}\,. \tag{4.12}$$

Just as in §1, we follow the conventions of *general relativity* and suppress the superscript "-1" for the inverse metric, distinguishing the metric $G^{(1)}_{\alpha_1\alpha_2}$ and the inverse metric $G^{\alpha_1\alpha_2}_{(1)}$ by whether sample indices are lowered or raised. With this notation, the Gaussian distribution for the first-layer preactivations is expressed as

$$p\Big(z^{(1)}\Big|\mathcal{D}\Big) = \frac{1}{Z} e^{-S\left(z^{(1)}\right)}\,, \tag{4.13}$$

with the quadratic action

$$S\Big(z^{(1)}\Big) = \frac{1}{2}\sum_{i=1}^{n_1}\sum_{\alpha_1,\alpha_2\in\mathcal{D}} G^{\alpha_1\alpha_2}_{(1)} z^{(1)}_{i;\alpha_1} z^{(1)}_{i;\alpha_2}\,, \tag{4.14}$$

and the partition function

$$Z = \int \left[\prod_{i,\alpha} dz^{(1)}_{i;\alpha} \right] e^{-S\left(z^{(1)}\right)} = \left| 2\pi G^{(1)} \right|^{\frac{n_1}{2}} , \tag{4.15}$$

where $\left|2\pi G^{(1)}\right|$ is the determinant of the $N_{\mathcal{D}}$-by-$N_{\mathcal{D}}$ matrix $2\pi G^{(1)}_{\alpha_1\alpha_2}$ and, whenever we write out a determinant involving the metric, it will always be that of the metric and *not* of the inverse metric.

Hubbard-Stratonovich this way: algebraic derivation via action

Rather than first computing correlators and then backing out the distribution that generates them, we can instead work with the distribution directly. Let's start with the formal expression for the preactivation distribution (2.33) worked out in the last chapter[3]

$$p(z|\mathcal{D}) = \int \left[\prod_i db_i \; p(b_i) \right] \left[\prod_{i,j} dW_{ij} \; p(W_{ij}) \right] \prod_{i,\alpha} \delta \left(z_{i;\alpha} - b_i - \sum_j W_{ij} x_{j;\alpha} \right) , \tag{4.16}$$

where we have momentarily suppressed the layer superscripts "(1)" because it is distracting. At this point, we could try to eliminate some of the integrals over the model parameters against the constraints imposed by the Dirac delta functions, but it's easy to get confused by the different numbers of model-parameter integrals and delta-function constraints.

To clarify matters, we import a neat trick from theoretical physics called the **Hubbard-Stratonovich transformation**. Specifically, using the following integral representation of the Dirac delta function (2.32)

$$\delta(z-a) = \int \frac{d\Lambda}{2\pi} e^{i\Lambda(z-a)} \tag{4.17}$$

for each constraint and also plugging in explicit expressions for the Gaussian distributions over the parameters, we obtain

$$\begin{aligned} p(z|\mathcal{D}) = \int & \left[\prod_i \frac{db_i}{\sqrt{2\pi C_b}} \right] \left[\prod_{i,j} \frac{dW_{ij}}{\sqrt{2\pi C_W / n_0}} \right] \left[\prod_{i,\alpha} \frac{d\Lambda_i{}^{\alpha}}{2\pi} \right] \\ & \times \exp \left[- \sum_i \frac{b_i^2}{2C_b} - n_0 \sum_{i,j} \frac{W_{ij}^2}{2C_W} + i \sum_{i,\alpha} \Lambda_i{}^{\alpha} \left(z_{i;\alpha} - b_i - \sum_j W_{ij} x_{j;\alpha} \right) \right] . \end{aligned} \tag{4.18}$$

[3]For architectures other than MLPs, the expression inside the Dirac delta function would be different, but we expect much of the following to hold so long as the parameters are sampled from simple distributions.

Completing the square in the exponential for both the biases b and weights W, we see that the action is quadratic in the model parameters

$$\begin{aligned} &-\sum_i \frac{b_i^2}{2C_b} - n_0 \sum_{i,j} \frac{W_{ij}^2}{2C_W} + i \sum_{i,\alpha} \Lambda_i{}^{\alpha} \left(z_{i;\alpha} - b_i - \sum_j W_{ij} x_{j;\alpha} \right) \\ =& -\frac{1}{2C_b} \sum_i \left(b_i + iC_b \sum_\alpha \Lambda_i{}^{\alpha} \right)^2 - \frac{C_b}{2} \sum_i \left(\sum_\alpha \Lambda_i{}^{\alpha} \right)^2 \\ &- \frac{n_0}{2C_W} \sum_{i,j} \left(W_{ij} + i \frac{C_W}{n_0} \sum_\alpha \Lambda_i{}^{\alpha} x_{j;\alpha} \right)^2 - \frac{C_W}{2n_0} \sum_{i,j} \left(\sum_\alpha \Lambda_i{}^{\alpha} x_{j;\alpha} \right)^2 + i \sum_{i,\alpha} \Lambda_i{}^{\alpha} z_{i;\alpha} \,. \end{aligned} \tag{4.19}$$

The biases and weights can then be integrated out, yielding an integral representation for the first-layer distribution $p(z)$ as

$$\int \left[\prod_{i,\alpha} \frac{d\Lambda_i{}^{\alpha}}{2\pi} \right] \exp \left[-\frac{1}{2} \sum_{i,\alpha_1,\alpha_2} \Lambda_i{}^{\alpha_1} \Lambda_i{}^{\alpha_2} \left(C_b + C_W \sum_j \frac{x_{j;\alpha_1} x_{j;\alpha_2}}{n_0} \right) + i \sum_{i,\alpha} \Lambda_i{}^{\alpha} z_{i;\alpha} \right] . \tag{4.20}$$

In essence, we've so far traded the delta-function constraints and the model parameters for the auxiliary Hubbard-Stratonovich variables $\Lambda_i{}^{\alpha}$, which have quadratic action and a simple linear interaction with the preactivations $z_{i;\alpha}$.

Note that the inverse variance for the Hubbard-Stratonovich variables $\Lambda_i{}^{\alpha}$ is just the first-layer metric (4.8) we introduced in the Wick-contraction derivation,

$$C_b^{(1)} + C_W^{(1)} \sum_j \frac{x_{j;\alpha_1} x_{j;\alpha_2}}{n_0} = G^{(1)}_{\alpha_1 \alpha_2} \,, \tag{4.21}$$

where by now enough dust has settled that layer superscripts "(1)" have been restored. Once again completing the square, the argument of the exponential becomes

$$-\frac{1}{2} \sum_{i,\alpha_1,\alpha_2} \left[G^{(1)}_{\alpha_1 \alpha_2} \left(\Lambda_i{}^{\alpha_1} - i \sum_{\beta_1} G_{(1)}^{\alpha_1 \beta_1} z^{(1)}_{i;\beta_1} \right) \left(\Lambda_i{}^{\alpha_2} - i \sum_{\beta_2} G_{(1)}^{\alpha_2 \beta_2} z^{(1)}_{i;\beta_2} \right) + G_{(1)}^{\alpha_1 \alpha_2} z^{(1)}_{i;\alpha_1} z^{(1)}_{i;\alpha_2} \right] , \tag{4.22}$$

which finally lets us integrate out the Hubbard-Stratonovich variables $\Lambda_i{}^{\alpha}$ and recover our previous result

$$p\Big(z^{(1)}\Big|\mathcal{D}\Big) = \frac{1}{\left|2\pi G^{(1)}\right|^{\frac{n_1}{2}}} \exp\left(-\frac{1}{2} \sum_{i=1}^{n_1} \sum_{\alpha_1,\alpha_2 \in \mathcal{D}} G_{(1)}^{\alpha_1 \alpha_2} z^{(1)}_{i;\alpha_1} z^{(1)}_{i;\alpha_2} \right) . \tag{4.23}$$

As before, $\left|2\pi G^{(1)}\right|$ represents the determinant of the matrix $2\pi G^{(1)}_{\alpha_1 \alpha_2}$. The first-layer distribution is Gaussian with each neuron independent, and correlations between preactivations for different samples are encoded entirely in the metric $G^{(1)}_{\alpha_1 \alpha_2}$.

Gaussian action in action

Now that we've obtained an action representation for the distribution of the first-layer preactivations in two different ways, let's get a feel for how to compute with it. We'll start by computing the expectation of some quantities that will be needed in §4.2: the expectation of two activations on the same neuron, $\mathbb{E}\left[\sigma\left(z^{(1)}_{i_1;\alpha_1}\right)\sigma\left(z^{(1)}_{i_1;\alpha_2}\right)\right]$, and the expectation of four activations, $\mathbb{E}\left[\sigma\left(z^{(1)}_{i_1;\alpha_1}\right)\sigma\left(z^{(1)}_{i_1;\alpha_2}\right)\sigma\left(z^{(1)}_{i_2;\alpha_3}\right)\sigma\left(z^{(1)}_{i_2;\alpha_4}\right)\right]$, either with all four on the same neuron $i_1 = i_2$ or with each pair on two separate neurons $i_1 \neq i_2$.

Let's start with the two-point correlator of activations. Using the definition of the expectation and inserting the action representation of the distribution (4.23), we get

$$\begin{aligned}
&\mathbb{E}\left[\sigma\left(z^{(1)}_{i_1;\alpha_1}\right)\sigma\left(z^{(1)}_{i_1;\alpha_2}\right)\right] \qquad (4.24)\\
&= \int \left[\prod_{i=1}^{n_1} \frac{\prod_{\alpha\in\mathcal{D}} dz_{i;\alpha}}{\sqrt{|2\pi G^{(1)}|}}\right] \exp\left(-\frac{1}{2}\sum_{j=1}^{n_1}\sum_{\beta_1,\beta_2\in\mathcal{D}} G^{\beta_1\beta_2}_{(1)} z_{j;\beta_1} z_{j;\beta_2}\right) \sigma(z_{i_1;\alpha_1})\,\sigma(z_{i_1;\alpha_2})\\
&= \left\{\prod_{i\neq i_1} \int \left[\frac{\prod_{\alpha\in\mathcal{D}} dz_{i;\alpha}}{\sqrt{|2\pi G^{(1)}|}}\right] \exp\left(-\frac{1}{2}\sum_{\beta_1,\beta_2\in\mathcal{D}} G^{\beta_1\beta_2}_{(1)} z_{i;\beta_1} z_{i;\beta_2}\right)\right\}\\
&\quad\times \int \left[\frac{\prod_{\alpha\in\mathcal{D}} dz_{i_1;\alpha}}{\sqrt{|2\pi G^{(1)}|}}\right] \exp\left(-\frac{1}{2}\sum_{\beta_1,\beta_2\in\mathcal{D}} G^{\beta_1\beta_2}_{(1)} z_{i_1;\beta_1} z_{i_1;\beta_2}\right) \sigma(z_{i_1;\alpha_1})\,\sigma(z_{i_1;\alpha_2})\\
&= \{1\} \times \left[\int \frac{\prod_{\alpha\in\mathcal{D}} dz_{\alpha}}{\sqrt{|2\pi G^{(1)}|}}\right] \exp\left(-\frac{1}{2}\sum_{\beta_1,\beta_2\in\mathcal{D}} G^{\beta_1\beta_2}_{(1)} z_{\beta_1} z_{\beta_2}\right) \sigma(z_{\alpha_1})\,\sigma(z_{\alpha_2})\\
&\equiv \left\langle \sigma(z_{\alpha_1})\,\sigma(z_{\alpha_2})\right\rangle_{G^{(1)}} .
\end{aligned}$$

The second equality states that the probability distribution factorizes for each neuron due to the relation $e^{x+y} = e^x e^y$. To go from the second equality to the third, we compute the integrals for the neurons with $i \neq i_1$, which are all trivial, and we also rename the dummy integral variable $z_{i_1;\alpha}$ to z_α. The final equality reintroduces the notation (1.68)

$$\left\langle F\left(z_{\alpha_1},\ldots,z_{\alpha_m}\right)\right\rangle_g \equiv \int \left[\frac{\prod_{\alpha\in\mathcal{D}} dz_\alpha}{\sqrt{|2\pi g|}}\right] \exp\left(-\frac{1}{2}\sum_{\beta_1,\beta_2\in\mathcal{D}} g^{\beta_1\beta_2} z_{\beta_1} z_{\beta_2}\right) F(z_{\alpha_1},\ldots,z_{\alpha_m}) \qquad (4.25)$$

to describe a Gaussian expectation with variance g and an arbitrary function $F(z_{\alpha_1},\ldots,z_{\alpha_m})$ over variables with sample indices *only*. In other parts of this book we'll explicitly evaluate this type of Gaussian expectation in various setups for concrete choices of activation functions, but for the purpose of this chapter we will view computations as complete when they are reduced to such Gaussian expectations without any neural indices. Introducing further the simplifying notation

$$\sigma_\alpha \equiv \sigma(z_\alpha) \ , \qquad (4.26)$$

the result of the computation above can be succinctly summarized as

$$\mathbb{E}\left[\sigma\left(z_{i_1;\alpha_1}^{(1)}\right)\sigma\left(z_{i_1;\alpha_2}^{(1)}\right)\right] = \langle \sigma_{\alpha_1}\sigma_{\alpha_2}\rangle_{G^{(1)}} \ . \tag{4.27}$$

It's easy to generalize this to correlators of more than two activations. For instance, for four activations on the same neuron $i_1 = i_2$, we have by the exact same manipulations

$$\mathbb{E}\left[\sigma\left(z_{i_1;\alpha_1}^{(1)}\right)\sigma\left(z_{i_1;\alpha_2}^{(1)}\right)\sigma\left(z_{i_1;\alpha_3}^{(1)}\right)\sigma\left(z_{i_1;\alpha_4}^{(1)}\right)\right] = \langle \sigma_{\alpha_1}\sigma_{\alpha_2}\sigma_{\alpha_3}\sigma_{\alpha_4}\rangle_{G^{(1)}} \ , \tag{4.28}$$

and for each pair on two different neurons $i_1 \neq i_2$, we have

$$\begin{aligned}
&\mathbb{E}\left[\sigma\left(z_{i_1;\alpha_1}^{(1)}\right)\sigma\left(z_{i_1;\alpha_2}^{(1)}\right)\sigma\left(z_{i_2;\alpha_3}^{(1)}\right)\sigma\left(z_{i_2;\alpha_4}^{(1)}\right)\right] \qquad (4.29)\\
&= \left\{\prod_{i\notin\{i_1,i_2\}}\int\left[\frac{\prod_{\alpha\in\mathcal{D}}dz_{i;\alpha}}{\sqrt{|2\pi G^{(1)}|}}\right]\exp\left(-\frac{1}{2}\sum_{\beta_1,\beta_2\in\mathcal{D}}G_{(1)}^{\beta_1\beta_2}z_{i;\beta_1}z_{i;\beta_2}\right)\right\}\\
&\quad\times\int\left[\frac{\prod_{\alpha\in\mathcal{D}}dz_{i_1;\alpha}}{\sqrt{|2\pi G^{(1)}|}}\right]\exp\left(-\frac{1}{2}\sum_{\beta_1,\beta_2\in\mathcal{D}}G_{(1)}^{\beta_1\beta_2}z_{i_1;\beta_1}z_{i_1;\beta_2}\right)\sigma(z_{i_1;\alpha_1})\,\sigma(z_{i_1;\alpha_2})\\
&\quad\times\int\left[\frac{\prod_{\alpha\in\mathcal{D}}dz_{i_2;\alpha}}{\sqrt{|2\pi G^{(1)}|}}\right]\exp\left(-\frac{1}{2}\sum_{\beta_1,\beta_2\in\mathcal{D}}G_{(1)}^{\beta_1\beta_2}z_{i_2;\beta_1}z_{i_2;\beta_2}\right)\sigma(z_{i_2;\alpha_3})\,\sigma(z_{i_2;\alpha_4})\\
&= \langle \sigma_{\alpha_1}\sigma_{\alpha_2}\rangle_{G^{(1)}}\,\langle \sigma_{\alpha_3}\sigma_{\alpha_4}\rangle_{G^{(1)}} \ ,
\end{aligned}$$

where it's clear each neuron factorizes and gives separate Gaussian integrals. This illustrates the fact that neurons are independent, and thus there is no interaction among different neurons in the first layer. In deeper layers the preactivation distributions are non-Gaussian and things will be a bit more complicated.

4.2 Second Layer: Genesis of Non-Gaussianity

In this section, we'll move onto evaluating the distribution of preactivations in the second layer of an MLP. The second-layer preactivations are defined via

$$z_{i;\alpha}^{(2)} \equiv z_i^{(2)}(x_\alpha) = b_i^{(2)} + \sum_{j=1}^{n_1} W_{ij}^{(2)}\sigma_{j;\alpha}^{(1)} \,, \quad \text{for} \quad i = 1,\ldots,n_2 \,, \tag{4.30}$$

with the first-layer activations denoted as

$$\sigma_{i;\alpha}^{(1)} \equiv \sigma\left(z_{i;\alpha}^{(1)}\right) \ , \tag{4.31}$$

and the biases $b^{(2)}$ and weights $W^{(2)}$ sampled from Gaussian distributions.

The joint distribution of preactivations in the first and second layers can be factorized as

$$p\left(z^{(2)}, z^{(1)}\middle|\mathcal{D}\right) = p\left(z^{(2)}\middle|z^{(1)}\right)p\left(z^{(1)}\middle|\mathcal{D}\right) \ . \tag{4.32}$$

Here the first-layer marginal distribution $p\Big(z^{(1)}\Big|\mathcal{D}\Big)$ was evaluated in the last section, §4.1, to be a Gaussian distribution (4.23) with the variance given in terms of the first-layer metric $G^{(1)}_{\alpha_1\alpha_2}$. As for the conditional distribution, we know that it can be expressed as[4]

$$p\Big(z^{(2)}\Big|z^{(1)}\Big) \tag{4.33}$$
$$= \int \left[\prod_i db_i^{(2)}\; p\Big(b_i^{(2)}\Big)\right]\left[\prod_{i,j} dW_{ij}^{(2)}\; p\Big(W_{ij}^{(2)}\Big)\right]\prod_{i,\alpha}\delta\left(z^{(2)}_{i;\alpha} - b_i^{(2)} - \sum_j W_{ij}^{(2)}\sigma^{(1)}_{j;\alpha}\right),$$

from the formal expression (2.34) for the preactivation distribution conditioned on the activations in the previous layer. The marginal distribution of the second-layer preactivations can then be obtained by **marginalizing over** or **integrating out** the first-layer preactivations as

$$p\Big(z^{(2)}\Big|\mathcal{D}\Big) = \int\left[\prod_{i,\alpha} dz^{(1)}_{i;\alpha}\right]\; p\Big(z^{(2)}\Big|z^{(1)}\Big)\, p\Big(z^{(1)}\Big|\mathcal{D}\Big)\;. \tag{4.34}$$

To evaluate this expression for the marginal distribution $p\Big(z^{(2)}\Big|\mathcal{D}\Big)$, first we'll discuss how to treat the conditional distribution $p\Big(z^{(2)}\Big|z^{(1)}\Big)$, and then we'll explain how to integrate over the first-layer preactivations $z^{(1)}$ governed by the Gaussian distribution (4.23).

Second-layer conditional distribution

The conditional distribution (4.33) can be evaluated exactly in the same way as we evaluated the first-layer distribution (4.16) conditioned on the inputs, with the simple replacement of the layer indices ℓ as $1 \to 2$ and exchanging the network input for the first-layer preactivation as $x_{j;\alpha} \to \sigma^{(1)}_{j;\alpha}$. Giving you a moment to flip back to (4.16) to make these substitutions and then remind yourself of the answer (4.23), it's easy to see that this evaluation yields

$$p\Big(z^{(2)}\Big|z^{(1)}\Big) = \frac{1}{\sqrt{\Big|2\pi\widehat{G}^{(2)}\Big|^{n_2}}}\exp\left(-\frac{1}{2}\sum_{i=1}^{n_2}\sum_{\alpha_1,\alpha_2\in\mathcal{D}}\widehat{G}^{\alpha_1\alpha_2}_{(2)}z^{(2)}_{i;\alpha_1}z^{(2)}_{i;\alpha_2}\right), \tag{4.35}$$

where we have defined the *stochastic* second-layer metric

$$\widehat{G}^{(2)}_{\alpha_1\alpha_2} \equiv C_b^{(2)} + C_W^{(2)}\frac{1}{n_1}\sum_{j=1}^{n_1}\sigma^{(1)}_{j;\alpha_1}\sigma^{(1)}_{j;\alpha_2}\,, \tag{4.36}$$

with a hat to emphasize that it is a random variable that depends on the stochastic variable $z^{(1)}$ through $\sigma^{(1)} \equiv \sigma\Big(z^{(1)}\Big)$. Thus, we see that the second-layer conditional

[4] Again, the expression in the Dirac delta function is specific to multilayer perceptron architectures, but this formalism can easily be adapted for other architectures.

distribution (4.35) is a Gaussian whose variance itself is a random variable. In particular, the stochastic second-layer metric fluctuates around the *mean* second-layer metric

$$G^{(2)}_{\alpha_1\alpha_2} \equiv \mathbb{E}\left[\widehat{G}^{(2)}_{\alpha_1\alpha_2}\right] = C_b^{(2)} + C_W^{(2)} \frac{1}{n_1}\sum_{j=1}^{n_1} \mathbb{E}\left[\sigma^{(1)}_{j;\alpha_1}\sigma^{(1)}_{j;\alpha_2}\right] \tag{4.37}$$
$$= C_b^{(2)} + C_W^{(2)} \left\langle \sigma_{\alpha_1}\sigma_{\alpha_2}\right\rangle_{G^{(1)}} ,$$

where in the last step we recalled the result (4.27) for evaluating the two-point correlator of the first-layer activations on the same neuron.

Around this mean, we define the fluctuation of the second-layer metric as

$$\widehat{\Delta G}^{(2)}_{\alpha_1\alpha_2} \equiv \widehat{G}^{(2)}_{\alpha_1\alpha_2} - G^{(2)}_{\alpha_1\alpha_2} = C_W^{(2)} \frac{1}{n_1}\sum_{j=1}^{n_1} \left(\sigma^{(1)}_{j;\alpha_1}\sigma^{(1)}_{j;\alpha_2} - \left\langle \sigma_{\alpha_1}\sigma_{\alpha_2}\right\rangle_{G^{(1)}}\right) , \tag{4.38}$$

which by construction has the mean zero when averaged over the first-layer preactivations,

$$\mathbb{E}\left[\widehat{\Delta G}^{(2)}_{\alpha_1\alpha_2}\right] = 0\,. \tag{4.39}$$

The typical size of the fluctuations is given by its two-point correlator. Recalling the expressions we derived for Gaussian integrals (4.27) and (4.28) of two and four activations on the same neuron and their factorization property on separate neurons (4.29), we obtain

$$\begin{aligned}
&\mathbb{E}\left[\widehat{\Delta G}^{(2)}_{\alpha_1\alpha_2}\widehat{\Delta G}^{(2)}_{\alpha_3\alpha_4}\right] \qquad (4.40)\\
=&\left(\frac{C_W^{(2)}}{n_1}\right)^2 \sum_{j,k=1}^{n_1} \mathbb{E}\left[\left(\sigma^{(1)}_{j;\alpha_1}\sigma^{(1)}_{j;\alpha_2} - \mathbb{E}\left[\sigma^{(1)}_{j;\alpha_1}\sigma^{(1)}_{j;\alpha_2}\right]\right)\left(\sigma^{(1)}_{k;\alpha_3}\sigma^{(1)}_{k;\alpha_4} - \mathbb{E}\left[\sigma^{(1)}_{k;\alpha_3}\sigma^{(1)}_{k;\alpha_4}\right]\right)\right]\\
=&\left(\frac{C_W^{(2)}}{n_1}\right)^2 \sum_{j=1}^{n_1} \left\{\mathbb{E}\left[\sigma^{(1)}_{j;\alpha_1}\sigma^{(1)}_{j;\alpha_2}\sigma^{(1)}_{j;\alpha_3}\sigma^{(1)}_{j;\alpha_4}\right] - \mathbb{E}\left[\sigma^{(1)}_{j;\alpha_1}\sigma^{(1)}_{j;\alpha_2}\right]\mathbb{E}\left[\sigma^{(1)}_{j;\alpha_3}\sigma^{(1)}_{j;\alpha_4}\right]\right\}\\
=&\frac{1}{n_1}\left(C_W^{(2)}\right)^2 \left[\left\langle \sigma_{\alpha_1}\sigma_{\alpha_2}\sigma_{\alpha_3}\sigma_{\alpha_4}\right\rangle_{G^{(1)}} - \left\langle \sigma_{\alpha_1}\sigma_{\alpha_2}\right\rangle_{G^{(1)}}\left\langle \sigma_{\alpha_3}\sigma_{\alpha_4}\right\rangle_{G^{(1)}}\right]\\
\equiv&\frac{1}{n_1} V^{(2)}_{(\alpha_1\alpha_2)(\alpha_3\alpha_4)} ,
\end{aligned}$$

where at the end we introduced the second-layer **four-point vertex** $V^{(2)}_{(\alpha_1\alpha_2)(\alpha_3\alpha_4)} = V\left(x_{\alpha_1}, x_{\alpha_2}; x_{\alpha_3}, x_{\alpha_4}\right)$, which depends on four input data points and is symmetric under the exchanges of sample indices $\alpha_1 \leftrightarrow \alpha_2$, $\alpha_3 \leftrightarrow \alpha_4$, and $(\alpha_1, \alpha_2) \leftrightarrow (\alpha_3, \alpha_4)$. We will understand the significance of this quantity soon in a future equation, (4.43).

Here, we also see our first hint of simplification in the wide regime $n_1 \gg 1$: since the four-point vertex here is manifestly of order one, we see that the metric fluctuation will be suppressed in that regime. Essentially, as the number of neurons in the first layer grows, the metric fluctuation becomes more and more Gaussian due to the central limit

theorem. In the strict limit of infinite n_1, the metric would *self-average*, meaning that the fluctuation would vanish.

Now that we have a feel for the distribution of metric fluctuations, we are only too ready to actually integrate out the first-layer preactivations $z^{(1)}$ and obtain the marginal distribution of the second-layer preactivations $p\Big(z^{(2)}\Big|\mathcal{D}\Big)$. We again provide two derivations, one brute-force and the other clever.

Wick Wick Wick: combinatorial derivation

The correlators of the second-layer preactivations can be written nicely in terms of the expectations of the stochastic metric that we just computed. In order to compute the correlators, first we use the fact that the conditional distribution $p\Big(z^{(2)}\Big|z^{(1)}\Big)$ is Gaussian (4.35) to Wick contract the second-layer preactivations $z^{(2)}$, resulting in expressions involving expectations of the stochastic metric $\widehat{G}^{(2)}_{\alpha_1\alpha_2}$; we then insert expressions for the expectations of the stochastic metric obtained above.

With this in mind, the two-point correlator of the second-layer preactivations is given by

$$\mathbb{E}\left[z^{(2)}_{i_1;\alpha_1}z^{(2)}_{i_2;\alpha_2}\right]=\delta_{i_1i_2}\mathbb{E}\left[\widehat{G}^{(2)}_{\alpha_1\alpha_2}\right]=\delta_{i_1i_2}G^{(2)}_{\alpha_1\alpha_2}=\delta_{i_1i_2}\left(C_b^{(2)}+C_W^{(2)}\left\langle\sigma_{\alpha_1}\sigma_{\alpha_2}\right\rangle_{G^{(1)}}\right), \tag{4.41}$$

where to be clear we first used (4.35) to do the single Wick contraction and then inserted the expression (4.37) for the mean of the stochastic metric.

Similarly, the full four-point function can be evaluated as

$$\begin{aligned}&\mathbb{E}\left[z^{(2)}_{i_1;\alpha_1}z^{(2)}_{i_2;\alpha_2}z^{(2)}_{i_3;\alpha_3}z^{(2)}_{i_4;\alpha_4}\right] \\ =&\delta_{i_1i_2}\delta_{i_3i_4}\mathbb{E}\left[\widehat{G}^{(2)}_{\alpha_1\alpha_2}\widehat{G}^{(2)}_{\alpha_3\alpha_4}\right]+\delta_{i_1i_3}\delta_{i_2i_4}\mathbb{E}\left[\widehat{G}^{(2)}_{\alpha_1\alpha_3}\widehat{G}^{(2)}_{\alpha_2\alpha_4}\right]+\delta_{i_1i_4}\delta_{i_2i_3}\mathbb{E}\left[\widehat{G}^{(2)}_{\alpha_1\alpha_4}\widehat{G}^{(2)}_{\alpha_2\alpha_3}\right], \\ =&\delta_{i_1i_2}\delta_{i_3i_4}G^{(2)}_{\alpha_1\alpha_2}G^{(2)}_{\alpha_3\alpha_4}+\delta_{i_1i_3}\delta_{i_2i_4}G^{(2)}_{\alpha_1\alpha_3}G^{(2)}_{\alpha_2\alpha_4}+\delta_{i_1i_4}\delta_{i_2i_3}G^{(2)}_{\alpha_1\alpha_4}G^{(2)}_{\alpha_2\alpha_3} \\ &+\frac{1}{n_1}\left[\delta_{i_1i_2}\delta_{i_3i_4}V^{(2)}_{(\alpha_1\alpha_2)(\alpha_3\alpha_4)}+\delta_{i_1i_3}\delta_{i_2i_4}V^{(2)}_{(\alpha_1\alpha_3)(\alpha_2\alpha_4)}+\delta_{i_1i_4}\delta_{i_2i_3}V^{(2)}_{(\alpha_1\alpha_4)(\alpha_2\alpha_3)}\right],\end{aligned} \tag{4.42}$$

where in the first line we made three Wick contractions of the four second-layer preactivations $z^{(2)}$'s using the Gaussian distribution (4.35), and then in the second line we recalled (4.39) and (4.40) for the expectations of the stochastic metric $\widehat{G}^{(2)}_{\alpha_1\alpha_2}=G^{(2)}_{\alpha_1\alpha_2}+\widehat{\Delta G}^{(2)}_{\alpha_1\alpha_2}$ over the first-layer preactivations $z^{(1)}$. This means that the *connected* four-point correlator – recall (1.54) – after subtracting the contributions from the two-point correlators of the second-layer preactivations is given by

$$\begin{aligned}&\mathbb{E}\left[z^{(2)}_{i_1;\alpha_1}z^{(2)}_{i_2;\alpha_2}z^{(2)}_{i_3;\alpha_3}z^{(2)}_{i_4;\alpha_4}\right]\Big|_{\text{connected}} \\ =&\frac{1}{n_1}\left[\delta_{i_1i_2}\delta_{i_3i_4}V^{(2)}_{(\alpha_1\alpha_2)(\alpha_3\alpha_4)}+\delta_{i_1i_3}\delta_{i_2i_4}V^{(2)}_{(\alpha_1\alpha_3)(\alpha_2\alpha_4)}+\delta_{i_1i_4}\delta_{i_2i_3}V^{(2)}_{(\alpha_1\alpha_4)(\alpha_2\alpha_3)}\right].\end{aligned} \tag{4.43}$$

Here we see the true importance of the four-point vertex we introduced in (4.40); it gives the connected second-layer four-point correlator and controls the non-Gaussianity of the

second-layer preactivation distribution. Thus, we see that this connected correlator is suppressed in the wide regime of $n_1 \gg 1$, suggesting that the preactivation distribution will become more and more Gaussian as the network gets wider and wider. Given this, we see that the second-layer preactivation distribution $p\left(z^{(2)}\middle|\mathcal{D}\right)$ is in general *non-Gaussian* but also simplifies significantly in the large-n_1 regime, becoming Gaussian in the strict $n_1 = \infty$ limit and with the four-point vertex $V^{(2)}_{(\alpha_1\alpha_3)(\alpha_2\alpha_4)}$ measuring the leading deviation from Gaussianity.

To complete our combinatorial derivation, we need to find an action that generates correlations (4.41) and (4.43). As we know, a quadratic action cannot generate non-Gaussian distributions with nontrivial connected four-point correlators, so we need a different action that's appropriate for a non-Gaussian distribution. Intuition from single-variable non-Gaussian integrals in §1.2 suggests that we could perhaps generate the requisite correlations by including a quartic term in the action.

With that in mind, let's start with a quartic action for an $(nN_{\mathcal{D}})$-dimensional random variable z

$$\begin{aligned} S[z] = &\frac{1}{2}\sum_{\alpha_1,\alpha_2\in\mathcal{D}} g^{\alpha_1\alpha_2}\sum_{i=1}^{n} z_{i;\alpha_1} z_{i;\alpha_2} \\ &- \frac{1}{8}\sum_{\alpha_1,\ldots,\alpha_4\in\mathcal{D}} v^{(\alpha_1\alpha_2)(\alpha_3\alpha_4)}\sum_{i_1,i_2=1}^{n} z_{i_1;\alpha_1} z_{i_1;\alpha_2}\, z_{i_2;\alpha_3} z_{i_2;\alpha_4}\,, \end{aligned} \tag{4.44}$$

with undetermined couplings g and v. We will treat the quartic coupling v perturbatively, an assumption that we will justify later by relating the quartic coupling v to the $1/n_1$-suppressed connected four-point correlator. Note that by construction the quartic coupling $v^{(\alpha_1\alpha_2)(\alpha_3\alpha_4)}$ has the same symmetric structure as the four-point vertex $V^{(2)}_{(\alpha_1\alpha_2)(\alpha_3\alpha_4)}$ with respect to the sample indices.[5] Using this action, we can compute to the first order in v the two-point and four-point correlators. Then, by matching with the expressions (4.41) and (4.43) for these quantities, we'll learn how to adjust the couplings g and v to reproduce the right statistics of second-layer preactivations in the wide regime.

Before proceeding further, it is convenient to introduce some notation. In (4.25), we defined $\langle F(z_{\alpha_1},\ldots,z_{\alpha_m})\rangle_g$ for the average of an arbitrary function F over a Gaussian distribution with variance g, where preactivation variables z_α have sample indices *only*. In addition, we here define

$$\begin{aligned} &\langle\!\langle F(z_{i_1;\alpha_1},\ldots,z_{i_m;\alpha_m})\rangle\!\rangle_g \\ &\equiv \int\left[\prod_{i=1}^{n}\frac{\prod_{\alpha\in\mathcal{D}} dz_{i;\alpha}}{\sqrt{|2\pi g|}}\right]\exp\left(-\frac{1}{2}\sum_{j=1}^{n}\sum_{\beta_1,\beta_2\in\mathcal{D}} g^{\beta_1\beta_2} z_{j;\beta_1} z_{j;\beta_2}\right) F(z_{i_1;\alpha_1},\ldots,z_{i_m;\alpha_m})\,, \end{aligned} \tag{4.45}$$

which now includes neural indices. As we saw while working through (4.27) and (4.29), this type of average factorizes into integrals of the form (4.25) for each neuron.

[5] The conventional factor of 1/8 in (4.44) is to account for this symmetry.

With this notation in hand, the expectation of an arbitrary function $F(z_{i_1;\alpha_1},\ldots,z_{i_m;\alpha_m})$ against a distribution with the quartic action (4.44) can be rewritten in terms of Gaussian expectations, enabling the perturbative expansion in the coupling v as

$$\begin{aligned}
&\mathbb{E}\left[F(z_{i_1;\alpha_1},\ldots,z_{i_m;\alpha_m})\right] \qquad (4.46)\\
=&\frac{\int\left[\prod_{i,\alpha}dz_{i;\alpha}\right]e^{-S(z)}F(z_{i_1;\alpha_1},\ldots,z_{i_m;\alpha_m})}{\int\left[\prod_{i,\alpha}dz_{i;\alpha}\right]e^{-S(z)}}\\
=&\frac{\left\langle\!\!\left\langle \exp\left\{\frac{1}{8}\sum_{\beta_1,\ldots,\beta_4\in\mathcal{D}}v^{(\beta_1\beta_2)(\beta_3\beta_4)}\sum_{j_1,j_2=1}^{n}z_{j_1;\beta_1}z_{j_1;\beta_2}\,z_{j_2;\beta_3}z_{j_2;\beta_4}\right\}F(z_{i_1;\alpha_1},\ldots,z_{i_m;\alpha_m})\right\rangle\!\!\right\rangle_g}{\left\langle\!\!\left\langle \exp\left\{\frac{1}{8}\sum_{\beta_1,\ldots,\beta_4\in\mathcal{D}}v^{(\beta_1\beta_2)(\beta_3\beta_4)}\sum_{j_1,j_2=1}^{n}z_{j_1;\beta_1}z_{j_1;\beta_2}\,z_{j_2;\beta_3}z_{j_2;\beta_4}\right\}\right\rangle\!\!\right\rangle_g}\\
=&\left\langle\!\!\left\langle F(z_{i_1;\alpha_1},\ldots,z_{i_m;\alpha_m})\right\rangle\!\!\right\rangle_g\\
&+\frac{1}{8}\sum_{\beta_1,\ldots,\beta_4\in\mathcal{D}}v^{(\beta_1\beta_2)(\beta_3\beta_4)}\sum_{j_1,j_2=1}^{n}\Big[\left\langle\!\!\left\langle z_{j_1;\beta_1}z_{j_1;\beta_2}\,z_{j_2;\beta_3}z_{j_2;\beta_4}F(z_{i_1;\alpha_1},\ldots,z_{i_m;\alpha_m})\right\rangle\!\!\right\rangle_g\\
&\qquad\qquad-\left\langle\!\!\left\langle z_{j_1;\beta_1}z_{j_1;\beta_2}\,z_{j_2;\beta_3}z_{j_2;\beta_4}\right\rangle\!\!\right\rangle_g\left\langle\!\!\left\langle F(z_{i_1;\alpha_1},\ldots,z_{i_m;\alpha_m})\right\rangle\!\!\right\rangle_g\Big]\\
&+O\Big(v^2\Big)\,,
\end{aligned}$$

where in the first line we used the definition of the expectation, in the second line we rewrote the numerator and denominator using the notation (4.45) that we just introduced, and in the third line we expanded the exponential in the coupling v, both in the denominator and numerator. In short, this tells us how to perturbatively express an expectation against the full distribution with the quartic action (4.44) in terms of the leading Gaussian expectation and perturbative corrections; these perturbative contributions nonetheless involve only Gaussian expectations and hence are easy to evaluate.

With this in mind, let's consider some particular choices for F. Starting with the two-point correlator, we get

$$\begin{aligned}
&\mathbb{E}\left[z_{i_1;\alpha_1}z_{i_2;\alpha_2}\right] \qquad (4.47)\\
=&\delta_{i_1i_2}\left[g_{\alpha_1\alpha_2}+\frac{1}{2}\sum_{\beta_1,\ldots,\beta_4\in\mathcal{D}}v^{(\beta_1\beta_2)(\beta_3\beta_4)}\left(ng_{\alpha_1\beta_1}g_{\alpha_2\beta_2}g_{\beta_3\beta_4}+2g_{\alpha_1\beta_1}g_{\alpha_2\beta_3}g_{\beta_2\beta_4}\right)\right]+O\Big(v^2\Big)\,.
\end{aligned}$$

Here the variance $g_{\alpha_1\alpha_2}$ is the inverse of the quadratic coupling, with $\sum_\beta g_{\alpha_1\beta}g^{\beta\alpha_2}=\delta_{\alpha_1}^{\alpha_2}$.

Similarly, we find that the connected four-point correlator evaluates to

$$\begin{aligned}
&\mathbb{E}\left[z_{i_1;\alpha_1} z_{i_2;\alpha_2} z_{i_3;\alpha_3} z_{i_4;\alpha_4}\right]\Big|_{\text{connected}} \qquad (4.48)\\
\equiv&\mathbb{E}\left[z_{i_1;\alpha_1} z_{i_2;\alpha_2} z_{i_3;\alpha_1} z_{i_4;\alpha_2}\right] - \mathbb{E}\left[z_{i_1;\alpha_1} z_{i_2;\alpha_2}\right]\mathbb{E}\left[z_{i_3;\alpha_3} z_{i_4;\alpha_4}\right]\\
&- \mathbb{E}\left[z_{i_1;\alpha_1} z_{i_3;\alpha_3}\right]\mathbb{E}\left[z_{i_2;\alpha_2} z_{i_4;\alpha_4}\right] - \mathbb{E}\left[z_{i_1;\alpha_1} z_{i_4;\alpha_4}\right]\mathbb{E}\left[z_{i_2;\alpha_2} z_{i_3;\alpha_3}\right]\\
=&\delta_{i_1 i_2}\delta_{i_3 i_4}\sum_{\beta_1,\ldots,\beta_4\in\mathcal{D}} v^{(\beta_1\beta_2)(\beta_3\beta_4)} g_{\alpha_1\beta_1} g_{\alpha_2\beta_2} g_{\alpha_3\beta_3} g_{\alpha_4\beta_4}\\
&+ \delta_{i_1 i_3}\delta_{i_2 i_4}\sum_{\beta_1,\ldots,\beta_4\in\mathcal{D}} v^{(\beta_1\beta_3)(\beta_2\beta_4)} g_{\alpha_1\beta_1} g_{\alpha_3\beta_3} g_{\alpha_2\beta_2} g_{\alpha_4\beta_4}\\
&+ \delta_{i_1 i_4}\delta_{i_2 i_3}\sum_{\beta_1,\ldots,\beta_4\in\mathcal{D}} v^{(\beta_1\beta_4)(\beta_2\beta_3)} g_{\alpha_1\beta_1} g_{\alpha_4\beta_4} g_{\alpha_2\beta_2} g_{\alpha_3\beta_3} + O\Big(v^2\Big)\,.
\end{aligned}$$

Comparing these expressions, (4.47) and (4.48), with correlators in the second layer, (4.41) and (4.43), it's easy to see that setting the couplings as

$$g^{\alpha_1\alpha_2} = G_{(2)}^{\alpha_1\alpha_2} + O\Big(\frac{1}{n_1}\Big)\,, \qquad (4.49)$$

$$v^{(\alpha_1\alpha_2)(\alpha_3\alpha_4)} = \frac{1}{n_1} V_{(2)}^{(\alpha_1\alpha_2)(\alpha_3\alpha_4)} + O\Big(\frac{1}{n_1^2}\Big)\,, \qquad (4.50)$$

reproduces the second-layer preactivation correlators to the leading order in $1/n_1$, with the marginal distribution

$$p\left(z^{(2)}\middle|\mathcal{D}\right) = \frac{1}{Z} e^{-S\left(z^{(2)}\right)} \qquad (4.51)$$

and quartic action (4.44). Here for convenience we have defined a version of the four-point vertex with indices *raised* by the inverse of the second-layer mean metric

$$V_{(2)}^{(\alpha_1\alpha_2)(\alpha_3\alpha_4)} \equiv \sum_{\beta_1,\ldots,\beta_4} G_{(2)}^{\alpha_1\beta_1} G_{(2)}^{\alpha_2\beta_2} G_{(2)}^{\alpha_3\beta_3} G_{(2)}^{\alpha_4\beta_4} V^{(2)}_{(\beta_1\beta_2)(\beta_3\beta_4)}\,. \qquad (4.52)$$

Note that the quartic coupling v is $O(1/n_1)$, justifying our earlier perturbative treatment of the coupling for wide networks. Note also that these couplings – the inverse metric $G_{(2)}^{\alpha_1\alpha_2}$ and the quartic coupling $V_{(2)}^{(\alpha_1\alpha_2)(\alpha_3\alpha_4)}$ – are input-dependent. In particular, the effective strength of interaction between neurons is set by the particular set of inputs to the network.

This completes our first combinatorial derivation of the second-layer preactivation distribution.

Schwinger-Dyson this way: algebraic derivation

Here is a neat way to derive the action for the second-layer preactivation distribution. Plugging the conditional distribution (4.35) into the marginalization equation (4.34), the second-layer marginal distribution becomes

$$p\Big(z^{(2)}\Big|\mathcal{D}\Big) = \int\left[\prod_{i,\alpha} dz_{i;\alpha}^{(1)}\right] p\Big(z^{(1)}\Big|\mathcal{D}\Big)\,\frac{\exp\Big(-\frac{1}{2}\sum_{j=1}^{n_2}\sum_{\alpha_1,\alpha_2\in\mathcal{D}}\widehat{G}_{(2)}^{\alpha_1\alpha_2} z_{j;\alpha_1}^{(2)} z_{j;\alpha_2}^{(2)}\Big)}{\sqrt{\Big|2\pi\widehat{G}^{(2)}\Big|^{n_2}}}\,. \qquad (4.53)$$

We saw that the stochastic metric has a natural decomposition into mean and fluctuating parts as

$$\widehat{G}^{(2)}_{\alpha_1\alpha_2} = G^{(2)}_{\alpha_1\alpha_2} + \widehat{\Delta G}^{(2)}_{\alpha_1\alpha_2} \,. \tag{4.54}$$

Inverting this matrix to the second order in the fluctuation around the mean, we get the inverse stochastic metric[6]

$$\begin{aligned}\widehat{G}^{\alpha_1\alpha_2}_{(2)} =& G^{\alpha_1\alpha_2}_{(2)} - \sum_{\beta_1,\beta_2\in\mathcal{D}} G^{\alpha_1\beta_1}_{(2)} \widehat{\Delta G}^{(2)}_{\beta_1\beta_2} G^{\beta_2\alpha_2}_{(2)} \\ &+ \sum_{\beta_1,\ldots,\beta_4\in\mathcal{D}} G^{\alpha_1\beta_1}_{(2)} \widehat{\Delta G}^{(2)}_{\beta_1\beta_2} G^{\beta_2\beta_3}_{(2)} \widehat{\Delta G}^{(2)}_{\beta_3\beta_4} G^{\beta_4\alpha_2}_{(2)} + O\Big(\Delta^3\Big) \,.\end{aligned} \tag{4.55}$$

Putting this into the exponential that appears in the integrand of the marginal distribution (4.53) and Taylor-expanding in the fluctuation $\widehat{\Delta G}^{(2)}_{\alpha_1\alpha_2}$, we find

$$\begin{aligned}&\exp\left(-\frac{1}{2}\sum_{j=1}^{n_2}\sum_{\alpha_1,\alpha_2\in\mathcal{D}} \widehat{G}^{\alpha_1\alpha_2}_{(2)} z^{(2)}_{j;\alpha_1} z^{(2)}_{j;\alpha_2}\right) \\ =&\exp\left(-\frac{1}{2}\sum_{j=1}^{n_2}\sum_{\alpha_1,\alpha_2\in\mathcal{D}} G^{\alpha_1\alpha_2}_{(2)} z^{(2)}_{j;\alpha_1} z^{(2)}_{j;\alpha_2}\right) \\ &\times\Bigg\{1 + \frac{1}{2}\sum_{i=1}^{n_2}\sum_{\alpha_1,\alpha_2\in\mathcal{D}}\left(\sum_{\beta_1,\beta_2\in\mathcal{D}} G^{\alpha_1\beta_1}_{(2)} \widehat{\Delta G}^{(2)}_{\beta_1\beta_2} G^{\beta_2\alpha_2}_{(2)}\right) z^{(2)}_{i;\alpha_1} z^{(2)}_{i;\alpha_2} \\ &\quad - \frac{1}{2}\sum_{i=1}^{n_2}\sum_{\alpha_1,\alpha_2\in\mathcal{D}}\left(\sum_{\beta_1,\ldots,\beta_4\in\mathcal{D}} G^{\alpha_1\beta_1}_{(2)} \widehat{\Delta G}^{(2)}_{\beta_1\beta_2} G^{\beta_2\beta_3}_{(2)} \widehat{\Delta G}^{(2)}_{\beta_3\beta_4} G^{\beta_4\alpha_2}_{(2)}\right) z^{(2)}_{i;\alpha_1} z^{(2)}_{i;\alpha_2} \\ &\quad + \frac{1}{2!}\left(\frac{1}{2}\right)^2 \sum_{i_1,i_2=1}^{n_2}\sum_{\alpha_1,\ldots,\beta_4\in\mathcal{D}} G^{\alpha_1\beta_1}_{(2)}\cdots G^{\alpha_4\beta_4}_{(2)} \widehat{\Delta G}^{(2)}_{\beta_1\beta_2} \widehat{\Delta G}^{(2)}_{\beta_3\beta_4} z^{(2)}_{i_1;\alpha_1} z^{(2)}_{i_1;\alpha_2} z^{(2)}_{i_2;\alpha_3} z^{(2)}_{i_2;\alpha_4} + O\Big(\Delta^3\Big)\Bigg\} .\end{aligned} \tag{4.56}$$

Using this expression, the determinant in the denominator becomes

$$\begin{aligned}&\sqrt{\left|2\pi\widehat{G}^{(2)}\right|^{n_2}} = \int\left[\prod_{i,\alpha} dz^{(2)}_{i;\alpha}\right]\exp\left(-\frac{1}{2}\sum_{j=1}^{n_2}\sum_{\alpha_1,\alpha_2\in\mathcal{D}} \widehat{G}^{\alpha_1\alpha_2}_{(2)} z^{(2)}_{j;\alpha_1} z^{(2)}_{j;\alpha_2}\right) \\ =&\sqrt{\left|2\pi G^{(2)}\right|^{n_2}}\Bigg[1 + \frac{n_2}{2}\sum_{\beta_1,\beta_2\in\mathcal{D}} \widehat{\Delta G}^{(2)}_{\beta_1\beta_2} G^{\beta_1\beta_2}_{(2)} \\ &\quad + \sum_{\beta_1,\ldots,\beta_4\in\mathcal{D}} \widehat{\Delta G}^{(2)}_{\beta_1\beta_2} \widehat{\Delta G}^{(2)}_{\beta_3\beta_4}\left(\frac{n_2^2}{8} G^{\beta_1\beta_2}_{(2)} G^{\beta_3\beta_4}_{(2)} - \frac{n_2}{4} G^{\beta_1\beta_3}_{(2)} G^{\beta_2\beta_4}_{(2)}\right) + O\Big(\Delta^3\Big)\Bigg] ,\end{aligned} \tag{4.57}$$

[6] This together with the defining equation for the metric fluctuation (4.38) are sometimes called the Schwinger-Dyson equations [30, 31] from which this subsubsection takes its title.

where on the first line we re-expressed the determinant as a Gaussian integral, and on the subsequent line we plugged in (4.56) and integrated over the second-layer preactivations $z^{(2)}$.

Next, plugging these two expressions (4.56) and (4.57) back into our expression for the second-layer distribution (4.53), we can now integrate out the first-layer preactivations, giving

$$\begin{aligned} p\Big(z^{(2)}\Big|\mathcal{D}\Big) =& \frac{1}{\sqrt{\left|2\pi G^{(2)}\right|^{n_2}}} \exp\left(-\frac{1}{2}\sum_{j=1}^{n_2}\sum_{\alpha_1,\alpha_2\in\mathcal{D}} G^{\alpha_1\alpha_2}_{(2)} z^{(2)}_{j;\alpha_1} z^{(2)}_{j;\alpha_2}\right) \\ & \times\left\{\left[1+O\Big(\frac{1}{n_1}\Big)\right] + \sum_{i=1}^{n_2}\sum_{\alpha_1,\alpha_2\in\mathcal{D}}\left[O\Big(\frac{1}{n_1}\Big)\right] z^{(2)}_{i_1;\alpha_1} z^{(2)}_{i_1;\alpha_2} \right. \\ & \left. + \frac{1}{8n_1}\sum_{i_1,i_2=1}^{n_2}\sum_{\alpha_1,\ldots,\alpha_4\in\mathcal{D}} V^{(\alpha_1\alpha_2)(\alpha_3\alpha_4)}_{(2)} z^{(2)}_{i_1;\alpha_1} z^{(2)}_{i_1;\alpha_2} z^{(2)}_{i_2;\alpha_3} z^{(2)}_{i_2;\alpha_4}\right\} + O\Big(\frac{1}{n_1^2}\Big)\,, \end{aligned} \tag{4.58}$$

where we have used the fact that expectations of the metric fluctuation are given by $\mathbb{E}\left[\widehat{\Delta G}^{(2)}_{\beta_1\beta_2}\right] = 0$ and $\mathbb{E}\left[\widehat{\Delta G}^{(2)}_{\beta_1\beta_2}\widehat{\Delta G}^{(2)}_{\beta_3\beta_4}\right] = \frac{1}{n_1}V^{(2)}_{(\beta_1\beta_2)(\beta_3\beta_4)}$.[7] Taking the logarithm to isolate the action and absorbing the irrelevant constant terms into the partition function, we arrive at the correct expression for the second-layer quartic action to leading order in the first layer width

$$\begin{aligned} S(z) =& \frac{1}{2}\sum_{\alpha_1,\alpha_2\in\mathcal{D}}\left[G^{\alpha_1\alpha_2}_{(2)} + O\Big(\frac{1}{n_1}\Big)\right]\sum_{i=1}^{n_2} z_{i;\alpha_1} z_{i;\alpha_2} \\ & - \frac{1}{8}\sum_{\alpha_1,\ldots,\alpha_4\in\mathcal{D}}\frac{1}{n_1}V^{(\alpha_1\alpha_2)(\alpha_3\alpha_4)}_{(2)}\sum_{i_1,i_2=1}^{n_2} z_{i_1;\alpha_1} z_{i_1;\alpha_2} z_{i_2;\alpha_3} z_{i_2;\alpha_4} + O\Big(\frac{1}{n_1^2}\Big)\,. \end{aligned} \tag{4.60}$$

Here, a prudent reader might wonder about our dropping of the $1/n_1$ correction to the quadratic coupling, while keeping the quartic coupling despite being of the same order. The main reason for this is that such a correction is a *subleading* contribution

[7] We tacitly assumed that the expectation of $\widehat{\Delta G}^{m\geq 3}$ are of order $O\big(1/n_1^2\big)$ or greater. For instance, you can follow exactly the same steps as in (4.40) and compute

$$\begin{aligned} & \mathbb{E}\left[\widehat{\Delta G}^{(2)}_{\beta_1\beta_2}\widehat{\Delta G}^{(2)}_{\beta_3\beta_4}\widehat{\Delta G}^{(2)}_{\beta_5\beta_6}\right] \\ =& \frac{1}{n_1^2}\left(C_W^{(2)}\right)^3\Big[\left\langle\sigma_{\alpha_1}\sigma_{\alpha_2}\sigma_{\alpha_3}\sigma_{\alpha_4}\sigma_{\alpha_5}\sigma_{\alpha_6}\right\rangle_{G^{(1)}} - \left\langle\sigma_{\alpha_1}\sigma_{\alpha_2}\right\rangle_{G^{(1)}}\left\langle\sigma_{\alpha_3}\sigma_{\alpha_4}\sigma_{\alpha_5}\sigma_{\alpha_6}\right\rangle_{G^{(1)}} \\ & - \left\langle\sigma_{\alpha_3}\sigma_{\alpha_4}\right\rangle_{G^{(1)}}\left\langle\sigma_{\alpha_5}\sigma_{\alpha_6}\sigma_{\alpha_1}\sigma_{\alpha_2}\right\rangle_{G^{(1)}} - \left\langle\sigma_{\alpha_5}\sigma_{\alpha_6}\right\rangle_{G^{(1)}}\left\langle\sigma_{\alpha_1}\sigma_{\alpha_2}\sigma_{\alpha_3}\sigma_{\alpha_4}\right\rangle_{G^{(1)}} \\ & + 2\left\langle\sigma_{\alpha_1}\sigma_{\alpha_2}\right\rangle_{G^{(1)}}\left\langle\sigma_{\alpha_3}\sigma_{\alpha_4}\right\rangle_{G^{(1)}}\left\langle\sigma_{\alpha_5}\sigma_{\alpha_6}\right\rangle_{G^{(1)}}\Big]\,. \end{aligned} \tag{4.59}$$

Just as in the middle step of (4.40), here again you've likely noticed that nonzero contributions arise only when all the neural indices coincide. You can further use that same insight to show that $\mathbb{E}\left[\left(\widehat{\Delta G}^{(2)}\right)^m\right] = O\big(1/n_1^{m-1}\big)$.

to the two-point correlator, while the quartic coupling gives the *leading* contribution to the connected four-point correlator. Indeed, we shall encounter various observables whose leading contributions stem solely from the nontrivial neuron-neuron interaction induced by the quartic coupling. By contrast, the correction to the quadratic coupling at finite-width is just a small quantitative effect. Nevertheless, we will compute this subleading correction in §4.5 for completeness.[8]

Non-Gaussian action in action

Having completed the two derivations, before moving on to the next section, let's use this opportunity to get a bit more of a feel for how to compute with a non-Gaussian distribution. Paralleling what we did with the Gaussian action in the last section, let's evaluate the expectation of two activations on the same neuron and four activations, with all four on the same neuron or pairs on separate neurons. The resulting expressions will enable us to obtain the distributions of the preactivations in deeper layers.

In the following, we are just applying the formula (4.46) for the expectation of a general function. These expressions will be valid for any layer $\ell > 1$. First, for two activations on the same neuron, we find

$$\begin{aligned}
&\mathbb{E}\left[\sigma(z_{i_1;\alpha_1})\,\sigma(z_{i_1;\alpha_2})\right] \qquad (4.61)\\
&= \left\langle \sigma_{\alpha_1}\sigma_{\alpha_2}\right\rangle_g + \frac{1}{8}\sum_{\beta_1,\ldots,\beta_4\in\mathcal{D}} v^{(\beta_1\beta_2)(\beta_3\beta_4)}\\
&\qquad\times\Big[\left\langle \sigma_{\alpha_1}\sigma_{\alpha_2}\left(z_{\beta_1}z_{\beta_2}-g_{\beta_1\beta_2}\right)\left(z_{\beta_3}z_{\beta_4}-g_{\beta_3\beta_4}\right)\right\rangle_g\\
&\qquad\quad + 2n\left\langle \sigma_{\alpha_1}\sigma_{\alpha_2}\left(z_{\beta_1}z_{\beta_2}-g_{\beta_1\beta_2}\right)\right\rangle_g g_{\beta_3\beta_4} - 2\left\langle \sigma_{\alpha_1}\sigma_{\alpha_2}\right\rangle_g g_{\beta_1\beta_3}g_{\beta_2\beta_4}\Big] + O\Big(v^2\Big)\,,
\end{aligned}$$

where we assume the reader is by now familiar enough with Gaussian integrals and factorization into separate neurons so as not to include the middle steps. This result highlights that the addition of the quartic coupling v has a nontrivial effect even on the two-point correlator of same-neuron activations. We can similarly compute the expectation of four activations on the same neuron, but we'll need only the leading Gaussian contribution, namely

$$\begin{aligned}
&\mathbb{E}\left[\sigma(z_{i_1;\alpha_1})\,\sigma(z_{i_1;\alpha_2})\,\sigma(z_{i_1;\alpha_3})\,\sigma(z_{i_1;\alpha_4})\right] - \mathbb{E}\left[\sigma(z_{i_1;\alpha_1})\,\sigma(z_{i_1;\alpha_2})\right]\mathbb{E}\left[\sigma(z_{i_1;\alpha_3})\,\sigma(z_{i_1;\alpha_4})\right] \qquad (4.62)\\
&= \left\langle \sigma_{\alpha_1}\sigma_{\alpha_2}\sigma_{\alpha_3}\sigma_{\alpha_4}\right\rangle_g - \left\langle \sigma_{\alpha_1}\sigma_{\alpha_2}\right\rangle_g\left\langle \sigma_{\alpha_3}\sigma_{\alpha_4}\right\rangle_g + O(v)\,,
\end{aligned}$$

where we subtracted off the contribution from the two-point correlators as that's what'll appear in the next section. Finally, the similar expectation of four activations on two

[8] It will also turn out (§5.4) that by fine-tuning the initialization hyperparameters such subleading corrections are suppressed with depth in comparison to non-Gaussian corrections, so in a sense this subleading correction to the quadratic coupling can be doubly ignored.

different pairs of neurons $i_1 \neq i_2$ can be evaluated by the application of the formula (4.46) and neuron factorizations in Gaussian expectations, yielding

$$\begin{aligned}
&\mathbb{E}\left[\sigma(z_{i_1;\alpha_1})\,\sigma(z_{i_1;\alpha_2})\,\sigma(z_{i_2;\alpha_3})\,\sigma(z_{i_2;\alpha_4})\right] - \mathbb{E}\left[\sigma(z_{i_1;\alpha_1})\,\sigma(z_{i_1;\alpha_2})\right]\mathbb{E}\left[\sigma(z_{i_2;\alpha_3})\,\sigma(z_{i_2;\alpha_4})\right] \\
=&\frac{1}{8}\sum_{\beta_1,\ldots,\beta_4\in\mathcal{D}} v^{(\beta_1\beta_2)(\beta_3\beta_4)} \sum_{j_1,j_2=1}^{n} \\
&\times\Big[\left\langle\!\!\left\langle z_{j_1;\beta_1} z_{j_1;\beta_2}\, z_{j_2;\beta_3} z_{j_2;\beta_4} \sigma_{i_1;\alpha_1}\sigma_{i_1;\alpha_2}\sigma_{i_2;\alpha_3}\sigma_{i_2;\alpha_4}\right\rangle\!\!\right\rangle_g \\
&\quad - \left\langle\!\!\left\langle z_{j_1;\beta_1} z_{j_1;\beta_2}\, z_{j_2;\beta_3} z_{j_2;\beta_4} \sigma_{i_1;\alpha_1}\sigma_{i_1;\alpha_2}\right\rangle\!\!\right\rangle_g \left\langle\!\!\left\langle \sigma_{i_2;\alpha_3}\sigma_{i_2;\alpha_4}\right\rangle\!\!\right\rangle_g \\
&\quad - \left\langle\!\!\left\langle z_{j_1;\beta_1} z_{j_1;\beta_2}\, z_{j_2;\beta_3} z_{j_2;\beta_4} \sigma_{i_2;\alpha_3}\sigma_{i_2;\alpha_4}\right\rangle\!\!\right\rangle_g \left\langle\!\!\left\langle \sigma_{i_1;\alpha_1}\sigma_{i_1;\alpha_2}\right\rangle\!\!\right\rangle_g \\
&\quad + \left\langle\!\!\left\langle z_{j_1;\beta_1} z_{j_1;\beta_2}\, z_{j_2;\beta_3} z_{j_2;\beta_4}\right\rangle\!\!\right\rangle_g \left\langle\!\!\left\langle \sigma_{i_1;\alpha_1}\sigma_{i_1;\alpha_2}\right\rangle\!\!\right\rangle_g \left\langle\!\!\left\langle \sigma_{i_2;\alpha_3}\sigma_{i_2;\alpha_4}\right\rangle\!\!\right\rangle_g\Big] \\
=&\frac{1}{4}\sum_{\beta_1,\ldots,\beta_4\in\mathcal{D}} v^{(\beta_1\beta_2)(\beta_3\beta_4)} \left\langle \sigma_{\alpha_1}\sigma_{\alpha_2}\left(z_{\beta_1}z_{\beta_2} - g_{\beta_1\beta_2}\right)\right\rangle_g \left\langle \sigma_{\alpha_3}\sigma_{\alpha_4}\left(z_{\beta_3}z_{\beta_4} - g_{\beta_3\beta_4}\right)\right\rangle_g + O\Big(v^2\Big)\,,
\end{aligned} \tag{4.63}$$

where we get nonzero contributions only when $j_1 = i_1$ and $j_2 = i_2$ or when $j_1 = i_2$ and $j_2 = i_1$. This shows that pairs of activations can only correlate with the addition of the quartic coupling to the action, hinting at the role of finite width for features learning. More generally, consider functions $\mathcal{F}(z_{i_1;\mathcal{A}_1})$ and $\mathcal{G}(z_{i_2;\mathcal{A}_2})$ of preactivations that depend on subsamples $\mathcal{A}_1$ and $\mathcal{A}_2 \subset \mathcal{D}$, respectively, where with a slight abuse of notation we put the set dependences into the subscripts. For distinct neurons $i_1 \neq i_2$, the calculation identical to the one just above shows that their covariance is given by

$$\begin{aligned}
&\mathrm{Cov}\Big[\mathcal{F}(z_{i_1;\mathcal{A}_1}),\, \mathcal{G}(z_{i_2;\mathcal{A}_2})\Big] \\
\equiv&\mathbb{E}\Big[\mathcal{F}(z_{i_1;\mathcal{A}_1})\,\mathcal{G}(z_{i_2;\mathcal{A}_2})\Big] - \mathbb{E}\Big[\mathcal{F}(z_{i_1;\mathcal{A}_1})\Big]\,\mathbb{E}\Big[\mathcal{G}(z_{i_2;\mathcal{A}_2})\Big] \\
=&\frac{1}{4}\sum_{\beta_1,\ldots,\beta_4\in\mathcal{D}} v^{(\beta_1\beta_2)(\beta_3\beta_4)} \Big\langle \left(z_{\beta_1}z_{\beta_2} - g_{\beta_1\beta_2}\right)\mathcal{F}(z_{\mathcal{A}_1})\Big\rangle_g \Big\langle \left(z_{\beta_3}z_{\beta_4} - g_{\beta_3\beta_4}\right)\mathcal{G}(z_{\mathcal{A}_2})\Big\rangle_g + O\Big(v^2\Big)\,.
\end{aligned} \tag{4.64}$$

This formula will be very useful in the future.

4.3 Deeper Layers: Accumulation of Non-Gaussianity

The preactivations in the deeper layers are recursively given by

$$z^{(\ell+1)}_{i;\alpha} = b^{(\ell+1)}_i + \sum_{j=1}^{n_\ell} W^{(\ell+1)}_{ij}\sigma^{(\ell)}_{j;\alpha}\,, \quad \text{for} \quad i = 1,\ldots,n_{\ell+1}\,, \tag{4.65}$$

with the activations in the previous layer abbreviated as

$$\sigma^{(\ell)}_{i;\alpha} \equiv \sigma\Big(z^{(\ell)}_{i;\alpha}\Big)\,. \tag{4.66}$$

We can obtain the marginal distributions of the preactivations in these deeper layers – including the output distribution $p\Big(z^{(L)}\Big|\mathcal{D}\Big)$ – by following the procedure that we implemented for the second-layer distribution. The only complication is that the preactivation distribution in the previous layer is no longer Gaussian, like it was for the first layer.

The three key concepts of the derivation are: recursion, action, and $1/n$-expansion. Let's walk through them one by one.

Recursion

The idea of recursion is to start with information contained in the marginal distribution $p\Big(z^{(\ell)}\Big|\mathcal{D}\Big)$ in the ℓ-th layer and obtain the marginal distribution for the $(\ell+1)$-th layer. The change of the marginal preactivation distribution from layer to layer can be captured by first writing out the joint probability distribution of preactivations in adjacent layers ℓ and $\ell+1$,

$$p\Big(z^{(\ell+1)},z^{(\ell)}\Big|\mathcal{D}\Big)=p\Big(z^{(\ell+1)}\Big|z^{(\ell)}\Big)\,p\Big(z^{(\ell)}\Big|\mathcal{D}\Big)\ , \tag{4.67}$$

then calculating the conditional probability distribution $p\left(z^{(\ell+1)}\Big|z^{(\ell)}\right)$, and finally marginalizing over the preactivations at the ℓ-th layer as

$$p\Big(z^{(\ell+1)}\Big|\mathcal{D}\Big)=\int\left[\prod_{i,\alpha}dz^{(\ell)}_{i;\alpha}\right]p\Big(z^{(\ell+1)}\Big|z^{(\ell)}\Big)\,p\Big(z^{(\ell)}\Big|\mathcal{D}\Big)\ . \tag{4.68}$$

In particular, the conditional probability distribution $p\Big(z^{(\ell+1)}\Big|z^{(\ell)}\Big)$ serves as a **transition matrix**, bridging preactivation distributions in adjacent layers.

The calculation of this conditional distribution proceeds identically to the one we performed for the first layer (4.16) and then repurposed for computing the second-layer conditional distribution (4.33). If you'd like, you can again follow along with §4.1, replacing $z^{(1)}$ by $z^{(\ell+1)}$ and $x_{j;\alpha}$ by $\sigma^{(\ell)}_{j;\alpha}$, and obtain

$$p\Big(z^{(\ell+1)}\Big|z^{(\ell)}\Big)=\frac{1}{\sqrt{\left|2\pi\widehat{G}^{(\ell+1)}\right|^{n_{\ell+1}}}}\exp\left(-\frac{1}{2}\sum_{i=1}^{n_{\ell+1}}\sum_{\alpha_1,\alpha_2\in\mathcal{D}}\widehat{G}^{\alpha_1\alpha_2}_{(\ell+1)}z^{(\ell+1)}_{i;\alpha_1}z^{(\ell+1)}_{i;\alpha_2}\right)\ , \tag{4.69}$$

with the $(\ell+1)$-th-layer stochastic metric

$$\widehat{G}^{(\ell+1)}_{\alpha_1\alpha_2}\equiv C_b^{(\ell+1)}+C_W^{(\ell+1)}\frac{1}{n_\ell}\sum_{j=1}^{n_\ell}\sigma^{(\ell)}_{j;\alpha_1}\sigma^{(\ell)}_{j;\alpha_2}\,, \tag{4.70}$$

depending on the random variables $z^{(\ell)}$ in the previous layer ℓ through the activations $\sigma^{(\ell)}$. Note that all the correlators with odd numbers of the $(\ell+1)$-th-layer preactivations vanish while even-point correlators are obtained through Wick's contractions, yielding

$$\mathbb{E}\left[z^{(\ell+1)}_{i_1;\alpha_1}\cdots z^{(\ell+1)}_{i_{2m};\alpha_{2m}}\right]=\sum_{\text{all pairings}}\delta_{i_{k_1}i_{k_2}}\cdots\delta_{i_{k_{2m-1}}i_{k_{2m}}}\mathbb{E}\left[\widehat{G}^{(\ell+1)}_{\alpha_{k_1}\alpha_{k_2}}\cdots\widehat{G}^{(\ell+1)}_{\alpha_{k_{2m-1}}\alpha_{k_{2m}}}\right], \tag{4.71}$$

where the sum runs over all the $(2m-1)!!$ parings of auxiliary indices $(k_1, \ldots, k_{2m})$. On the left hand, the expectation value characterizes the $(\ell+1)$-th-layer preactivation distribution; on the right hand, the expectation value becomes a correlator of ℓ-th-layer activations upon plugging in the stochastic metric (4.70), which can be evaluated with the ℓ-th-layer distribution.

The mean of the stochastic metric is given by

$$G^{(\ell+1)}_{\alpha_1\alpha_2} \equiv \mathbb{E}\left[\widehat{G}^{(\ell+1)}_{\alpha_1\alpha_2}\right] = C_b^{(\ell+1)} + C_W^{(\ell+1)} \frac{1}{n_\ell} \sum_{j=1}^{n_\ell} \mathbb{E}\left[\sigma^{(\ell)}_{j;\alpha_1} \sigma^{(\ell)}_{j;\alpha_2}\right] , \tag{4.72}$$

and this mean metric governs the two-point correlator in the $(\ell+1)$-th layer through

$$\mathbb{E}\left[z^{(\ell+1)}_{i_1;\alpha_1} z^{(\ell+1)}_{i_2;\alpha_2}\right] = \delta_{i_1 i_2} \mathbb{E}\left[\widehat{G}^{(\ell+1)}_{\alpha_1\alpha_2}\right] = \delta_{i_1 i_2} G^{(\ell+1)}_{\alpha_1\alpha_2} , \tag{4.73}$$

as we saw for the second layer (4.41) as a special case of the equation (4.71). Meanwhile, the fluctuation around the mean

$$\widehat{\Delta G}^{(\ell+1)}_{\alpha_1\alpha_2} \equiv \widehat{G}^{(\ell+1)}_{\alpha_1\alpha_2} - G^{(\ell+1)}_{\alpha_1\alpha_2} = C_W^{(\ell+1)} \frac{1}{n_\ell} \sum_{j=1}^{n_\ell} \left(\sigma^{(\ell)}_{j;\alpha_1} \sigma^{(\ell)}_{j;\alpha_2} - \mathbb{E}\left[\sigma^{(\ell)}_{j;\alpha_1} \sigma^{(\ell)}_{j;\alpha_2}\right] \right) , \tag{4.74}$$

obviously has zero mean,

$$\mathbb{E}\left[\widehat{\Delta G}^{(\ell+1)}_{\alpha_1\alpha_2}\right] = 0 , \tag{4.75}$$

and has a magnitude

$$\frac{1}{n_\ell} V^{(\ell+1)}_{(\alpha_1\alpha_2)(\alpha_3\alpha_4)} \equiv \mathbb{E}\left[\widehat{\Delta G}^{(\ell+1)}_{\alpha_1\alpha_2} \widehat{\Delta G}^{(\ell+1)}_{\alpha_3\alpha_4}\right] = \mathbb{E}\left[\widehat{G}^{(\ell+1)}_{\alpha_1\alpha_2} \widehat{G}^{(\ell+1)}_{\alpha_3\alpha_4}\right] - G^{(\ell+1)}_{\alpha_1\alpha_2} G^{(\ell+1)}_{\alpha_3\alpha_4} . \tag{4.76}$$

Here we have introduced the $(\ell+1)$-th-layer four-point vertex $V^{(\ell+1)}_{(\alpha_1\alpha_2)(\alpha_3\alpha_4)}$, generalizing the second-layer four-point vertex (4.40), which governs the connected four-point correlator in the $(\ell+1)$-th layer. Specifically, following along with the manipulations for the second layer – cf. (4.42) and (4.43) – or simply applying the general expression (4.71), we see

$$\begin{aligned} &\mathbb{E}\left[z^{(\ell+1)}_{i_1;\alpha_1} z^{(\ell+1)}_{i_2;\alpha_2} z^{(\ell+1)}_{i_3;\alpha_3} z^{(\ell+1)}_{i_4;\alpha_4}\right]\Big|_{\text{connected}} \\ =&\frac{1}{n_\ell}\left[\delta_{i_1 i_2}\delta_{i_3 i_4} V^{(\ell+1)}_{(\alpha_1\alpha_2)(\alpha_3\alpha_4)} + \delta_{i_1 i_3}\delta_{i_2 i_4} V^{(\ell+1)}_{(\alpha_1\alpha_3)(\alpha_2\alpha_4)} + \delta_{i_1 i_4}\delta_{i_2 i_3} V^{(\ell+1)}_{(\alpha_1\alpha_4)(\alpha_2\alpha_3)}\right] . \end{aligned} \tag{4.77}$$

In summary, what we have so far are the expressions for the two-point correlator (4.73) and the connected four-point correlator (4.77) of the $(\ell+1)$-th-layer preactivations in terms of the correlators of the ℓ-th-layer activations, and related expressions for higher-point correlators (4.71) if the need arises. The strategy of our recursive approach is to first evaluate these ℓ-th-layer activation correlators given the ℓ-th-layer distribution $p\Big(z^{(\ell)}\Big|\mathcal{D}\Big)$ and from them obtain the $(\ell+1)$-th-layer preactivation correlators. Using these correlators, we can then reconstruct the $(\ell+1)$-th layer marginal distribution $p\Big(z^{(\ell+1)}\Big|\mathcal{D}\Big)$. Both the evaluation of the ℓ-th-layer activation correlators and the reconstruction of the distribution at the $(\ell+1)$-th layer can be efficiently implemented through the use of the action.

Action

The preactivation distribution $p\Big(z^{(\ell)}\Big|\mathcal{D}\Big)$ can be written in terms of an action as

$$p\Big(z^{(\ell)}\Big|\mathcal{D}\Big) = \frac{e^{-S\left(z^{(\ell)}\right)}}{Z(\ell)}\,, \tag{4.78}$$

with the ℓ-th layer partition function given by

$$Z(\ell) \equiv \int \left[\prod_{i,\alpha} dz^{(\ell)}_{i;\alpha}\right] e^{-S\left(z^{(\ell)}\right)}\,, \tag{4.79}$$

and our ansatz for the action given by the following expansion:

$$\begin{aligned} S\Big(z^{(\ell)}\Big) \equiv & \frac{1}{2}\sum_{i=1}^{n_\ell}\sum_{\alpha_1,\alpha_2\in\mathcal{D}} g^{\alpha_1\alpha_2}_{(\ell)} z^{(\ell)}_{i;\alpha_1} z^{(\ell)}_{i;\alpha_2} \\ & - \frac{1}{8}\sum_{i_1,i_2=1}^{n_\ell}\sum_{\alpha_1,\ldots,\alpha_4\in\mathcal{D}} v^{(\alpha_1\alpha_2)(\alpha_3\alpha_4)}_{(\ell)} z^{(\ell)}_{i_1;\alpha_1} z^{(\ell)}_{i_1;\alpha_2} z^{(\ell)}_{i_2;\alpha_3} z^{(\ell)}_{i_2;\alpha_4} + \ldots\,. \end{aligned} \tag{4.80}$$

This ansatz encompasses both the actions we had in §4.1 for the first-layer preactivations – with $g^{\alpha_1\alpha_2}_{(1)} = G^{\alpha_1\alpha_2}_{(1)}$ and $v_{(1)} = 0$ – and for the second-layer preactivations in §4.2 – with the couplings $g_{(2)}$ and $v_{(2)}$ given by (4.49) and (4.50), respectively. In fact, this represents the most general expansion around the Gaussian action, given the symmetries of preactivation correlators (4.71). In particular, only even powers of preactivations show up in the action since we know that correlators with odd numbers of preactivations vanish.

Here, the coefficients $g^{\alpha_1\alpha_2}_{(\ell)}$, $v^{(\alpha_1\alpha_2)(\alpha_3\alpha_4)}_{(\ell)}$, and the implied additional terms in the expansion are **data-dependent couplings** that together govern the interactions of the neural preactivations and are simply related to the correlators of preactivations $z^{(\ell)}$. In particular, in §4.2 we gave two derivations for the relations between quadratic and quartic couplings on the one hand and two-point and four-point correlators on the other hand. The same argument applies for an arbitrary layer ℓ, and so we have

$$g^{\alpha_1\alpha_2}_{(\ell)} = G^{\alpha_1\alpha_2}_{(\ell)} + O(v,\ldots)\,, \tag{4.81}$$

$$v^{(\alpha_1\alpha_2)(\alpha_3\alpha_4)}_{(\ell)} = \frac{1}{n_{\ell-1}} V^{(\alpha_1\alpha_2)(\alpha_3\alpha_4)}_{(\ell)} + O\Big(v^2,\ldots\Big)\,, \tag{4.82}$$

with the understanding that the raised indices of the four-point vertex are shorthand for contraction with the ℓ-th-layer inverse metric

$$V^{(\alpha_1\alpha_2)(\alpha_3\alpha_4)}_{(\ell)} \equiv \sum_{\beta_1,\ldots,\beta_4\in\mathcal{D}} G^{\alpha_1\beta_1}_{(\ell)} G^{\alpha_2\beta_2}_{(\ell)} G^{\alpha_3\beta_3}_{(\ell)} G^{\alpha_4\beta_4}_{(\ell)} V^{(\ell)}_{(\beta_1\beta_2)(\beta_3\beta_4)}\,. \tag{4.83}$$

Note that the higher-order terms $O(...)$ in (4.81) and (4.82) can be neglected self-consistently if and only if the quartic coupling v and higher-order couplings are perturbatively small. This is indeed the case when networks are sufficiently wide, as we will show next.

Large-width expansion

Now we have our work cut out for us. First, note that these mappings, (4.81) and (4.82), between the correlators and couplings already accomplish one task mentioned in our recursive strategy. Namely, when applied to the $(\ell+1)$-th layer, they reconstruct the $(\ell+1)$-th-layer distribution out of the $(\ell+1)$-th-layer preactivation correlators. The only remaining task then is to use the ℓ-th-layer action (4.80) to compute the expectations of the ℓ-th-layer activations $\sigma^{(\ell)}$ that appear in the expressions for the two-point correlator (4.73) and four-point correlator (4.77) of the $(\ell+1)$-th-layer preactivations $z^{(\ell+1)}$.

These calculations simplify in the wide regime with a large number of neurons per layer

$$n_1, n_2, \ldots, n_{L-1} \sim n \gg 1 \,. \tag{4.84}$$

As has been advertised, this large-but-finite-width regime is where networks become both practically usable and theoretically tractable. Specifically, the relations (4.81) and (4.82) between correlators and couplings simplify in this regime and higher-order non-Gaussian corrections can be self-consistently truncated in a series in $1/n$.[9] To be precise, we inductively assume that the mean metric $G^{(\ell)} = O(1)$ and the four-point vertex $V^{(\ell)} = O(1)$ are both of order one at the ℓ-th layer – as was the case for the first and second layers – and show that the same holds true at the $(\ell+1)$-th layer. This inductive assumption in particular implies through (4.81) and (4.82) that the quartic coupling $v_{(\ell)} = O(1/n)$ is perturbatively small at the ℓ-th layer and that the quadratic coupling is given by $g_{(\ell)} = G_{(\ell)} + O(1/n)$. In carrying out this inductive proof, we obtain the recursion relations that govern the change in the preactivation distributions from the ℓ-th layer to the $(\ell+1)$-th layer.

To begin, we see that the two-point correlator in the $(\ell+1)$-th layer (4.73) is given simply in terms of the metric

$$G^{(\ell+1)}_{\alpha_1\alpha_2} = C_b^{(\ell+1)} + C_W^{(\ell+1)} \frac{1}{n_\ell} \sum_{j=1}^{n_\ell} \mathbb{E}\left[\sigma^{(\ell)}_{j;\alpha_1} \sigma^{(\ell)}_{j;\alpha_2}\right] . \tag{4.85}$$

With foresight, we already evaluated this particular two-point correlator of activations (4.61) in the last section. Inserting this result, along with the quadratic coupling $g_{(\ell)} = G_{(\ell)} + O(1/n)$ and quartic coupling $v_{(\ell)} = O(1/n)$, we find

$$G^{(\ell+1)}_{\alpha_1\alpha_2} = C_b^{(\ell+1)} + C_W^{(\ell+1)} \left\langle \sigma_{\alpha_1} \sigma_{\alpha_2} \right\rangle_{G^{(\ell)}} + O\left(\frac{1}{n}\right) , \tag{4.86}$$

which is the leading recursion for the two-point correlator of preactivations.[10] We see that this is self-consistent; any metric $G^{(\ell)}$ that is of order one will give an order-one

[9] In the language of §4.6, such a truncation is preserved under the RG flow.

[10] Note that the difference from the second-layer calculation in §4.2 is just that the expectation in (4.85) is not exactly Gaussian, but has a $1/n$ correction. This highlights the main difference with that section, which is that the distribution in the prior layer is perturbatively non-Gaussian.

metric $G^{(\ell+1)}$ in the next layer as well. The correction is suppressed by $O(1/n)$, which affects only the subleading term in the quadratic coupling $g_{(\ell+1)} = G_{(\ell+1)} + O(1/n)$. Note that, neglecting the subleading $1/n$ correction and replacing G by g, the recursion (4.86) for the two-point correlator can also be thought of as the leading recursion for the quadratic coupling.

Next, let's evaluate the four-point correlator (4.77), which involves computing the magnitude of the metric fluctuation (4.76). Substituting in our general expression for the $(\ell+1)$-th-layer metric fluctuation (4.74), we get

$$\begin{aligned} &\frac{1}{n_\ell} V^{(\ell+1)}_{(\alpha_1\alpha_2)(\alpha_3\alpha_4)} \\ &= \left(\frac{C_W^{(\ell+1)}}{n_\ell}\right)^2 \sum_{j,k=1}^{n_\ell} \left\{ \mathbb{E}\left[\sigma^{(\ell)}_{j;\alpha_1}\sigma^{(\ell)}_{j;\alpha_2}\sigma^{(\ell)}_{k;\alpha_3}\sigma^{(\ell)}_{k;\alpha_4}\right] - \mathbb{E}\left[\sigma^{(\ell)}_{j;\alpha_1}\sigma^{(\ell)}_{j;\alpha_2}\right]\mathbb{E}\left[\sigma^{(\ell)}_{k;\alpha_3}\sigma^{(\ell)}_{k;\alpha_4}\right] \right\} . \end{aligned} \tag{4.87}$$

Here, there are two types of the contributions: from coincident neurons and from separate pairs of neurons. Again, with foresight, we have already evaluated both types of four-point activation correlators in the last section. When all four are coincident $j = k$, substituting in (4.62) we find

$$\begin{aligned} &\mathbb{E}\left[\sigma^{(\ell)}_{j;\alpha_1}\sigma^{(\ell)}_{j;\alpha_2}\sigma^{(\ell)}_{j;\alpha_3}\sigma^{(\ell)}_{j;\alpha_4}\right] - \mathbb{E}\left[\sigma^{(\ell)}_{j;\alpha_1}\sigma^{(\ell)}_{j;\alpha_2}\right]\mathbb{E}\left[\sigma^{(\ell)}_{j;\alpha_3}\sigma^{(\ell)}_{j;\alpha_4}\right] \\ &= \left\langle \sigma_{\alpha_1}\sigma_{\alpha_2}\sigma_{\alpha_3}\sigma_{\alpha_4}\right\rangle_{G^{(\ell)}} - \left\langle \sigma_{\alpha_1}\sigma_{\alpha_2}\right\rangle_{G^{(\ell)}}\left\langle \sigma_{\alpha_3}\sigma_{\alpha_4}\right\rangle_{G^{(\ell)}} + O\left(\frac{1}{n}\right) , \end{aligned} \tag{4.88}$$

where we have truncated to leading order in $1/n$ as a consequence of the inductive assumption at the ℓ-th layer. Meanwhile, when $j \neq k$ and the correlation is between two neurons, we substitute in our expression (4.63), finding

$$\begin{aligned} &\mathbb{E}\left[\sigma^{(\ell)}_{j;\alpha_1}\sigma^{(\ell)}_{j;\alpha_2}\sigma^{(\ell)}_{k;\alpha_3}\sigma^{(\ell)}_{k;\alpha_4}\right] - \mathbb{E}\left[\sigma^{(\ell)}_{j;\alpha_1}\sigma^{(\ell)}_{j;\alpha_2}\right]\mathbb{E}\left[\sigma^{(\ell)}_{k;\alpha_3}\sigma^{(\ell)}_{k;\alpha_4}\right] \\ &= \frac{1}{4n_{\ell-1}} \sum_{\beta_1,\ldots,\beta_4\in\mathcal{D}} V^{(\beta_1\beta_2)(\beta_3\beta_4)}_{(\ell)} \left\langle \sigma_{\alpha_1}\sigma_{\alpha_2}\left(z_{\beta_1}z_{\beta_2} - g_{\beta_1\beta_2}\right)\right\rangle_{G^{(\ell)}} \left\langle \sigma_{\alpha_3}\sigma_{\alpha_4}\left(z_{\beta_3}z_{\beta_4} - g_{\beta_3\beta_4}\right)\right\rangle_{G^{(\ell)}} \\ &\quad + O\left(\frac{1}{n^2}\right) , \end{aligned} \tag{4.89}$$

where again we have truncated to leading order in the large-width expansion using the inductive assumption.[11] Inserting both of these expressions back into (4.87) and

[11] Again, the difference with the second-layer calculation is that in §4.2 these expectations are over the exactly Gaussian first-layer distribution. In that case, there was a contribution of the form (4.88) from the case with all neurons coincident, but *not* of the form (4.89) from the two neurons – cf. (4.40).

performing the sums, we get a recursion for the four-point vertex

$$\begin{aligned}
&\frac{1}{n_\ell} V^{(\ell+1)}_{(\alpha_1\alpha_2)(\alpha_3\alpha_4)} \qquad (4.90)\\
=&\frac{1}{n_\ell}\left(C_W^{(\ell+1)}\right)^2 \left[\langle \sigma_{\alpha_1}\sigma_{\alpha_2}\sigma_{\alpha_3}\sigma_{\alpha_4}\rangle_{G^{(\ell)}} - \langle \sigma_{\alpha_1}\sigma_{\alpha_2}\rangle_{G^{(\ell)}} \langle \sigma_{\alpha_3}\sigma_{\alpha_4}\rangle_{G^{(\ell)}}\right]\\
&+ \frac{1}{n_{\ell-1}} \frac{\left(C_W^{(\ell+1)}\right)^2}{4} \sum_{\beta_1,\ldots,\beta_4\in\mathcal{D}} V^{(\beta_1\beta_2)(\beta_3\beta_4)}_{(\ell)} \langle \sigma_{\alpha_1}\sigma_{\alpha_2}\left(z_{\beta_1}z_{\beta_2} - g_{\beta_1\beta_2}\right)\rangle_{G^{(\ell)}}\\
&\qquad\qquad \times \langle \sigma_{\alpha_3}\sigma_{\alpha_4}\left(z_{\beta_3}z_{\beta_4} - g_{\beta_3\beta_4}\right)\rangle_{G^{(\ell)}} + O\left(\frac{1}{n^2}\right) .
\end{aligned}$$

Importantly, we see that

$$\frac{1}{n_\ell} V^{(\ell+1)} = O\left(\frac{1}{n}\right) , \qquad (4.91)$$

and $V^{(\ell+1)} = O(1)$, thus completing our inductive proof and concluding our derivations of the recursion relations (4.86) and (4.90) for the two-point and four-point correlators. As was the case for the quadratic coupling, if we neglect the subleading $1/n^2$ correction and replace G by g and V by v, the recursion (4.90) for the connected four-point correlator can also be thought of as the recursion for the quartic coupling.

Note that in the strict $n \to \infty$ limit, the quartic coupling vanishes, and the marginal distribution of preactivations $p\left(z^{(\ell)}\middle|\mathcal{D}\right)$ is Gaussian for all layers ℓ. The first nontrivial correction to this infinite-width limit is captured by studying the quartic action with couplings $v_{(\ell)}$. In what follows, we will mostly focus on the effective theory with this quartic action, as we expect significant qualitative differences in the behavior of networks described by the quadratic action vs. the quartic action. The additional finite-width corrections given by the higher-order terms in the action can change quantitative results but should not really exhibit qualitative differences.

4.4 Marginalization Rules

In the past sections, at each step in the recursions we marginalized over all the preactivations in a given layer. This section collects two remarks on other sorts of *partial* marginalizations we can perform, rather than integrating out an *entire* layer. In particular, we'll discuss marginalization over a subset of the $N_\mathcal{D}$ samples in the dataset $\mathcal{D}$ and marginalization over a subset of neurons in a layer.

Loosely speaking, these marginalizations let us focus on specific input data and neurons of interest. Tightly speaking, let's consider evaluating the expectation of a function $F(z_{I;\mathcal{A}}) = F\left(\{z_{i;\alpha}\}_{i\in I;\alpha\in\mathcal{A}}\right)$ that depends on a subsample $\mathcal{A} \subset \mathcal{D}$ and a subset of neurons $I \subset \{1,\ldots,n_\ell\} \equiv \mathcal{N}$ in a layer ℓ, where with a slight abuse of notation we

put the set dependences into the subscripts. We then have

$$\begin{aligned}
&\mathbb{E}\left[F(z_{I;\mathcal{A}})\right] \\
&= \int \left[\prod_{i\in\mathcal{N}}\prod_{\alpha\in\mathcal{D}} dz_{i;\alpha}\right] F(z_{I;\mathcal{A}})\; p\Big(z_{\mathcal{N};\mathcal{D}}\Big|\mathcal{D}\Big) \\
&= \int \left[\prod_{i\in I}\prod_{\alpha\in\mathcal{A}} dz_{i;\alpha}\right] F(z_{I;\mathcal{A}}) \left\{\int \left[\prod_{(j;\beta)\in[\mathcal{N}\times\mathcal{D}-I\times\mathcal{A}]} dz_{j;\beta}\right] p\Big(z_{\mathcal{N};\mathcal{D}}\Big|\mathcal{D}\Big)\right\} \\
&= \int \left[\prod_{i\in I}\prod_{\alpha\in\mathcal{A}} dz_{i;\alpha}\right] F(z_{I;\mathcal{A}})\; p\Big(z_{I;\mathcal{A}}\Big|\mathcal{A}\Big)
\end{aligned} \tag{4.92}$$

where the last equality is just the marginalization over the spectator variables that do not enter into the observable of interest and, in a sense, defines the subsampled and subneuroned distribution as

$$p\Big(z_{I;\mathcal{A}}\Big|\mathcal{A}\Big) \equiv \int \left[\prod_{(j;\beta)\in[\mathcal{N}\times\mathcal{D}-I\times\mathcal{A}]} dz_{j;\beta}\right]\; p\Big(z_{\mathcal{N};\mathcal{D}}\Big|\mathcal{D}\Big)\;. \tag{4.93}$$

In words, in evaluating the expectation of the function $F(z_{I;\mathcal{A}})$, the full distribution $p\Big(z_{\mathcal{N};\mathcal{D}}\Big|\mathcal{D}\Big)$ can simply be restricted to that of the subsample $\mathcal{A}$ and subneurons I, i.e., $p\Big(z_{I;\mathcal{A}}\Big|\mathcal{A}\Big)$. We call this property a **marginalization rule**. Yes, this is somewhat trivial – we're just restating the consistency of probability distributions with respect to marginalization – but it has two rather useful consequences for us.

Marginalization over samples

The first corollary of the marginalization rule is that we can use it to reduce a gigantic integral over all the samples in the dataset to a compact integral over only a handful of samples. For example, in recursively obtaining the two-point correlator through

$$\mathbb{E}\left[z^{(\ell+1)}_{i_1;\alpha_1} z^{(\ell+1)}_{i_2;\alpha_2}\right] = \delta_{i_1 i_2}\left[C_b^{(\ell+1)} + C_W^{(\ell+1)}\left\langle\sigma_{\alpha_1}\sigma_{\alpha_2}\right\rangle_{G^{(\ell)}} + O\Big(\frac{1}{n}\Big)\right], \tag{4.94}$$

we can reduce the $N_{\mathcal{D}}$-dimensional Gaussian integrals $\left\langle\sigma_{\alpha_1}\sigma_{\alpha_2}\right\rangle_{G^{(\ell)}}$ with the $N_{\mathcal{D}}$-by-$N_{\mathcal{D}}$ variance matrix $G^{(\ell)}$ to a manageable two-dimensional integral with a two-by-two submatrix spanned by α_1 and α_2 (or a one-dimensional integral if $\alpha_1 = \alpha_2$). Similarly, a Gaussian integral for four activations $\left\langle\sigma_{\alpha_1}\sigma_{\alpha_2}\sigma_{\alpha_3}\sigma_{\alpha_4}\right\rangle_{G^{(\ell)}}$ that appears in the recursion for four-point vertex involves integrals over four variables *at most*. Generally, in using the action (4.80) to evaluate a specific expectation, the summation over the whole dataset $\mathcal{D}$ in the action can be restricted to the subset of input data that actually appears in the expectation.

Marginalization over neurons

The second corollary involves integrating out a subset of neurons in a layer. Prudent readers might have worried that the quartic term in the ℓ-th-layer action,

$$-\frac{1}{8}\sum_{i_1,i_2=1}^{n_\ell}\sum_{\alpha_1,\ldots,\alpha_4\in\mathcal{D}} v_{(\ell)}^{(\alpha_1\alpha_2)(\alpha_3\alpha_4)} z_{i_1;\alpha_1}^{(\ell)} z_{i_1;\alpha_2}^{(\ell)} z_{i_2;\alpha_3}^{(\ell)} z_{i_2;\alpha_4}^{(\ell)}\,, \tag{4.95}$$

seems to naively scale like $\sim n_\ell^2/n_{\ell-1} = O(n)$, since there are two sums over n_ℓ, and we know from (4.82) that the coupling $v_{(\ell)}$ scales like $\sim 1/n_{\ell-1}$. Similarly, the quadratic term,

$$\frac{1}{2}\sum_{i=1}^{n_\ell}\sum_{\alpha_1,\alpha_2\in\mathcal{D}} g_{(\ell)}^{\alpha_1\alpha_2} z_{i;\alpha_1}^{(\ell)} z_{i;\alpha_2}^{(\ell)}\,, \tag{4.96}$$

has a single sum over n_ℓ and so seems naively $O(n)$ as well. This would imply that the quartic term isn't perturbatively suppressed in comparison to the quadratic term, naively calling our perturbative approach into question.

We first observe that this problem never arises for the final layer $\ell = L$, since the output dimension n_L is never parametrically large: the quadratic term scales as $\sim n_L = O(1)$ while the quartic term scales as $\sim n_L^2/n_{L-1} = O(1/n)$, which is perturbatively suppressed.

This observation, combined with the marginalization rule, points at a resolution to the naive scale-counting problem above for the hidden layers. Indeed, all the expectation we evaluated so far – both preactivation and activation correlators – each individually involves only a few neurons m_ℓ in any given layer ℓ, with $m_\ell \ll n_\ell$. This will always be true; we can't actually correlate an infinite number of neurons at once! Thus, when using the action representation (4.80) of the probability distribution to compute these correlators at the ℓ-th layer, we can first use the marginalization rule (4.92) to integrate out the $(n_\ell - m_\ell)$ spectator neurons that do not participate in the computation, letting us focus on those m_ℓ relevant neurons that actually appear in the expectation. This in turn lets us replace the summations over n_ℓ neurons by ones over the m_ℓ neurons.[12]

All the while, the numbers of neurons in the previous layers $n_1, \ldots, n_{\ell-1}$, having been integrated out to get the action representation at the ℓ-th layer, *are* parametrically large. This means that the quadratic term in the ℓ-th-layer action, reduced to the m_ℓ relevant neurons, scales as $\sim m_\ell = O(1)$, while the quartic term scales as $\sim m_\ell^2/n_{\ell-1} = O(1/n)$. Thus, this ensures a perturbative treatment of the non-Gaussianity.

Running couplings with partial marginalizations

In focusing our attention on only a subset of samples or neurons, the data-dependent couplings of the action need to be adjusted. Since this running of the couplings is

[12] In evaluating generic expectation value such as (4.46), one can always check that the contributions from the $(n_\ell - m_\ell)$ spectator neurons consistently cancel out at each order in $1/n_{\ell-1}$ expansion. If you go back to your personal note that fills in the small gaps between lines in our computations, you will surely notice this cancellation due to Gaussian factorization.

instructive and will be necessary for later computations, let us illustrate here how the quadratic coupling $g^{\alpha_1\alpha_2}_{(\ell),m_\ell}$ depends on the number of neurons m_ℓ in the action.

For simplicity in our illustration, let us specialize to a single input x and drop all the sample indices. Then, denote the distribution over m_ℓ neurons as

$$p\Big(z_1^{(\ell)},\ldots,z_{m_\ell}^{(\ell)}\Big) \propto e^{-S\left(z_1^{(\ell)},\ldots,z_{m_\ell}^{(\ell)}\right)} \tag{4.97}$$
$$= \exp\left[-\frac{g_{(\ell),m_\ell}}{2}\sum_{j=1}^{m_\ell} z_j^{(\ell)} z_j^{(\ell)} + \frac{v_{(\ell)}}{8}\sum_{j_1,j_2=1}^{m_\ell} z_{j_1}^{(\ell)} z_{j_1}^{(\ell)} z_{j_2}^{(\ell)} z_{j_2}^{(\ell)}\right],$$

which is expressed by the same action we've already been using (4.80), though now the dependence of the quadratic coupling on m_ℓ is made explicit.[13] We'll now see in two ways how the quadratic coupling $g_{(\ell),m_\ell}$ *runs* with m_ℓ.

The first way is to begin with the action for n_ℓ neurons and formally integrate out $(n_\ell - m_\ell)$ neurons. Without loss of generality, let's integrate out the *last* $(n_\ell - m_\ell)$ neurons, leaving the *first* m_ℓ neurons labeled as $1,\ldots,m_\ell$. Using the marginalization rule (4.93), we see that

$$e^{-S\left(z_1^{(\ell)},\ldots,z_{m_\ell}^{(\ell)}\right)} \propto p\Big(z_1^{(\ell)},\ldots,z_{m_\ell}^{(\ell)}\Big) = \int dz_{m_\ell+1}^{(\ell)}\cdots dz_{n_\ell}^{(\ell)}\; p\Big(z_1^{(\ell)},\ldots,z_{n_\ell}^{(\ell)}\Big) \tag{4.98}$$
$$\propto \int dz_{m_\ell+1}^{(\ell)}\cdots dz_{n_\ell}^{(\ell)} \exp\left[-\frac{g_{(\ell),n_\ell}}{2}\sum_{i=1}^{n_\ell} z_i^{(\ell)} z_i^{(\ell)} + \frac{v_{(\ell)}}{8}\sum_{i_1,i_2=1}^{n_\ell} z_{i_1}^{(\ell)} z_{i_1}^{(\ell)} z_{i_2}^{(\ell)} z_{i_2}^{(\ell)}\right],$$

throughout which we neglected normalization factors that are irrelevant if we're just interested in the running of the coupling. Next, we can separate out the dependence on the m_ℓ neurons, perturbatively expand the integrand in quartic coupling, and finally integrate out the last $(n_\ell - m_\ell)$ neurons by computing a few simple Gaussian integrals:

$$p\Big(z_1^{(\ell)},\ldots,z_{m_\ell}^{(\ell)}\Big) \tag{4.99}$$
$$\propto \exp\left[-\frac{g_{(\ell),n_\ell}}{2}\sum_{j=1}^{m_\ell} z_j^{(\ell)} z_j^{(\ell)} + \frac{v_{(\ell)}}{8}\sum_{j_1,j_2=1}^{m_\ell} z_{j_1}^{(\ell)} z_{j_1}^{(\ell)} z_{j_2}^{(\ell)} z_{j_2}^{(\ell)}\right]$$
$$\times \int dz_{m_\ell+1}^{(\ell)}\cdots dz_{n_\ell}^{(\ell)} \exp\left[-\frac{g_{(\ell),n_\ell}}{2}\sum_{k=m_\ell+1}^{n_\ell} z_k^{(\ell)} z_k^{(\ell)}\right]$$
$$\times \left[1 + \frac{2v_{(\ell)}}{8}\sum_{j=1}^{m_\ell}\sum_{k=m_\ell+1}^{n_\ell} z_j^{(\ell)} z_j^{(\ell)} z_k^{(\ell)} z_k^{(\ell)} + \frac{v_{(\ell)}}{8}\sum_{k_1,k_2=m_\ell+1}^{n_\ell} z_{k_1}^{(\ell)} z_{k_1}^{(\ell)} z_{k_2}^{(\ell)} z_{k_2}^{(\ell)} + O\Big(v^2\Big)\right]$$
$$= \exp\left[-\frac{g_{(\ell),n_\ell}}{2}\sum_{j=1}^{m_\ell} z_j^{(\ell)} z_j^{(\ell)} + \frac{v_{(\ell)}}{8}\sum_{j_1,j_2=1}^{m_\ell} z_{j_1}^{(\ell)} z_{j_1}^{(\ell)} z_{j_2}^{(\ell)} z_{j_2}^{(\ell)}\right]$$
$$\times \left\{1 + \frac{(n_\ell - m_\ell)}{4}\frac{v_{(\ell)}}{g_{(\ell),n_\ell}}\left(\sum_{i=1}^{m_\ell} z_i^{(\ell)} z_i^{(\ell)}\right) + \frac{v_{(\ell)}}{8g^2_{(\ell),n_\ell}}\Big[(n_\ell - m_\ell)^2 + 2(n_\ell - m_\ell)\Big] + O\Big(v^2\Big)\right\}.$$

[13] Note that in principle the quartic coupling should also depend on m_ℓ: $v_{(\ell)} \to v_{(\ell),m_\ell}$. However, since such a dependence only shows up at higher order in v, we will suppress it.

Finally, resumming the correction arising from the quartic coupling proportional to $\sum_{i=1}^{m_\ell} z_i^{(\ell)} z_i^{(\ell)}$ back into the exponential, ignoring the proportionality factor, and comparing with the action for m_ℓ neurons (4.97), we find

$$g_{(\ell),m_\ell} = g_{(\ell),n_\ell} - \frac{(n_\ell - m_\ell)}{2} \frac{v_{(\ell)}}{g_{(\ell),n_\ell}} + O\Big(v^2\Big) \tag{4.100}$$

as the running equation for the quadratic coupling.

The second way to see the coupling run – and find a solution to the running equation (4.100) – is to compute the single-input metric $G^{(\ell)} \equiv \mathbb{E}\left[z_i^{(\ell)} z_i^{(\ell)}\right]$ and compute it directly using the m_ℓ-neuron action (4.97). We've already computed this in (4.47) using the quartic action for multiple inputs. Specializing to a single input, considering an action of m_ℓ neurons, and being explicit about the dependence of the quadratic coupling on the number of neurons, we get

$$G^{(\ell)} = \left[\frac{1}{g_{(\ell),m_\ell}} + \frac{(m_\ell + 2)}{2} \frac{v^{(\ell)}}{g_{(\ell),m_\ell}^3}\right] + O\Big(v^2\Big) \ . \tag{4.101}$$

Solving this equation for $g_{(\ell),m_\ell}$ by perturbatively expanding in $v^{(\ell)}$, we find

$$\frac{1}{g_{(\ell),m_\ell}} = G^{(\ell)} - \frac{(m_\ell + 2)}{2} \frac{V^{(\ell)}}{n_{\ell-1} G^{(\ell)}} + O\left(\frac{1}{n^2}\right) \ , \tag{4.102}$$

where we have also plugged in

$$v_{(\ell)} = \frac{V^{(\ell)}}{n_{\ell-1} \left(G^{(\ell)}\right)^4} + O\left(\frac{1}{n^2}\right) \ , \tag{4.103}$$

using (4.82) and (4.83) to relate the quartic coupling to the four-point vertex and again specializing to a single input. Now, it's easy to check that this expression (4.102) solves the running equation (4.100).[14]

The key step in this alternative derivation is realizing that observables without any neural indices such as $G^{(\ell)}$ should *not* depend on which version of the m_ℓ action we use in computing them. Interpreted another way, what this running of the coupling means is that for different numbers of neurons in a layer ℓ – e.g. m_ℓ and n_ℓ – we need different quadratic couplings – in this case $g_{(\ell),m_\ell}$ and $g_{(\ell),n_\ell}$ – in order to give the correct value for an ℓ-th-layer observable such as $G^{(\ell)}$. If you're ever in doubt, it's always safest to express an observable of interest in terms of the metric $G^{(\ell)}$ and the four-point vertex $V^{(\ell)}$ rather than the couplings.

[14]Note that the coupling $g_{(\ell),m_\ell}$ depends on m_ℓ – and also on the other hidden-layer widths $n_1, n_2, \ldots, n_{\ell-1}$ – but does *not* depend on the overall width of the current layer n_ℓ. This implies that the quadratic coupling $g_{(\ell),m_\ell}$ is the same coupling we would have used if instead there were actually only m_ℓ neurons in the ℓ-th layer.

4.5 Subleading Corrections

At finite width, all of the correlators receive an infinite series of subleading corrections. Concretely, the metric governing two-point correlator and the four-point vertex governing the connected four-point correlator have $1/n$ series expansions of the form

$$G^{(\ell)}_{\alpha_1\alpha_2} = G^{\{0\}(\ell)}_{\alpha_1\alpha_2} + \frac{1}{n_{\ell-1}} G^{\{1\}(\ell)}_{\alpha_1\alpha_2} + \frac{1}{n^2_{\ell-1}} G^{\{2\}(\ell)}_{\alpha_1\alpha_2} + O\left(\frac{1}{n^3}\right) , \tag{4.104}$$

$$V^{(\ell)}_{(\alpha_1\alpha_2)(\alpha_3\alpha_4)} = V^{\{0\}(\ell)}_{(\alpha_1\alpha_2)(\alpha_3\alpha_4)} + \frac{1}{n_{\ell-1}} V^{\{1\}(\ell)}_{(\alpha_1\alpha_2)(\alpha_3\alpha_4)} + O\left(\frac{1}{n^2}\right) . \tag{4.105}$$

While so far we have focused on the leading contributions $G^{\{0\}(\ell)}_{\alpha_1\alpha_2}$ and $V^{\{0\}(\ell)}_{(\alpha_1\alpha_2)(\alpha_3\alpha_4)}$, the subleading corrections can be systematically calculated as well. Let us illustrate the procedure by deriving the recursion for the next-to-leading-order (NLO) correction to the metric, $G^{\{1\}(\ell)}_{\alpha_1\alpha_2}$.

Before proceeding, let us remark that the leading contribution of the mean metric fully describes the infinite-width limit of the preactivation distributions and so is given a symbol

$$K^{(\ell)}_{\alpha_1\alpha_2} \equiv G^{\{0\}(\ell)}_{\alpha_1\alpha_2} , \tag{4.106}$$

and name, the **kernel**. Since the kernel captures the leading-order correlation between any pair of samples, it will be a central object of study for us in the following chapters. In a similar vein, we will call $G^{\{1\}(\ell)}_{\alpha_1\alpha_2}$ the **NLO metric**.

Our first step will be to express the layer-ℓ quadratic coupling $g^{\beta_1\beta_2}_{(\ell)}$ to order $1/n$ in terms of the $1/n$ correlator data in (4.104) and (4.105). Let's begin by recalling the expression (4.47) for the two-point correlator that we derived from the quartic action, reprinted here for layer ℓ

$$\mathbb{E}\left[z^{(\ell)}_{i_1;\alpha_1} z^{(\ell)}_{i_2;\alpha_2}\right] = \delta_{i_1 i_2} G^{(\ell)}_{\alpha_1\alpha_2} \tag{4.107}$$

$$= \delta_{i_1 i_2}\left[g^{(\ell)}_{\alpha_1\alpha_2} + \frac{1}{2}\sum_{\beta_1,\ldots,\beta_4\in\mathcal{D}} v^{(\beta_1\beta_2)(\beta_3\beta_4)}_{(\ell)}\left(n_\ell\, g^{(\ell)}_{\alpha_1\beta_1} g^{(\ell)}_{\alpha_2\beta_2} g^{(\ell)}_{\beta_3\beta_4} + 2 g^{(\ell)}_{\alpha_1\beta_1} g^{(\ell)}_{\alpha_2\beta_3} g^{(\ell)}_{\beta_2\beta_4}\right)\right] + O\left(v^2\right) .$$

As a reminder $g^{(\ell)}_{\alpha_1\alpha_2}$ is the matrix inverse of the quadratic coupling $g^{\alpha_1\alpha_2}_{(\ell)}$. Substituting in the expansion (4.104) into (4.107), substituting for the quartic coupling $v_{(\ell)} = V_{(\ell)}/n_{\ell-1}$ (4.82), and rearranging to solve for $g^{(\ell)}_{\alpha_1\alpha_2}$ to the subleading order, we get

$$g^{(\ell)}_{\alpha_1\alpha_2} = K^{(\ell)}_{\alpha_1\alpha_2} + \frac{1}{n_{\ell-1}}\left[G^{\{1\}(\ell)}_{\alpha_1\alpha_2} - \sum_{\beta_1,\beta_2\in\mathcal{D}} K^{\beta_1\beta_2}_{(\ell)}\left(\frac{n_\ell}{2} V^{(\ell)}_{(\alpha_1\alpha_2)(\beta_1\beta_2)} + V^{(\ell)}_{(\alpha_1\beta_1)(\alpha_2\beta_2)}\right)\right] + O\left(\frac{1}{n^2}\right) . \tag{4.108}$$

Note that in obtaining the above, we have self-consistently replaced $g^{(\ell)}$ by $K^{(\ell)}$ in the subleading term, which in turn let us lower the indices of the four-point vertices. Inverting this expression (4.108) yields the subleading correction to the quadratic coupling in

terms of the correlators

$$g_{(\ell)}^{\beta_1\beta_2} - K_{(\ell)}^{\beta_1\beta_2} \tag{4.109}$$
$$= \frac{1}{n_{\ell-1}} \sum_{\beta_3,\beta_4 \in \mathcal{D}} \left[-K_{(\ell)}^{\beta_1\beta_3} K_{(\ell)}^{\beta_2\beta_4} G^{\{1\}(\ell)}_{\beta_3\beta_4} + K^{(\ell)}_{\beta_3\beta_4} \left(\frac{n_\ell}{2} V_{(\ell)}^{(\beta_1\beta_2)(\beta_3\beta_4)} + V_{(\ell)}^{(\beta_1\beta_3)(\beta_2\beta_4)} \right) \right] + O\left(\frac{1}{n^2}\right) .$$

Note that one term in this correction scales as $n_\ell/n_{\ell-1}$. As discussed in the previous section, the marginalization rule for the ℓ-th-layer action guarantees that we can treat this quantity as small, $n_\ell/n_{\ell-1} \ll 1$, ensuring that $g^{(\ell)}_{\alpha_1\alpha_2} - K^{(\ell)}_{\alpha_1\alpha_2}$ is a subleading-in-$1/n$ correction to the quadratic coupling. In line with this statement, we'll soon see the cancellation for the factor of n_ℓ when computing the recursion for this subleading correction to the metric $G^{\{1\}(\ell)}$.

Having finished working out the $1/n$-corrected ℓ-th-layer action, we turn to computing the $(\ell+1)$-th-layer two-point correlator.[15] This will let us express the $(\ell+1)$-th-layer two-point correlator in terms of the ℓ-th-layer statistics, ultimately yielding a recursion for $G^{\{1\}(\ell)}_{\alpha_1\alpha_2}$. Starting with the expansion (4.104) in the $(\ell+1)$-th layer and substituting in the expression (4.85) for the two-point correlator, we obtain

$$K^{(\ell+1)}_{\alpha_1\alpha_2} + \frac{1}{n_\ell} G^{\{1\}(\ell+1)}_{\alpha_1\alpha_2} + O\left(\frac{1}{n^2}\right) = G^{(\ell+1)}_{\alpha_1\alpha_2} = C_b^{(\ell+1)} + C_W^{(\ell+1)} \frac{1}{n_\ell} \sum_{j=1}^{n_\ell} \mathbb{E}\left[\sigma^{(\ell)}_{j;\alpha_1} \sigma^{(\ell)}_{j;\alpha_2} \right] . \tag{4.110}$$

Thus, we need the expectation of two activations in the ℓ-th layer up to the order $O(1/n)$, which we evaluated before in expression (4.61) in terms of the ℓ-th-layer couplings.

Looking at (4.61), there are two types of contributions at the subleading order, one arising from the $1/n$ correction to the quadratic coupling $g_{(\ell)}$ in (4.108) and the other from the non-Gaussianty of the distribution due to the quartic coupling $v_{(\ell)}$. The latter contribution is easy to handle: since the quartic coupling is already suppressed by $1/n$, we can just make the replacement $g^{(\ell)} \to K^{(\ell)}$ in the second term in (4.61), yielding

$$\frac{1}{8n_{\ell-1}} \sum_{\beta_1,\ldots,\beta_4 \in \mathcal{D}} V_{(\ell)}^{(\beta_1\beta_2)(\beta_3\beta_4)} \tag{4.111}$$
$$\times \Big[\left\langle \sigma_{\alpha_1}\sigma_{\alpha_2} \left(z_{\beta_1} z_{\beta_2} - K^{(\ell)}_{\beta_1\beta_2} \right) \left(z_{\beta_3} z_{\beta_4} - K^{(\ell)}_{\beta_3\beta_4} \right) \right\rangle_{K^{(\ell)}}$$
$$+ 2n_\ell \left\langle \sigma_{\alpha_1}\sigma_{\alpha_2} \left(z_{\beta_1} z_{\beta_2} - K^{(\ell)}_{\beta_1\beta_2} \right) \right\rangle_{K^{(\ell)}} K^{(\ell)}_{\beta_3\beta_4} - 2 \left\langle \sigma_{\alpha_1}\sigma_{\alpha_2} \right\rangle_{K^{(\ell)}} K^{(\ell)}_{\beta_1\beta_3} K^{(\ell)}_{\beta_2\beta_4} \Big] + O\left(\frac{1}{n^2}\right) .$$

However, for the former contribution, the Gaussian term $\langle \sigma_{\alpha_1}\sigma_{\alpha_2} \rangle_{g^{(\ell)}}$ needs be carefully separated into the leading and subleading pieces. To that end, we can trade the Gaussian

[15] We already knew the $1/n$ contribution to the quartic coupling, namely the relation $v_{(\ell)} = V_{(\ell)}/n_{\ell-1}$.

expectation with $g^{(\ell)}$ for one in terms of the leading kernel $K^{(\ell)}$

$$\begin{aligned}
&\left\langle \sigma_{\alpha_1}\sigma_{\alpha_2} \right\rangle_{g^{(\ell)}} \qquad (4.112)\\
=&\frac{\left\langle \sigma_{\alpha_1}\sigma_{\alpha_2} \exp\left[-\frac{1}{2}\sum_{\beta_1,\beta_2}\left(g_{(\ell)}^{\beta_1\beta_2}-K_{(\ell)}^{\beta_1\beta_2}\right)z_{\beta_1}z_{\beta_2}\right]\right\rangle_{K^{(\ell)}}}{\left\langle \exp\left[-\frac{1}{2}\sum_{\beta_1,\beta_2}\left(g_{(\ell)}^{\beta_1\beta_2}-K_{(\ell)}^{\beta_1\beta_2}\right)z_{\beta_1}z_{\beta_2}\right]\right\rangle_{K^{(\ell)}}}\\
=&\left\langle \sigma_{\alpha_1}\sigma_{\alpha_2} \right\rangle_{K^{(\ell)}} - \frac{1}{2}\sum_{\beta_1,\beta_2}\left(g_{(\ell)}^{\beta_1\beta_2}-K_{(\ell)}^{\beta_1\beta_2}\right)\left\langle \sigma_{\alpha_1}\sigma_{\alpha_2}\left(z_{\beta_1}z_{\beta_2}-K^{(\ell)}_{\beta_1\beta_2}\right)\right\rangle_{K^{(\ell)}} + O\left(\frac{1}{n^2}\right).
\end{aligned}$$

Plugging (4.109) into (4.112), we obtain the subleading contribution due to the change in the quadratic coupling, giving

$$\begin{aligned}
&\left\langle \sigma_{\alpha_1}\sigma_{\alpha_2} \right\rangle_{g^{(\ell)}} \qquad (4.113)\\
=&\left\langle \sigma_{\alpha_1}\sigma_{\alpha_2} \right\rangle_{K^{(\ell)}} + \frac{1}{2n_{\ell-1}}K_{(\ell)}^{\beta_1\beta_3}K_{(\ell)}^{\beta_2\beta_4}G^{\{1\}(\ell)}_{\beta_3\beta_4}\left\langle \sigma_{\alpha_1}\sigma_{\alpha_2}\left(z_{\beta_1}z_{\beta_2}-K^{(\ell)}_{\beta_1\beta_2}\right)\right\rangle_{K^{(\ell)}}\\
&-\frac{1}{n_{\ell-1}}\sum_{\beta_1,\ldots,\beta_4}\left(\frac{n_\ell}{4}V_{(\ell)}^{(\beta_1\beta_2)(\beta_3\beta_4)}+\frac{1}{2}V_{(\ell)}^{(\beta_1\beta_3)(\beta_2\beta_4)}\right)K^{(\ell)}_{\beta_3\beta_4}\left\langle \sigma_{\alpha_1}\sigma_{\alpha_2}\left(z_{\beta_1}z_{\beta_2}-K^{(\ell)}_{\beta_1\beta_2}\right)\right\rangle_{K^{(\ell)}}\\
&+O\left(\frac{1}{n^2}\right).
\end{aligned}$$

Now that we've computed everything, we can add the two contributions to $\mathbb{E}\left[\sigma^{(\ell)}_{j;\alpha_1}\sigma^{(\ell)}_{j;\alpha_2}\right]$, (4.111) and (4.113), and plug them into the expression for the preactivation correlator (4.110). Collecting terms, we recover the leading contribution, the recursion for the kernel

$$K^{(\ell+1)}_{\alpha_1\alpha_2} = C_b^{(\ell+1)} + C_W^{(\ell+1)}\left\langle \sigma_{\alpha_1}\sigma_{\alpha_2} \right\rangle_{K^{(\ell)}}, \qquad (4.114)$$

and also find a recursion for the NLO metric as promised

$$\begin{aligned}
&\frac{1}{n_\ell}G^{\{1\}(\ell+1)}_{\alpha_1\alpha_2} \qquad (4.115)\\
=&C_W^{(\ell+1)}\frac{1}{n_{\ell-1}}\sum_{\beta_1,\ldots,\beta_4\in\mathcal{D}}\left[\frac{1}{2}K_{(\ell)}^{\beta_1\beta_3}K_{(\ell)}^{\beta_2\beta_4}G^{\{1\}(\ell)}_{\beta_3\beta_4}\left\langle \sigma_{\alpha_1}\sigma_{\alpha_2}\left(z_{\beta_1}z_{\beta_2}-K^{(\ell)}_{\beta_1\beta_2}\right)\right\rangle_{K^{(\ell)}}\right.\\
&\qquad+\frac{1}{8}V_{(\ell)}^{(\beta_1\beta_2)(\beta_3\beta_4)}\left\langle \sigma_{\alpha_1}\sigma_{\alpha_2}\left(z_{\beta_1}z_{\beta_2}-K^{(\ell)}_{\beta_1\beta_2}\right)\left(z_{\beta_3}z_{\beta_4}-K^{(\ell)}_{\beta_3\beta_4}\right)\right\rangle_{K^{(\ell)}}\\
&\qquad\left.+\frac{1}{4}V_{(\ell)}^{(\beta_1\beta_3)(\beta_2\beta_4)}K^{(\ell)}_{\beta_3\beta_4}\left\langle \sigma_{\alpha_1}\sigma_{\alpha_2}\left(-2z_{\beta_1}z_{\beta_2}+K^{(\ell)}_{\beta_1\beta_2}\right)\right\rangle_{K^{(\ell)}}\right].
\end{aligned}$$

In going through this calculation in your personal notes or on the margins of this book, you can explicitly see the cancellation of contributions from $n_\ell - 1$ spectator neurons that does not participate in the expectation $\mathbb{E}\left[\sigma^{(\ell)}_{j;\alpha_1}\sigma^{(\ell)}_{j;\alpha_2}\right]$ as required by the marginalization rule for the ℓ-th-layer action. Indeed, every term in the square bracket on the right-hand side of the equation (4.115) is manifestly of order one.

This process can be systematically pushed to higher orders. Just as the computation of the NLO metric $G^{\{1\}(\ell)}_{\alpha_1\alpha_2}$ involved the leading quartic coupling, the computation of the subleading correction to the four-point vertex, $V^{\{1\}(\ell)}_{(\alpha_1\alpha_2)(\alpha_3\alpha_4)}$, and the computation of the order $1/n^2$ correction to the two-point correlator, $G^{\{2\}(\ell)}_{\alpha_1\alpha_2}$, would involve the leading sextic coupling. Such a sextic coupling appears at order $1/n^2$ in the action and contributes to the connected six-point function, which also vanishes as $O(1/n^2)$.[16]

4.6 RG Flow and RG Flow

Since the past five sections have been a whirlwind of equations, algebra, and integration, let's take a moment to recap and assemble the main results.

The goal of this chapter was to find the marginal distribution of preactivations $p\left(z^{(\ell)}\middle|\mathcal{D}\right)$ in a given layer ℓ in terms of an **effective action** with data-dependent couplings. These couplings change – or **run** – from layer to layer, and the running is determined via recursions, which in turn determine how the distribution of preactivations changes with depth. Equivalently, these recursions tell us how correlation functions of preactivations evolve with layer. In this language, starting with independent neurons in the first layer (§4.1), we saw how interactions among neurons are induced in the second layer (§4.2) and then amplified in deeper layers (§4.3).

Concretely, let's summarize the behavior of finite-width networks to leading order in the wide-network expansion. Expressing the two-point correlator of preactivations in terms of the **kernel** $K^{(\ell)}_{\alpha_1\alpha_2}$ as

$$\mathbb{E}\left[z^{(\ell)}_{i_1;\alpha_1} z^{(\ell)}_{i_2;\alpha_2}\right] = \delta_{i_1 i_2} G^{(\ell)}_{\alpha_1\alpha_2} = \delta_{i_1 i_2}\left[K^{(\ell)}_{\alpha_1\alpha_2} + O\left(\frac{1}{n}\right)\right], \tag{4.116}$$

and expressing the four-point connected correlator in terms of the **four-point vertex** $V^{(\ell)}_{(\alpha_1\alpha_2)(\alpha_3\alpha_4)}$ as

$$\begin{aligned}&\mathbb{E}\left[z^{(\ell)}_{i_1;\alpha_1} z^{(\ell)}_{i_2;\alpha_2} z^{(\ell)}_{i_3;\alpha_3} z^{(\ell)}_{i_4;\alpha_4}\right]\Big|_{\text{connected}} \\ =&\frac{1}{n_{\ell-1}}\left[\delta_{i_1 i_2}\delta_{i_3 i_4} V^{(\ell)}_{(\alpha_1\alpha_2)(\alpha_3\alpha_4)} + \delta_{i_1 i_3}\delta_{i_2 i_4} V^{(\ell)}_{(\alpha_1\alpha_3)(\alpha_2\alpha_4)} + \delta_{i_1 i_4}\delta_{i_2 i_3} V^{(\ell)}_{(\alpha_1\alpha_4)(\alpha_2\alpha_3)}\right],\end{aligned} \tag{4.117}$$

[16] For those familiar with field theory, the leading part of the couplings in the action are *tree-level* contributions to correlators. They are to be contrasted with subleading corrections to the two-point correlator discussed in this section, which included both *loop-level* contributions from quartic interaction and tree-level contributions from the NLO correction to the bare quadratic coupling.

the running of these correlators is given by the recursions

$$K_{\alpha_1\alpha_2}^{(\ell+1)} = C_b^{(\ell+1)} + C_W^{(\ell+1)} \left\langle \sigma_{\alpha_1}\sigma_{\alpha_2} \right\rangle_{K^{(\ell)}} , \tag{4.118}$$

$$\begin{aligned} V_{(\alpha_1\alpha_2)(\alpha_3\alpha_4)}^{(\ell+1)} = & \left(C_W^{(\ell+1)}\right)^2 \left[\left\langle \sigma_{\alpha_1}\sigma_{\alpha_2}\sigma_{\alpha_3}\sigma_{\alpha_4} \right\rangle_{K^{(\ell)}} - \left\langle \sigma_{\alpha_1}\sigma_{\alpha_2} \right\rangle_{K^{(\ell)}} \left\langle \sigma_{\alpha_3}\sigma_{\alpha_4} \right\rangle_{K^{(\ell)}} \right] \\ & + \frac{1}{4}\left(C_W^{(\ell+1)}\right)^2 \frac{n_\ell}{n_{\ell-1}} \sum_{\beta_1,\ldots,\beta_4\in\mathcal{D}} V_{(\ell)}^{(\beta_1\beta_2)(\beta_3\beta_4)} \left\langle \sigma_{\alpha_1}\sigma_{\alpha_2}\left(z_{\beta_1}z_{\beta_2} - K_{\beta_1\beta_2}^{(\ell)}\right) \right\rangle_{K^{(\ell)}} \\ & \times \left\langle \sigma_{\alpha_3}\sigma_{\alpha_4}\left(z_{\beta_3}z_{\beta_4} - K_{\beta_3\beta_4}^{(\ell)}\right) \right\rangle_{K^{(\ell)}} + O\left(\frac{1}{n^2}\right) , \end{aligned} \tag{4.119}$$

where the indices on the four-point vertex are raised by the inverse metric $G_{(\ell)}$

$$\begin{aligned} V_{(\ell)}^{(\alpha_1\alpha_2)(\alpha_3\alpha_4)} &\equiv \sum_{\beta_1,\ldots,\beta_4\in\mathcal{D}} G_{(\ell)}^{\alpha_1\beta_1} G_{(\ell)}^{\alpha_2\beta_2} G_{(\ell)}^{\alpha_3\beta_3} G_{(\ell)}^{\alpha_4\beta_4} V_{(\beta_1\beta_2)(\beta_3\beta_4)}^{(\ell)} \\ &= \sum_{\beta_1,\ldots,\beta_4\in\mathcal{D}} K_{(\ell)}^{\alpha_1\beta_1} K_{(\ell)}^{\alpha_2\beta_2} K_{(\ell)}^{\alpha_3\beta_3} K_{(\ell)}^{\alpha_4\beta_4} V_{(\beta_1\beta_2)(\beta_3\beta_4)}^{(\ell)} + O\left(\frac{1}{n}\right) . \end{aligned} \tag{4.120}$$

These recursions dictate how the statistics of preactivations flow with depth.

This flow is very reminiscent of the following heuristic picture, which is offered as an explanation for how neural networks are supposed to work: given an input, such as the image of a cat, the first few layers identify low-level features from the pixels – such as the `edges` between areas of low and high intensity – and then the middle layers assemble these low-level features into mid-level features – such as the texture and pattern of `fur` – which are further aggregated in deeper layers into higher-level representations – such as `tails` and `ears` – which the last layer combines into an estimate of the probability that original pixels represents a `cat`. Indeed, some studies support this hierarchically-ordered arrangement of feature representation in trained networks [32].[17] The desirability of such an arrangement emphasizes both the role and importance of depth in deep learning.

Some of the terms we used in discussing this heuristic picture can actually be given more precise definitions. For instance, each neuron in the network – including not only those in the output layer but also those in the hidden layers – is a scalar function of the input and called a **feature**. The neurons of a given layer can be organized into a vector-valued function of the input, which we'll refer to as a **representation**.[18] In terms of these concepts, our formalism tracks the transformation of representations from one

[17] It has been suggested that even untrained networks have features that can act as types of filters, effectively allowing for primitive edge detecting in untrained networks. For a related set of ideas, see [33].

[18] While the main focus of our study of supervised learning (§7) will be understanding how the representation $z^{(L)}$ in the output layer is learned via gradient-based training, it is also be important to understand how representations are learned in hidden layers (§11). In addition to being necessary components of determining the coarse-grained representation at the output, in some applications of deep learning learned representations in the hidden layers can be used as inputs themselves for other learning tasks. This occurs quite often in unsupervised learning, for example with the word embeddings of natural language processing tasks. In these scenarios, the embeddings – representations for an input word in the larger context of a full sentence – typically are taken not just from the final layer, but from the concatenation of the final few layers. See e.g. [34].

layer to the next. It is this flow of representations that we term **representation group flow** or **RG flow** for short.[19] RG flow is induced via the repeated marginalization of fine-grained features in the shallow layers to give a coarse-grained representation in the output layer. Our notion of RG flow makes the heuristic picture given above concrete.

This pattern of coarse-graining has a parallel in theoretical physics, known as **renormalization group flow** or **RG flow** for short. In this case, the RG flow is generated by the repeated marginalization of the microscopic fine-grained degrees of freedom in the system in order to obtain an *effective theory* of the system in terms of macroscopic coarse-grained variables. Analogously, the physical couplings controlling the interactions of these effective degrees of freedom *run* with the length scale at which they are probed – e.g. the effective charge of the electron will change when interrogated at different scales. Similar to the recursion equations describing the running couplings of the network representations, one can derive differential equations – historically called beta functions – that govern the running of the physical couplings with scale.[20]

To make the connection between this RG flow and that RG flow abundantly clear, let's peek into how it is implemented in field theory in physics. In this scenario, the degrees of freedom are represented by a field $\phi(x)$ that may take different values as a function of spacetime coordinate x. First, one divides $\phi(x)$ into fine-grained variables ϕ^+ consisting of high-frequency modes and coarse-grained variables ϕ^- consisting of low-frequency modes, such that the field decomposes as $\phi(x) = \phi^+(x) + \phi^-(x)$. The full

[19] Two apologia are in order for the name *representation group flow*: *(i)* it is confusingly close to the notion of *group representation theory* in mathematics; and *(ii)* the flow is technically a *semigroup*, rather than a group. (Both group theory and semigroup theory are the studies of transformations but a group requires inverses while a semigroup does not; and the flow has no inverse.) This is just to repeat a historic mistake in physics as will explain further in a footnote below.

[20] *A brief history of renormalization in physics.*

Renormalization was originally developed in the 1930s and 1940s to deal with divergences – infinities – that plagued the calculations of experimental observables in quantum field theory. At first, these infinities were simply subtracted off – swept under the rug, if you will – yielding answers that, despite these shenanigans, matched extremely well with experiments. This whole state of affairs was considered embarrassing, leading to near abandonment of the theory.

These divergences arose essentially due to a failure to properly take into account that couplings can be scale-dependent. The idea of running couplings was first put forth by Gell-Mann and Low [35] in 1954, however a full conceptualization of renormalization wasn't available until Wilson developed the modern notion of RG flow [36, 37] in 1971, offering a theoretical explanation for critical phenomena in statistical physics as well as giving a sound grounding for the understanding of divergences in quantum field theory.

At this point, all historical accounts of RG are contractually obligated to mention the following: the renormalization group is not a group; it's a semigroup. (The mistake was made in an early paper by Stueckelberg and Petermann, referring to the flow as a "group of normalization" [38].) Mathematically, this is because there are no inverse elements; the marginalization of variables out of a joint distribution deletes information and cannot be undone. In particular, two different joint distributions can sometimes flow to the same distribution after marginalization. Intuitively, this is because these flows go from fine-grained descriptions to coarse-grained descriptions. (Such convergent flows lead to the notion of *universality*, which we will explain in §5 in the context of neural networks with different activations that flow to the same marginal distributions under RG.)

Clearly, RG flow in physics is a very rich subject. If you're interested in learning more, we recommend both [39, 40].

distribution is governed by the full action

$$S_{\text{full}}(\phi) = S(\phi^+) + S(\phi^-) + S_{\text{I}}(\phi^+, \phi^-)\,, \tag{4.121}$$

where in particular the last term describes the interactions between these two sets of modes.

Now, if all we care about are observables that depend only on the coarse-grained modes ϕ^- at macroscopic scales – and such long-range scales are usually the relevant ones for experiments – then this full description is too cumbersome to usefully describe the outcome of such experiments. In order to obtain an effective description in terms of only these coarse-grained variable ϕ^-, we can integrate out (i.e. marginalizes over) the fine-grained variables ϕ^+ as

$$e^{-S_{\text{eff}}(\phi^-)} = \int d\phi^+ \; e^{-S_{\text{full}}(\phi)}\,, \tag{4.122}$$

and obtain an **effective action** $S_{\text{eff}}(\phi^-)$, providing an effective theory for the observables of experimental interest. In practice, this marginalization is carried out scale by scale, dividing up the field as $\phi = \phi^{(1)} + \ldots + \phi^{(L)}$ from microscopic modes $\phi^{(1)}$ all the way to macroscopic modes $\phi^{(L)} = \phi^-$, and then integrating out the variables $\phi^{(1)}$, …, $\phi^{(L-1)}$ in sequence. Tracking the flow of couplings in the effective action through this marginalization results in the aforementioned beta functions, and in solving these differential equations up to the scale of interest, we get an effective description of observables at that scale.

This is precisely what we have been doing in this chapter for neural networks. The full field ϕ is analogous to a collection of all the preactivations $\left\{z^{(1)}, \ldots, z^{(L)}\right\}$. Their distribution is governed by the full joint distribution of preactivations

$$p\left(z^{(1)}, \ldots, z^{(L)} \middle| \mathcal{D}\right) = p\left(z^{(L)} \middle| z^{(L-1)}\right) \cdots p\left(z^{(2)} \middle| z^{(1)}\right) p\left(z^{(1)} \middle| \mathcal{D}\right)\,, \tag{4.123}$$

with the full action

$$S_{\text{full}}\left(z^{(1)}, \ldots, z^{(L)}\right) \equiv \sum_{\ell=1}^{L} S_{\text{M}}\left(z^{(\ell)}\right) + \sum_{\ell=1}^{L-1} S_{\text{I}}\left(z^{(\ell+1)} \middle| z^{(\ell)}\right)\,. \tag{4.124}$$

Here, the full action is decomposed into the mean quadratic action for variables $z^{(\ell)}$

$$S_{\text{M}}\left(z^{(\ell)}\right) = \frac{1}{2} \sum_{i=1}^{n_\ell} \sum_{\alpha_1, \alpha_2 \in \mathcal{D}} G^{\alpha_1 \alpha_2}_{(\ell)} z^{(\ell)}_{i;\alpha_1} z^{(\ell)}_{i;\alpha_2} \tag{4.125}$$

in terms of the mean metric $G^{(\ell)}$, (4.72), and the interaction between neighboring layers

$$S_{\text{I}}\left(z^{(\ell+1)} \middle| z^{(\ell)}\right) = \frac{1}{2} \sum_{i=1}^{n_{\ell+1}} \sum_{\alpha_1, \alpha_2 \in \mathcal{D}} \left[\widehat{G}^{\alpha_1 \alpha_2}_{(\ell+1)}\left(z^{(\ell)}\right) - G^{\alpha_1 \alpha_2}_{(\ell+1)}\right] z^{(\ell+1)}_{i;\alpha_1} z^{(\ell+1)}_{i;\alpha_2}\,. \tag{4.126}$$

Here we emphasized that the stochastic metric $\widehat{G}^{(\ell+1)}_{\alpha_1\alpha_2}$ is a function of $z^{(\ell)}$, and the induced coupling of $z^{(\ell)}$ with $z^{(\ell+1)}$ is what leads to the interlayer interactions.

Now, if all we care about are observables that depend only on the outputs of the network – which includes a very important observable . . . the output! – then this full description is too cumbersome. In order to obtain an effective (i.e. useful) description of the distribution of outputs $z^{(L)}$, we can marginalizes over all the features $\left\{z^{(1)}, \ldots, z^{(L-1)}\right\}$ as

$$e^{-S_{\text{eff}}\left(z^{(L)}\right)} = \int \left[\prod_{\ell=1}^{L-1} dz^{(\ell)}\right] e^{-S_{\text{full}}\left(z^{(1)},\ldots,z^{(L)}\right)}\,, \tag{4.127}$$

just as we integrated out the fine-grained modes ϕ^+ in (4.122) to get the effective description in terms of coarse-grained modes ϕ^-. And, just like in the field theory example, rather than carrying out this marginalization all at once, we proceeded sequentially, integrating out the preactivations layer by layer. This resulted in the recursion relations (4.118) and (4.119), and in solving these recursion relations up to the depth of interest, we get an effective description of neural network output at that depth.[21]

Now, this last sentence suggests a subtle but interesting shift of perspective, so let us elaborate. So far in this chapter, we have implicitly assumed a fixed network depth L and described how the preactivation distribution changes as an input x propagates through the intermediate layers, yielding recursion relations for correlators and couplings for the evolution from layer ℓ to layer $\ell+1$, for $\ell = 0, \ldots, L-1$. However, it is also valid to view the resulting recursion equations as governing the change in output distributions as the overall network depth changes from L to $L+1$.[22] In other words, these recursion relations describe the effect of adding an additional layer to the neural network by comparing distributions $p\left(z^{(L)}\middle|\mathcal{D}\right)$ and $p\left(z^{(L+1)}\middle|\mathcal{D}\right)$.

Given this perspective, our RG flow can address head-on the effect of the *deep* in *deep learning*. For instance, as a network get deeper, do the interactions between neurons – encoded in the finite-width corrections such as the four-point vertex $V^{(\ell)}$ – get amplified or attenuated? In the language of RG flow, couplings that grow with the flow are called **relevant** and those that shrink are called **irrelevant**.[23] These names are evocative of whether the interaction matters or not for the effective theory, and so we'll employ the same terminology. Thus, to explore the effect of depth on the neuron-neuron interactions, we are simply asking whether the four-point vertex $V^{(\ell)}$ is relevant or irrelevant.

This question has important implications for deep learning. If all the finite-width couplings were irrelevant, then finite-width networks would asymptote to infinite-width architectures under RG flow. This would then mean that these networks behave more like infinite-width models as they get deeper, and so deep learning would really be the study of these much simpler Gaussian models. Fortunately we'll soon find that

[21] Note to physicists: the flow in networks from input to output is a flow from the ultraviolet to the infrared.

[22] To be precise, the output dimension n_{out} is fixed. So, as the depth changes from L to $L+1$, we imagine holding fixed the widths for $\ell < L$, inserting a new layer L with $n_L \sim n \gg 1$, and then setting the final layer $L+1$ to have width n_{out}.

[23] Couplings that neither grow nor shrink are called **marginal**.

the couplings *are* relevant, making our life richer, albeit more complicated. In the next chapter, we'll show that finite networks deviate more and more from their infinite-width counterparts as they get deeper. This has important practical consequences in controlling the instantiation-to-instantiation fluctuations in supervised training (§10) as well as allowing networks to learn nontrivial representations of their input (§11).

The next chapter explores these relevant questions by explicitly solving recursion equations such as (4.118) and (4.119).

Chapter 5

Effective Theory of Preactivations at Initialization

> *We believe this realm of work to be immensely important and rich, but we expect its growth to require a degree of critical analysis that its more romantic advocates have always been reluctant to pursue ....*
>
> Minsky and Papert in the prologue to their 1988 expanded edition of *Perceptrons* [41].

The key defining feature of deep learning is the stacking of components on top of each other in order to get a *deep* neural network architecture. Despite the empirical preference for deeper networks, it's not at all obvious *why* deep is good for learning. For a fixed number of neurons per layer, deep implies many more parameters, and often in deep learning more parameters lead to better performance. But there are other ways to include more parameters. For instance, why not just have a single hidden layer that is very wide? In fact, in the strict infinite-width limit, such a single-layer model has the same number of parameters as any deeper MLP: infinity.

The proper way to think about the effects of depth is not to just count the number of model parameters, but instead to ask what happens when we add an additional layer to our MLP. In §4, we developed a formalism to address exactly this question through recursions for observable quantities of interest, enabling us to compute how the distributions of initial network outputs change upon adding a layer. What we need, then, is a tool to effectively extract the explicit depth dependence from these recursions.

Building on the effective theory formalism developed in §4, in this chapter we'll extend the criticality and fluctuation analyses performed in §3 to MLPs with any nonlinear activation function. Enlightened by the success of the previous chapter in finding simplification in the wide regime ($n \gg 1$), we now seek additional simplicity in the limit of large depth ($L \gg 1$).[1] We'll first analyze the limit of infinite number of neurons per layer, and then back off this limit to consider networks of large but finite width and

[1] What this means – in light of the discussion in §3.4 – is that we take the limit of large width *first* and then look at the limit of large depth.

depth. The result will be explicit expressions for the two-point and four-point correlators of preactivations in these asymptotic limits.[2]

This will let us address the question of what happens to input signals as they propagate through the many layers of deep neural networks at initialization (§5.1). We'll come to understand that the order-one values of the initialization hyperparameters – i.e. the bias variances $C_b^{(\ell)}$ and the rescaled weight variances $C_W^{(\ell)}$ – have pronounced qualitative effects on the behavior of the observables, just as we saw in §3 for deep linear networks. In particular, we'll explain how neural-network behavior becomes increasingly sensitive to these initialization hyperparameters with increasing depth.[3]

Such a tuning brings a network to *criticality*, a term we borrow from statistical physics used to describe self-similar systems. To this end, we give a general prescription for tuning initialization hyperparameters to their critical values for a given activation function and network architecture (§5.2 and §5.3). In the process, we also identify some activation functions that don't allow for criticality. We will also see that certain activation functions behave very similarly to each other when tuned to criticality, highlighting an important connection to the notion of *universality* in statistical physics.

The study of finite-width corrections at criticality then leads us to one of the main results of this chapter, an *emergent scale* given by the aspect ratio of the network depth to the network width, L/n (§5.4). This aspect ratio ultimately serves as the *cutoff* of our effective theory, controlling the region of validity of our effective theory as well as determining the strength and importance of the finite-width corrections to the infinite-width description. On the one end of the spectrum, we find that the shorter and fatter networks are, the more and more that they behave like their infinite-width counterparts. On the other end, skinny and tall networks become increasingly dominated by non-Gaussian *fluctuations* due to *interactions* between neurons. Overall, this serves to generalize our fluctuation analysis for deep linear networks in §3.3.

Lastly, we'll conclude the chapter by addressing and resolving a subtlety that arises in our criticality analysis for non-smooth activation functions such as the `ReLU` (§5.5).

5.1 Criticality Analysis of the Kernel

For the bulk of this chapter, our goal is to extend the notion of criticality discussed in §3 to deep MLPs with general activation functions $\sigma(z)$. Our starting point is the kernel recursion

$$K_{\alpha\beta}^{(\ell+1)} = C_b + C_W \left\langle \sigma_\alpha \sigma_\beta \right\rangle_{K^{(\ell)}} , \tag{5.1}$$

[2]Despite the asymptotic nature of these solutions, we note many of these tools were developed to study the strong interactions, where a parameter that is 3 in practice is taken to be infinity. Thus, sometimes even $3 \sim \infty$, since $1/3 \ll 1$ can be made to work as perturbative parameter [42].

[3]The initial part of this analysis was first carried out in a series of papers, [43–45], using a different set of techniques that are ultimately equivalent to ours in the infinite-width limit. Extending this analysis, we'll identify two very general conditions according to the *principle of criticality* that let us determine the correct order-one values for the initialization hyperparameters. In §10.3, we'll see that the need for these conditions can also be understood by demanding that fully-trained networks generalize well.

derived in the previous chapter. As a reminder, the *kernel* $K^{(\ell)}_{\alpha\beta}$, defined by (4.116), is the infinite-width limit of the mean metric $G^{(\ell)}_{\alpha\beta}$. Here, in order to restrict the number of distinct hyperparameters, we have set the bias variance $C_b^{(\ell)} = C_b$ and the rescaled weight variance $C_W^{(\ell)} = C_W$ to be layer-independent. The initial condition for this recursion is given by the first-layer kernel

$$K^{(1)}_{\alpha\beta} = C_b + C_W \left(\frac{1}{n_0} \sum_{i=1}^{n_0} x_{i;\alpha} x_{i;\beta} \right) , \tag{5.2}$$

set by the inner products of inputs $\sum_{i=1}^{n_0} x_{i;\alpha} x_{i;\beta}$. Our goal is to analyze how the kernel changes as a function of layer and, as we'll see, this analysis at the level of the kernel is sufficient to pin down the critical initialization hyperparameters to leading order in 1/width.

For deep linear networks, the Gaussian expectation of the activations with respect to the kernel is just given by the kernel, $\langle \sigma_\alpha \sigma_\beta \rangle_{K^{(\ell)}} = \langle z_\alpha z_\beta \rangle_{K^{(\ell)}} = K^{(\ell)}_{\alpha\beta}$, and the recursion equation was simple enough for us to obtain a full solution. Stepping outside the realm of the `linear` activation function, however, the kernel recursion acquires two new complications that require some care: (i) the expectation value $\langle \sigma_\alpha \sigma_\beta \rangle_{K^{(\ell)}}$ will be a nonlinear function of the kernel, and (ii) for two distinct inputs $\alpha \neq \beta$ the recursion mixes off-diagonal components $K^{(\ell)}_{\alpha\beta}$ with diagonal components $K^{(\ell)}_{\alpha\alpha}$ and $K^{(\ell)}_{\beta\beta}$.

Let's illustrate this with a quadratic activation function $\sigma(z) = z^2$. As should be second nature by now, evaluating (5.1) requires two pairs of Wick contractions, giving

$$\begin{aligned} K^{(\ell+1)}_{\alpha\beta} &= C_b + C_W \left\langle z_\alpha^2 z_\beta^2 \right\rangle_{K^{(\ell)}} \\ &= C_b + C_W \left(K^{(\ell)}_{\alpha\alpha} K^{(\ell)}_{\beta\beta} + 2 K^{(\ell)}_{\alpha\beta} K^{(\ell)}_{\alpha\beta} \right) . \end{aligned} \tag{5.3}$$

Thus, unlike deep linear networks, for quadratic activations $K^{(\ell+1)}_{\alpha\beta}$ depends not only on $K^{(\ell)}_{\alpha\beta}$ but also on $K^{(\ell)}_{\alpha\alpha}$ and $K^{(\ell)}_{\beta\beta}$, requiring us to solve three coupled nonlinear recursion equations for $\alpha \neq \beta$. This mixing is generic for nonlinear activation functions.

For practitioners, this is good news: the off-diagonal elements of the kernel are related to the generalization ability of the network. For deep linear networks the lack of mixing via nonlinearity suggests an *inductive bias* that limits such networks' ability to develop nontrivial correlations for pairs of samples. While mixing is a benefit in practice, it's an obstacle in theory, albeit a surmountable one. Since the kernel recursion mixes at most two inputs, it is sufficient to analyze the case with a single input and the case with two distinct inputs. We shall now perform these analyses in turn, with an eye towards deriving the general conditions for criticality.

A single input

Each diagonal component of the kernel can be solved self-consistently by itself. Specifically, labeling a single input by $\alpha = 0$,

$$\begin{aligned} K_{00}^{(\ell+1)} &= C_b + C_W \left\langle \sigma(z_0)\, \sigma(z_0) \right\rangle_{K^{(\ell)}} \\ &= C_b + C_W g\Big(K_{00}^{(\ell)}\Big) \ . \end{aligned} \tag{5.4}$$

Here, we introduced a helper function

$$g(K) \equiv \left\langle \sigma(z)\, \sigma(z) \right\rangle_K \equiv \frac{1}{\sqrt{2\pi K}} \int_{-\infty}^{\infty} dz\ e^{-\frac{z^2}{2K}}\, \sigma(z)\, \sigma(z)\,, \tag{5.5}$$

to emphasize that the expectation $\left\langle \sigma(z_0)\sigma(z_0) \right\rangle_{K^{(\ell)}}$ is a function only of a single component of the kernel, $K_{00}^{(\ell)}$. By focusing our attention first on the single-input kernel, we can deal with the nonlinearity before confronting the mixing of kernel components.

In particular, the single-input recursion (5.4) is really telling us how the average magnitude of the preactivations for the input

$$K_{00}^{(\ell)} = \mathbb{E}\left[\frac{1}{n_\ell} \sum_{i=1}^{n_\ell} \Big(z_{i;0}^{(\ell)}\Big)^2 \right] , \tag{5.6}$$

changes as a function of layer ℓ, with the initial condition for the recursion set by (5.2). For the same reasons we considered in §3.2, we would like that the kernel $K_{00}^{(\ell)}$ neither exponentially explode nor exponentially vanish. However, such exponential behavior is generic, and so for most choices of initialization hyperparameters (C_b, C_W), the kernel will either explode exponentially towards a *trivial* fixed point at infinity or collapse exponentially onto a trivial fixed point at a finite value $K_{00}^\star$. Thus, our first criticality condition is to mitigate this exploding or collapsing kernel problem for the single input.

There is an ancient technique used to analyze nonlinear recursions such as the single-input kernel recursion (5.4): linearization around a fixed point. Namely, we first identify a fixed point of the recursion, i.e. a value $K_{00}^\star$ that satisfies

$$K_{00}^\star = C_b + C_W g(K_{00}^\star) \ , \tag{5.7}$$

and then expand the kernel around it as

$$K_{00}^{(\ell)} = K_{00}^\star + \Delta K_{00}^{(\ell)} . \tag{5.8}$$

This expansion for the single-input recursion (5.4) results in the linearized recursion

$$\Delta K_{00}^{(\ell+1)} = \chi_\parallel(K_{00}^\star)\, \Delta K_{00}^{(\ell)} + O\Big(\Delta^2\Big) \ , \tag{5.9}$$

where we introduced the **parallel susceptibility**

$$\begin{aligned} \chi_\parallel(K) &\equiv C_W g'(K) \\ &= C_W \frac{d}{dK} \left[\frac{1}{\sqrt{2\pi K}} \int_{-\infty}^{\infty} dz\, e^{-\frac{z^2}{2K}} \sigma(z)\, \sigma(z) \right] \\ &= \frac{C_W}{2K^2} \Big\langle \sigma(z)\, \sigma(z) \left(z^2 - K\right) \Big\rangle_K \ . \end{aligned} \tag{5.10}$$

The susceptibility $\chi_{\parallel}(K)$ characterizes how susceptible the kernel is to perturbations around the fixed point, hence the name: the kernel value exponentially expands away from or contracts towards the fixed-point value, according to whether $\chi_{\parallel}\left(K^{\star}_{00}\right) > 1$ or $\chi_{\parallel}\left(K^{\star}_{00}\right) < 1$. (The label *parallel* will be explained at the very end of this section.)

Thus, we see that in order to mitigate this *exploding and vanishing kernel problem* for a single input at linear order, we require the tuning of initialization hyperparameters (C_b, C_W) such that

$$\chi_{\parallel}(K^{\star}_{00}) = 1\,, \tag{5.11}$$

with the fixed-point value $K^{\star}_{00}$ defined implicitly through the fixed-point equation (5.7). As we shall detail later, criticality can happen in three ways depending on the choice of activation functions.

- First, as we saw for deep linear networks, the single-input kernel can be perfectly preserved as $K^{(\ell)}_{00} = K^{(1)}_{00} = C_b + C_W\left(\sum_i x_{i;0}x_{i;0}/n_0\right)$, resulting in a line of fixed points parametrized by input norms $\sum_i x_{i;0}x_{i;0}$. We will see this happening in §5.2 for *scale-invariant* activation functions due to the absence of higher-order corrections $O\left(\Delta^{p>1}\right)$ in (5.9).

- Second, the kernel can slowly decay toward a fixed point $K^{\star}_{00} = 0$ for all input norms, with a power law $K^{(\ell)}_{00} \sim 1/\ell^q$ with $0 < q \leq 1$. We will see this happening in §5.3.3 for a class of activation functions that include `tanh` and `sin`, due to the presence of $O\left(\Delta^{p>1}\right)$ corrections in (5.9).

- Third, criticality can happen with a power-law decay towards a nonzero fixed-point value $K^{\star}_{00} \neq 0$. We will see this happening in §5.3.4 for the `SWISH` and `GELU` activation functions.

In all these cases, when initialization hyperparameters are tuned to criticality, we call $K^{\star}_{00} = K^{\star}_{00}\left(C_b^{\text{critical}}, C_W^{\text{critical}}\right)$ a **nontrivial fixed point**, distinguishing it from trivial fixed points for generic hyperparameters around which perturbations behave exponentially.

Two inputs

Now, given two distinct inputs, let's label them with sample indices $\alpha = \pm$. For such a pair of inputs, we have three distinct kernel components to consider: $K^{(\ell)}_{++}$, $K^{(\ell)}_{--}$, and $K^{(\ell)}_{+-} = K^{(\ell)}_{-+}$. The single-input analysis can be directly applied to determine the layer dependence of the diagonal components $K^{(\ell)}_{++}$ and $K^{(\ell)}_{--}$, so to complete our analysis we need to extract the layer dependence of the off-diagonal component $K^{(\ell)}_{+-}$, given solutions for the diagonal pieces. Such an analysis will yield a second criticality condition that, together with (5.11), will pin down the critical initialization hyperparameters $(C_b, C_W)^{\text{critical}}$ for a given activation function.

Ultimately, our approach will be to linearize around the degenerate limit where both inputs coincide identically, i.e., $x_{i;+}, x_{i;-} \to x_{i;0}$. In such a limit, all the ways of pairing

up the two inputs are the same, and so all the components of the full kernel matrix must take the same value, i.e., $K^{(\ell)}_{++}, K^{(\ell)}_{--}, K^{(\ell)}_{+-} \to K^{(\ell)}_{00}$. Thus, each recursion degenerates to the same single-input recursion (5.4), which we know has a fixed-point value $K^{\star}_{00}$. This means that the coincident-limit solution,

$$\begin{pmatrix} K^{(\ell)}_{++} & K^{(\ell)}_{+-} \\ K^{(\ell)}_{-+} & K^{(\ell)}_{--} \end{pmatrix} = \begin{pmatrix} K^{\star}_{00} & K^{\star}_{00} \\ K^{\star}_{00} & K^{\star}_{00} \end{pmatrix} = K^{\star}_{00} \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} , \tag{5.12}$$

must also be a fixed point of the full kernel recursion for the two inputs.

There are three different kinds of perturbations that we need to consider in order to understand the approach of the full kernel matrix to this degenerate fixed point. The first kind corresponds to the perturbation $\Delta K^{(\ell)}_{00}$ that appeared in our single-input analysis, which controls how the average of the kernel's four components approaches the fixed-point value $K^{\star}_{00}$. Next, in backing off the coincident limit, the single input splits into two distinct inputs, $x_{i;0} \to x_{i;+}, x_{i;-}$. Then, we could imagine separating these two inputs so that they become endowed with different magnitudes, i.e. $\sum_i x^2_{i;+} \neq \sum_i x^2_{i;-}$, and follow the expected evolution of this difference through the network:

$$R^{(\ell)} \equiv \mathbb{E}\left[\frac{1}{n_\ell}\sum_{i=1}^{n_\ell}\left(z^{(\ell)}_{i;+}\right)^2\right] - \mathbb{E}\left[\frac{1}{n_\ell}\sum_{i=1}^{n_\ell}\left(z^{(\ell)}_{i;-}\right)^2\right] = K^{(\ell)}_{++} - K^{(\ell)}_{--} . \tag{5.13}$$

Such a perturbation is actually still covered by the single-input analysis, since the evolutions of the diagonal components $K^{(\ell)}_{++}$ and $K^{(\ell)}_{--}$ are mutually independent of each other, with their approach to the fixed point simply controlled by the single-input recursion.

Finally, rather than considering the difference of the squares, we could consider the square of the difference

$$D^{(\ell)} \equiv \mathbb{E}\left[\frac{1}{n_\ell}\sum_{i=1}^{n_\ell}\left(z^{(\ell)}_{i;+} - z^{(\ell)}_{i;-}\right)^2\right] = K^{(\ell)}_{++} + K^{(\ell)}_{--} - 2K^{(\ell)}_{+-} , \tag{5.14}$$

where to get the expression on the right-hand side we expanded the binomial and then used the definition of the kernel components. This quantity measures the magnitude of the difference between the two inputs after being passed through ℓ layers of the network and can be non-vanishing even when such inputs themselves have the same magnitudes, i.e. $\sum_i x^2_{i;+} = \sum_i x^2_{i;-}$. Importantly, this distance measure $D^{(\ell)}$ depends on the off-diagonal component of the kernel, $K^{(\ell)}_{+-}$, and so we expect that analyzing this perturbation will give something new. As we will see, the approach of this third perturbation $D^{(\ell)}$ to the coincident fixed point with $D^{(\ell)} = 0$ will yield a second criticality condition. Together with the single-input criticality condition (5.11), this will be sufficient to completely determine the critical initialization hyperparameters.

Let's translate the above discussion into math. To do so, we will find it convenient to project the full kernel matrix into the following basis

$$K^{(\ell)}_{\alpha_1\alpha_2} = \begin{pmatrix} K^{(\ell)}_{++} & K^{(\ell)}_{+-} \\ K^{(\ell)}_{-+} & K^{(\ell)}_{--} \end{pmatrix} = K^{(\ell)}_{[0]}\gamma^{[0]}_{\alpha_1\alpha_2} + K^{(\ell)}_{[1]}\gamma^{[1]}_{\alpha_1\alpha_2} + K^{(\ell)}_{[2]}\gamma^{[2]}_{\alpha_1\alpha_2} , \tag{5.15}$$

where we've introduced symmetric matrices

$$\gamma^{[0]}_{\alpha_1\alpha_2} \equiv \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}, \quad \gamma^{[1]}_{\alpha_1\alpha_2} \equiv \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \quad \gamma^{[2]}_{\alpha_1\alpha_2} \equiv \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}. \tag{5.16}$$

In this basis, the components of the kernel are

$$K^{(\ell)}_{[0]} = \frac{1}{4}\left[K^{(\ell)}_{++} + K^{(\ell)}_{--} + 2K^{(\ell)}_{+-}\right] = \mathbb{E}\left[\frac{1}{n_\ell}\sum_{i=1}^{n_\ell}\left(\frac{z^{(\ell)}_{i;+} + z^{(\ell)}_{i;-}}{2}\right)^2\right], \tag{5.17}$$

$$K^{(\ell)}_{[1]} = \frac{1}{2}\left[K^{(\ell)}_{++} - K^{(\ell)}_{--}\right] = \frac{1}{2}R^{(\ell)}, \tag{5.18}$$

$$K^{(\ell)}_{[2]} = \frac{1}{4}\left[K^{(\ell)}_{++} + K^{(\ell)}_{--} - 2K^{(\ell)}_{+-}\right] = \frac{1}{4}D^{(\ell)}. \tag{5.19}$$

This basis was strategically chosen so that both $K^{(\ell)}_{[1]}$ and $K^{(\ell)}_{[2]}$ correspond to our two natural distance measures of two distinct inputs – the difference in the magnitudes $R^{(\ell)}$ and the magnitude of the difference $D^{(\ell)}$ – and both vanish in the coincident limit. The remaining component, $K^{(\ell)}_{[0]}$, measures the overall average magnitude or the magnitude of the center of mass of the two ℓ-th-layer preactivations.

This basis has two additional nice properties that will later prove useful: (i) both $K^{(\ell)}_{[0]}$ and $K^{(\ell)}_{[2]}$ are even (invariant) under the parity swap of two inputs $+ \leftrightarrow -$, while $K^{(\ell)}_{[1]}$ is odd, changing its sign $K^{(\ell)}_{[1]} \to -K^{(\ell)}_{[1]}$ as $+ \leftrightarrow -$, and (ii) the $\gamma^{[a]}$ matrices are orthogonal.[4]

Now, let's discuss perturbations around the coincident fixed point (5.12) in terms of this new basis. These perturbations have a very natural interpretation in terms of two infinitesimally-separated input points, $x_{i;+}$ and $x_{i;-}$, perturbed around a **midpoint input** $x_{i;0} \equiv (x_{i;+} + x_{i;-})/2$ as

$$x_{i;\pm} = x_{i;0} \pm \frac{1}{2}\delta x_i\,. \tag{5.21}$$

The dynamics of such preactivations $z^{(\ell)}_{i;\pm} \equiv z^{(\ell)}_i(x_\pm)$ then encode the evolution of these perturbed signals through the network as a function of layer depth ℓ. The coincident limit corresponds to $\delta x_i \to 0$ and, as we back off from this limit, we should be able

[4]This symmetry decomposition mirrors tensorial decomposition used by physicists to organize particles by their spin. Operationally, we can project out the components of any 2×2 matrix $M_{\alpha\beta}$ into components $M_{[a]}$ in the $\gamma^{[a]}$ basis by tracing over the sample indices and normalizing

$$M_{[a]} = \frac{\sum_{\alpha,\beta} M_{\alpha\beta}\gamma^{[a]}_{\beta\alpha}}{\sum_{\alpha,\beta}\gamma^{[a]}_{\alpha\beta}\gamma^{[a]}_{\beta\alpha}}, \tag{5.20}$$

with $\alpha, \beta \in \{+, -\}$. One can easily check that the $\gamma^{[a]}_{\alpha\beta}$ matrices themselves are orthogonal under this inner product (5.20).

to expand the $K^{(\ell)}_{[a]}$ components around their coincident-limit values $K^{(\ell)}_{[0]} = K^{(\ell)}_{00}$ and $K^{(\ell)}_{[1]} = K^{(\ell)}_{[2]} = 0$. This results in an expansion

$$K^{(\ell)}_{[0]} = K^{(\ell)}_{00} + \delta\delta K^{(\ell)}_{[0]} + O\Big(\delta^4\Big) \,, \tag{5.22}$$

$$K^{(\ell)}_{[1]} = \delta K^{(\ell)}_{[1]} + \delta\delta\delta K^{(\ell)}_{[1]} + O\Big(\delta^5\Big) \,, \tag{5.23}$$

$$K^{(\ell)}_{[2]} = \delta\delta K^{(\ell)}_{[2]} + \delta\delta\delta\delta K^{(\ell)}_{[2]} + O\Big(\delta^6\Big) \,, \tag{5.24}$$

where the order of the kernel perturbation in δx is denoted by a preceding δ^p, and we used the even/odd behavior of the components $K^{(\ell)}_{[a]}$ under the parity symmetry $+ \leftrightarrow -$ to limit which terms appear in each expansion.[5] Next, we will use these expansions to determine whether the behavior of the leading perturbations $\delta K^{(\ell)}_{[1]}$ and $\delta\delta K^{(\ell)}_{[2]}$ around their fixed point values are exponential, power-law, or constant.

To do so, we'll need to expand the original kernel recursion (5.1) order by order in δ. Before embarking on such an algebraic journey, let's think about what we should expect. At the zeroth order in δ, we'll just recover the recursion for the single-input kernel (5.4)

$$K^{(\ell+1)}_{00} = C_b + C_W g\Big(K^{(\ell)}_{00}\Big) \,. \tag{5.25}$$

This should demystify our notational choice in the single-input analysis, as $K^{(\ell)}_{00}$ simply represents the kernel for a single input x_0 corresponding to the midpoint of a pair of inputs x_+, x_-, as per (5.21). Going forward, we will call $K^{(\ell)}_{00}$ the **midpoint kernel**.[6]

Next, at first order in δ, we will get the recursion

$$\delta K^{(\ell+1)}_{[1]} = \chi_{\parallel}\Big(K^{(\ell)}_{00}\Big)\, \delta K^{(\ell)}_{[1]} \,. \tag{5.26}$$

This is to be expected. On account of the parity symmetry, $\delta K^{(\ell+1)}_{[1]}$ can only be proportional to $\delta K^{(\ell)}_{[1]}$ at this order, and the proportionality factor must be none other than the parallel susceptibility, because $K^{(\ell)}_{[1]} = \frac{1}{2}\Big[K^{(\ell)}_{++} - K^{(\ell)}_{--}\Big]$ behaves in the same way as the single-input kernels: if the single-input kernels behave exponentially, then this difference should as well.

Lastly, at the second order in δ, we expect a recursion of the form

$$\delta\delta K^{(\ell+1)}_{[2]} = [\text{something}]\, \delta\delta K^{(\ell)}_{[2]} + [\text{something}']\, \Big(\delta K^{(\ell)}_{[1]}\Big)^2 + [\text{something}'']\, \delta\delta K^{(\ell)}_{[0]} \,, \tag{5.27}$$

[5] These expansions are valid when the activation function $\sigma(z)$ is sufficiently smooth. For non-smooth activation functions such as `ReLU`, these expansions are more complicated, though still analyzable. We will consider these subtleties in more detail in §5.5.

[6] We note that $K^{(\ell)}_{[0]}$ is the kernel for the midpoint of the layer-ℓ preactivations, $(z^{(\ell)}_{i;+} + z^{(\ell)}_{i;-})/2$, which is not quite the same as the midpoint kernel $K^{(\ell)}_{00}$ for the preactivations of the midpoint input x_0 propagated to layer ℓ. The difference is expressed in (5.22) and will turn out negligible for quantities at leading order in the δ expansion.

which is the most general form it can take given the even parity symmetry of $\delta\delta K^{(\ell)}_{[2]}$, and where the [somethings] can be functions of the single-input kernel $K^{(\ell)}_{00}$. In the rest of this subsection we will derive the form of [something] and [something$'$], while also showing that [something$''$] vanishes due to the orthogonality of the $\gamma^{[a]}_{\alpha\beta}$ matrices.

Already at this heuristic level of the analysis, the bootstrapping nature of the system of equations should be clear. First, we find a solution for the midpoint kernel $K^{(\ell)}_{00}$, which then bootstraps the layer dependence of $\delta K^{(\ell)}_{[1]}$ through (5.26), the solution of which in turn feeds into (5.27) and together with $K^{(\ell)}_{00}$ bootstraps the layer dependence of $\delta\delta K^{(\ell)}_{[2]}$. In other words, rather than confronting three coupled nonlinear recursions, we can solve decoupled recursions one by one.

Deriving bootstrapped recursions

Now let's embark on our algebraic journey. Our goal is to expand the kernel recursion (5.1) up to the second order in δ. This requires us to evaluate $\langle\sigma(z_+)\sigma(z_+)\rangle_{K^{(\ell)}}$, $\langle\sigma(z_-)\sigma(z_-)\rangle_{K^{(\ell)}}$, and $\langle\sigma(z_+)\sigma(z_-)\rangle_{K^{(\ell)}}$ to that order, all of which are two-dimensional Gaussian integrals. Rather than treating all of these Gaussian integrals separately, we instead will evaluate the Gaussian expectation of an arbitrary function $\langle F(z_+,z_-)\rangle_{K^{(\ell)}}$ and then will plug in $F(z_+,z_-)=\sigma(z_+)\sigma(z_+)$, $\sigma(z_-)\sigma(z_-)$, or $\sigma(z_+)\sigma(z_-)$. Moreover, we will find that this general expression $\langle F(z_+,z_-)\rangle_{K^{(\ell)}}$ will come in handy in later chapters.

In order to evaluate this Gaussian expectation, it is natural to write the integral in the eigenbasis of the kernel rather than in the (z_+,z_-) coordinates. Denote such orthonormal eigenvectors by $\{\hat{e}^u,\hat{e}^w\}$, which satisfy the eigenvalue equations

$$\sum_{\beta=\pm} K^{(\ell)}_{\alpha\beta}\hat{e}^u_\beta = \lambda_u \hat{e}^u_\alpha\,, \qquad \sum_{\beta=\pm} K^{(\ell)}_{\alpha\beta}\hat{e}^w_\beta = \lambda_w \hat{e}^u_\alpha\,, \tag{5.28}$$

with eigenvalues λ_u and λ_w, respectively. Transforming to coordinates (u,w) defined via

$$z_\alpha(u,w) = u\hat{e}^u_\alpha + w\hat{e}^w_\alpha\,, \tag{5.29}$$

the Gaussian expectation becomes

$$\langle F(z_+,z_-)\rangle_{K^{(\ell)}} = \frac{\int du dw\ \exp\!\left(-\frac{u^2}{2\lambda_u}-\frac{w^2}{2\lambda_w}\right) F\Big(z_+(u,w)\,,\ z_-(u,w)\Big)}{\int du dw\ \exp\!\left(-\frac{u^2}{2\lambda_u}-\frac{w^2}{2\lambda_w}\right)}\,. \tag{5.30}$$

As we discussed in §1.3, this equation expresses the idea that the (u,w)-coordinate basis diagonalizes the kernel such that the distribution factorizes as $p(z_+,z_-)=p(u)p(w)$. The integral in the denominator represents the normalization factor of this factorized Gaussian expectation.

Now, we need to actually determine the eigenvalues λ_u and λ_w and eigenvectors $\{\hat{e}^u,\hat{e}^w\}$. We'll start our perturbative eigen-analysis by taking the by-now-familiar coincidental limit, $\delta\to 0$. As we discussed around (5.22), in this limit the kernel is

degenerate:

$$K^{(\ell)}_{\alpha\beta} = K^{(\ell)}_{00}\gamma^{[0]}_{\alpha\beta} = K^{(\ell)}_{00}\begin{pmatrix}1 & 1\\ 1 & 1\end{pmatrix}, \tag{5.31}$$

with the $\gamma^{[0]}_{\alpha\beta}$ component equal to the midpoint kernel $K^{(\ell)}_{[0]} = K^{(\ell)}_{00}$, and the other components vanishing. Such a matrix has the normalized eigenvectors

$$\hat{e}^u_\alpha = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\ 1\end{pmatrix}, \qquad \text{and} \qquad \hat{e}^w_\alpha = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\ -1\end{pmatrix}, \tag{5.32}$$

with eigenvalues $\lambda_u = 2K^{(\ell)}_{00}$ and $\lambda_w = 0$, respectively.[7]

Next, let's back off from the coincidental limit and look back at the δ expansions (5.22)–(5.24) for $K^{(\ell)}_{[0,1,2]}$ around the midpoint kernel. With similar expansions for the eigenvectors $\hat{e}^{u,w}_\pm$ and eigenvalues $\lambda_{u,w}$, we can solve the eigenvalue equations (5.28) order by order.[8] Carrying out such expansions (in the margins or – if this isn't your personal copy of our book – in a private notebook) and solving (5.28) to second order, we find normalized eigenvectors

$$\hat{e}^u_\alpha = \begin{pmatrix}\hat{e}^u_+\\ \hat{e}^u_-\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}1 + \frac{\delta K^{(\ell)}_{[1]}}{2K^{(\ell)}_{00}} - \frac{1}{8}\left(\frac{\delta K^{(\ell)}_{[1]}}{K^{(\ell)}_{00}}\right)^2\\ 1 - \frac{\delta K^{(\ell)}_{[1]}}{2K^{(\ell)}_{00}} - \frac{1}{8}\left(\frac{\delta K^{(\ell)}_{[1]}}{K^{(\ell)}_{00}}\right)^2\end{pmatrix} + O\Big(\delta^3\Big), \tag{5.33}$$

$$\hat{e}^w_\alpha = \begin{pmatrix}\hat{e}^w_+\\ \hat{e}^w_-\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}1 - \frac{\delta K^{(\ell)}_{[1]}}{2K^{(\ell)}_{00}} - \frac{1}{8}\left(\frac{\delta K^{(\ell)}_{[1]}}{K^{(\ell)}_{00}}\right)^2\\ -1 - \frac{\delta K^{(\ell)}_{[1]}}{2K^{(\ell)}_{00}} + \frac{1}{8}\left(\frac{\delta K^{(\ell)}_{[1]}}{K^{(\ell)}_{00}}\right)^2\end{pmatrix} + O\Big(\delta^3\Big),$$

and corresponding eigenvalues

$$\lambda_u = 2K^{(\ell)}_{00} + 2\delta\delta K^{(\ell)}_{[0]} + \frac{\Big(\delta K^{(\ell)}_{[1]}\Big)^2}{2K^{(\ell)}_{00}} + O\Big(\delta^4\Big), \tag{5.34}$$

$$\lambda_w = 2\delta\delta K^{(\ell)}_{[2]} - \frac{\Big(\delta K^{(\ell)}_{[1]}\Big)^2}{2K^{(\ell)}_{00}} + O\Big(\delta^4\Big).$$

Even if you don't have a private notebook, it's easy to check on a scrap of paper that (5.33) and (5.34) solve (5.28) to $O(\delta^2)$.

[7] Here, the zero eigenvalue for w signifies that the matrix is degenerate. This implies that the distribution for the w coordinate is given by a Dirac delta function, $p(w) = \delta(w)$, indicating that there's really only one input in this limit.

[8] For physicists, note that this is second-order time-independent perturbation theory from quantum mechanics.

Now, having solved the eigenproblem, we can implement the change of coordinates. Before doing so, notice that the u coordinate is closely related to the coordinate z_0, the preactivation corresponding to the midpoint input. This makes it very natural to use z_0 as a coordinate instead of u. We can implement this by rescaling u as

$$\frac{u^2}{2\lambda_u} = \frac{z_0^2}{2K_{00}^{(\ell)}}\,, \tag{5.35}$$

changing variables in the integral (5.30) so that the Gaussian integral over u becomes a Gaussian integral over z_0 with a variance given by the midpoint kernel $K_{00}^{(\ell)}$. With this rescaling, the full coordinate transformation becomes

$$z_\pm(z_0, w) = z_0\left[1 \pm \left(\frac{\delta K_{[1]}^{(\ell)}}{2K_{00}^{(\ell)}}\right) + \left(\frac{\delta\delta K_{[0]}^{(\ell)}}{2K_{00}^{(\ell)}}\right) + O\Big(\delta^3\Big)\right] + \frac{w}{\sqrt{2}}\left[\pm 1 + O(\delta)\right]. \tag{5.36}$$

Here, we can truncate the term in the square brackets multiplying w at $O(1)$, since the w coordinate has zero mean and a variance $\lambda_w = O(\delta^2)$. This means that, when performing the w integration, terms proportional to w^0 will be $O(1)$, terms proportional to w^2 will be $O(\delta^2)$, higher-order terms will be subleading, and, of course, all the odd terms will vanish. By contrast, the z_0 coordinate has zero mean and a variance $K_{00}^{(\ell)} = O(1)$, so we actually need keep terms up to $O(\delta^2)$.

Next, we need to plug this expression (5.36) into our arbitrary function

$$F(z_+, z_-) = F\Big(z_+(z_0, w),\ z_-(z_0, w)\Big)\,, \tag{5.37}$$

now viewed as a function of the two independent Gaussian variables z_0 and w, and perform the integration over them. To do so, first we need to Taylor expand the function in both δ and w around $F(z_0, z_0)$. This gives

$$\begin{aligned}
&F(z_+, z_-) \\
=&F(z_0, z_0) + z_0\left(\frac{\delta K_{[1]}^{(\ell)}}{2K_{00}^{(\ell)}}\right)(\partial_+ - \partial_-)\,F - z_0\left(\frac{\delta\delta K_{[0]}^{(\ell)}}{2K_{00}^{(\ell)}}\right)(\partial_+ + \partial_-)\,F \\
&+ z_0^2\left(\frac{\delta K_{[1]}^{(\ell)}}{2K_{00}^{(\ell)}}\right)^2\frac{(\partial_+ - \partial_-)^2\,F}{2} + \frac{w^2}{2}\frac{(\partial_+ - \partial_-)^2\,F}{2} \\
&+ (\text{odd in } w) + O\Big(\delta^3, w^2\delta, w^4\Big)\,,
\end{aligned} \tag{5.38}$$

with the abbreviation $\partial_+^p\partial_-^q F \equiv \partial_+^p\partial_-^q F(z_+, z_-)|_{z_+=z_-=z_0}$. The Gaussian integral over w is simple to perform, we just replace w^2 with its variance λ_w (5.34). Finally, we will express our final answer in terms of single-variable Gaussian expectations over the variable z_0 – which, as you should recall, has a variance given by the scalar midpoint

kernel $K^{(\ell)}_{00}$ – giving

$$\begin{aligned}
&\langle F(z_+, z_-)\rangle_{K^{(\ell)}} \qquad (5.39)\\
&= \langle F(z_0, z_0)\rangle_{K^{(\ell)}_{00}} + \left(\frac{\delta K^{(\ell)}_{[1]}}{2K^{(\ell)}_{00}}\right) \langle z_0 \left(\partial_+ - \partial_-\right) F\rangle_{K^{(\ell)}_{00}} + \left(\frac{\delta\delta K^{(\ell)}_{[0]}}{2K^{(\ell)}_{00}}\right) \langle z_0 \left(\partial_+ + \partial_-\right) F\rangle_{K^{(\ell)}_{00}}\\
&\quad + \frac{1}{2}\left\langle \left[\delta\delta K^{(\ell)}_{[2]} + \left(\frac{\delta K^{(\ell)}_{[1]}}{2K^{(\ell)}_{00}}\right)^2 \left(z_0^2 - K^{(\ell)}_{00}\right)\right] \left(\partial_+ - \partial_-\right)^2 F \right\rangle_{K^{(\ell)}_{00}} + O\left(\delta^3\right) .
\end{aligned}$$

This completes our computation of this general expectation.

In order to apply this formula to evaluate the expectations $\langle \sigma(z_\alpha)\sigma(z_\beta)\rangle_{K^{(\ell)}}$ in the kernel recursion (5.1), recall definitions of gamma matrices in (5.16) and note

$$\left[\sigma(z_\alpha)\sigma(z_\beta)\right]|_{z_+=z_-=z_0} = \sigma(z_0)\sigma(z_0)\gamma^{[0]}_{\alpha\beta} , \qquad (5.40)$$

$$\left\{\left(\partial_+ - \partial_-\right)\left[\sigma(z_\alpha)\sigma(z_\beta)\right]\right\}|_{z_+=z_-=z_0} = 2\sigma'(z_0)\sigma(z_0)\gamma^{[1]}_{\alpha\beta} , \qquad (5.41)$$

$$\left\{\left(\partial_+ + \partial_-\right)\left[\sigma(z_\alpha)\sigma(z_\beta)\right]\right\}|_{z_+=z_-=z_0} = 2\sigma'(z_0)\sigma(z_0)\gamma^{[0]}_{\alpha\beta} , \qquad (5.42)$$

$$\left\{\left(\partial_+ - \partial_-\right)^2\left[\sigma(z_\alpha)\sigma(z_\beta)\right]\right\}|_{z_+=z_-=z_0} = 2\sigma''(z_0)\sigma(z_0)\gamma^{[0]}_{\alpha\beta} + 2\sigma'(z_0)\sigma'(z_0)\gamma^{[2]}_{\alpha\beta} . \qquad (5.43)$$

Plugging these individually into our general expression (5.39), we get

$$\begin{aligned}
&\langle \sigma(z_\alpha)\sigma(z_\beta)\rangle_{K^{(\ell)}} \qquad (5.44)\\
&= \left[\langle \sigma(z_0)\sigma(z_0)\rangle_{K^{(\ell)}_{00}} + O\left(\delta^2\right)\right]\gamma^{[0]}_{\alpha\beta}\\
&\quad + \left[\left(\frac{\delta K^{(\ell)}_{[1]}}{K^{(\ell)}_{00}}\right)\langle z_0\sigma'(z_0)\sigma(z_0)\rangle_{K^{(\ell)}_{00}}\right]\gamma^{[1]}_{\alpha\beta}\\
&\quad + \left[\delta\delta K^{(\ell)}_{[2]}\langle \sigma'(z_0)\sigma'(z_0)\rangle_{K^{(\ell)}_{00}} + \left(\frac{\delta K^{(\ell)}_{[1]}}{2K^{(\ell)}_{00}}\right)^2 \left\langle \left(z_0^2 - K^{(\ell)}_{00}\right)\sigma'(z_0)\sigma'(z_0)\right\rangle_{K^{(\ell)}_{00}}\right]\gamma^{[2]}_{\alpha\beta} .
\end{aligned}$$

The coefficients of the matrix $\langle \sigma(z_\alpha)\sigma(z_\beta)\rangle_{K^{(\ell)}}$ in the $\gamma^{[a]}_{\alpha\beta}$ basis can be simply read off from the above expression. Therefore, plugging this into the right-hand side of the full kernel recursion (5.1), we can expand the left-hand side of that equation in this basis as

$$K^{(\ell+1)}_{\alpha\beta} = K^{(\ell+1)}_{[0]}\gamma^{[0]}_{\alpha\beta} + K^{(\ell+1)}_{[1]}\gamma^{[1]}_{\alpha\beta} + K^{(\ell+1)}_{[2]}\gamma^{[2]}_{\alpha\beta} , \qquad (5.45)$$

and equate both sides to find recursions for each component in this basis. These are given just below.

Summary

Just above, we explained how to derive the recursions

$$K_{00}^{(\ell+1)} = C_b + C_W\, g\Big(K_{00}^{(\ell)}\Big)\,, \tag{5.46}$$

$$\delta K_{[1]}^{(\ell+1)} = \chi_{\parallel}\Big(K_{00}^{(\ell)}\Big)\,\delta K_{[1]}^{(\ell)}\,, \tag{5.47}$$

$$\delta\delta K_{[2]}^{(\ell+1)} = \chi_{\perp}\Big(K_{00}^{(\ell)}\Big)\,\delta\delta K_{[2]}^{(\ell)} + h\Big(K_{00}^{(\ell)}\Big)\,\Big(\delta K_{[1]}^{(\ell)}\Big)^2\,. \tag{5.48}$$

Here the by-now familiar helper function (5.5) is defined as

$$g(K) = \big\langle \sigma(z)\,\sigma(z)\big\rangle_K\,, \tag{5.49}$$

the parallel susceptibility that we already encountered in (5.10) is given by

$$\chi_{\parallel}(K) = C_W g'(K) = \frac{C_W}{2K^2}\Big\langle \sigma(z)\,\sigma(z)\,\Big(z^2 - K\Big)\Big\rangle_K = \frac{C_W}{K}\,\big\langle z\,\sigma'(z)\,\sigma(z)\big\rangle_K\,, \tag{5.50}$$

the **perpendicular susceptibility** is newly introduced as

$$\chi_{\perp}(K) \equiv C_W\,\big\langle \sigma'(z)\,\sigma'(z)\big\rangle_K\,, \tag{5.51}$$

and the helper function that generates perturbations $\delta\delta K_{[2]}^{(\ell+1)}$ from perturbations $\delta K_{[1]}^{(\ell)}$ is given by

$$h(K) \equiv \frac{C_W}{4K^2}\Big\langle \sigma'(z)\,\sigma'(z)\,\Big(z^2 - K\Big)\Big\rangle_K = \frac{1}{2}\frac{d}{dK}\chi_{\perp}(K)\,. \tag{5.52}$$

In the last steps of (5.50) and (5.52) we made use of the following identity for the single-variable Gaussian expectation

$$\begin{aligned}\frac{d}{dK}\left[\frac{1}{\sqrt{2\pi K}}\int_{-\infty}^{\infty} dz\, e^{-\frac{z^2}{2K}} F(z)\right] =& \frac{1}{2K^2}\left[\frac{1}{\sqrt{2\pi K}}\int_{-\infty}^{\infty} dz\, e^{-\frac{z^2}{2K}} F(z)(z^2 - K)\right] \\ =& \frac{1}{2K}\left[\frac{1}{\sqrt{2\pi K}}\int_{-\infty}^{\infty} dz\, e^{-\frac{z^2}{2K}}\, z\frac{d}{dz}F(z)\right]\,,\end{aligned} \tag{5.53}$$

where to go from the first line to the second line we integrated by parts. These three recursions (5.46)–(5.48) are sufficient to completely fix the initialization hyperparameters and tune the network to criticality.

The first equation (5.46) is a recursion for the midpoint kernel $K_{00}^{(\ell)}$. To analyze this equation, we look for a fixed-point value $K_{00}^{\star}$ satisfying $K_{00}^{\star} = C_b + C_W g(K_{00}^{\star})$ and then linearize around such a fixed point as $K_{00}^{(\ell)} = K_{00}^{\star} + \Delta K_{00}^{(\ell)}$. Doing so, we see that $\Delta K_{00}^{(\ell+1)} = \chi_{\parallel}(K_{00}^{\star})\,\Delta K_{00}^{(\ell)} + O(\Delta^2)$ and realize that the parallel susceptibility $\chi_{\parallel}(K_{00}^{\star})$ governs the growth/decay of deviations $\Delta K_{00}^{(\ell)}$ from the fixed-point value $K_{00}^{\star}$.

The second equation (5.47) is the first equation (5.46) in disguise, since the $\delta K_{[1]}^{(\ell)}$ component is the leading difference in magnitude $R^{(\ell)} = \Big(K_{++}^{(\ell)} - K_{--}^{(\ell)}\Big)/2$ of preactivations for two inputs. As such, the same susceptibility $\chi_{\parallel}\Big(K_{00}^{(\ell)}\Big)$ governs its growth/decay.

Another perspective is that the $\delta K^{(\ell)}_{[1]}$ component can be generated by considering a perturbation $\delta x_i \propto x_{i;0}$ that is parallel to the original input $x_{i;0}$, creating a difference in the norm of the two inputs. This deviation is naturally measured by $R^{(\ell)}$, and setting $\chi_\parallel(K^\star_{00}) = 1$ ensures that such a perturbation neither exponentially explodes nor exponentially vanishes. And that, after a long-winded journey, explains why we called this susceptibility *parallel*.

This third recursion (5.48) is something new, controlling the layer dependence of the magnitude of the difference of the two inputs $D^{(\ell)} = 4\delta\delta K^{(\ell)}_{[2]} + O(\delta^4)$. Such a perturbation in layer $\ell+1$ is sourced by two types of perturbations in layer ℓ, as exhibited by the two terms on right-hand side of (5.48). One term $\propto \left(\delta K^{(\ell)}_{[1]}\right)^2$ is generated by preactivations in the ℓ-th layer with different norms. The other term $\propto \delta\delta K^{(\ell)}_{[2]}$ is generated by preactivations in the ℓ-th layer with a nonzero difference $D^{(\ell)}$ and is present even if the preactivations have the same norm. Such same-norm perturbations in the infinitesimal regime correspond to perturbations of the input that are *perpendicular* to the midpoint input, i.e. $\sum_{i=1}^{n_0} x_{i;0}\,\delta x_i = 0$. The perpendicular susceptibility $\chi_\perp(K^\star_{00})$ determines the dynamics of such perpendicular perturbations.[9] As a nonzero distance $D^{(\ell)}$ is essential for being able to compare and contrast the two inputs $x_{i;\pm}$ after being propagated to layer ℓ, we need to ensure that this quantity is well behaved. To avoid exponential behavior, we will demand $\chi_\perp(K^\star_{00}) = 1$.

Taken all together, our general notion of criticality requires the following two conditions to hold[10]

$$\chi_\parallel(K^\star_{00}) = 1\,, \qquad \chi_\perp(K^\star_{00}) = 1\ , \tag{5.55}$$

with the fixed-point value of the midpoint kernel $K^\star_{00}$ implicitly defined via

$$K^\star_{00} = C_b + C_W g(K^\star_{00})\ . \tag{5.56}$$

These conditions are sufficient to ensure that the entire kernel matrix is preserved to

[9] An alternative view is that, for a given instantiation of the network, this perpendicular susceptibility $\chi_\perp\left(K^{(\ell)}_{00}\right)$ controls changes of the preactivations with respect to changes in the input. To see that, note that the distance $D^{(\ell)}$ can be rewritten to leading order in the perturbation as

$$D^{(\ell)} = \frac{1}{n_\ell}\sum_{i=1}^{n_\ell}\mathbb{E}\left[\left(z^{(\ell)}_{i;+} - z^{(\ell)}_{i;-}\right)^2\right] = \frac{1}{n_\ell}\sum_{i=1}^{n_\ell}\mathbb{E}\left[\left(\sum_{j=1}^{n_0}\frac{dz^{(\ell)}_{i;0}}{dx_{j;0}}\delta x_j\right)^2\right] + O\big(\delta^4\big)\ . \tag{5.54}$$

This makes quantity $\chi_\perp(K^\star_{00})$ of interest for controlling the infamous *exploding and vanishing gradient problem*, a perspective that we will make more concrete in §9.

[10] Note that this further underscores the need for an ensemble. In §2.3, we motivated the initialization distribution by pointing out that the *zero initialization* $b^{(\ell)}_i = W^{(\ell)}_{ij} = 0$ doesn't break the permutation symmetry among the n_ℓ neurons of a layer. Here we see more generally that any zero-mean deterministic (i.e. $C_W = 0$) distribution for the weights – which includes the zero initialization – cannot satisfy the criticality conditions $\chi_\parallel = \chi_\perp = 1$, since both susceptibilities (5.50) and (5.51) are proportional to C_W. Such a zero-weight initialization will always suffer from an exponential decay towards a trivial fixed point at $K^\star_{00} = C_b$.

leading order, namely that

$$\Delta K_{00}^{(\ell+1)} = \Delta K_{00}^{(\ell)} + O\Big(\Delta^2\Big)\,, \qquad K_{[1]}^{(\ell+1)} = K_{[1]}^{(\ell)} + O\Big(\delta^3\Big)\,, \qquad K_{[2]}^{(\ell+1)} = K_{[2]}^{(\ell)} + O\Big(\delta^4\Big)\,. \tag{5.57}$$

This generalizes the notion of criticality that we discussed for deep linear networks in §3. Over two sections we will give a prescription for finding these critical initialization hyperparameters $(C_b, C_W)^{\text{critical}}$ for any nonlinear activation function.

5.2 Criticality for Scale-Invariant Activations

Now, let's extend our criticality analysis to *scale-invariant* activation functions by applying the formalism that we just developed. Recall from §2.2 that a scale-invariant activation function satisfies

$$\sigma(\lambda z) = \lambda \sigma(z)\,, \tag{5.58}$$

for any positive rescaling $\lambda > 0$, and always takes the form

$$\sigma(z) = \begin{cases} a_+ z\,, & z \geq 0\,, \\ a_- z\,, & z < 0\,. \end{cases} \tag{5.59}$$

As a reminder, this class of activations includes the `linear` activation – by setting $a_+ = a_- = 1$ – and the `ReLU` – by setting $a_+ = 1$ and $a_- = 0$.

These activation functions are particularly simple in that the criticality conditions (5.55), $\chi_\parallel\Big(K_{00}^{(\ell)}\Big) = \chi_\perp\Big(K_{00}^{(\ell)}\Big) = 1$, can be solved exactly. To start, we can easily compute $g(K)$, (5.49), which reduces to two Gaussian integrals on half the real line times an even polynomial, yielding

$$g(K) = A_2 K\,, \tag{5.60}$$

where we have introduced an activation-dependent constant

$$A_2 \equiv \frac{a_+^2 + a_-^2}{2}\,. \tag{5.61}$$

From (5.50), we see that we can find $\chi_\parallel(K)$ by differentiating this expression with respect to K and multiplying by C_W. Inspecting (5.51), we see that to get $\chi_\perp(K)$, we can perform two more simple Gaussian integrals on half the real line. Together, we find that both susceptibilities are equal and independent of $K_{00}^{(\ell)}$

$$\chi_\parallel\Big(K_{00}^{(\ell)}\Big) = \chi_\perp\Big(K_{00}^{(\ell)}\Big) = A_2 C_W \equiv \chi\,. \tag{5.62}$$

Lastly $h(K)$, (5.52), identically vanishes because it is a derivative of $\chi_\perp(K)$.

With all that, we can write the general kernel recursions (5.46), (5.47), and (5.48) for scale-invariant activations as

$$K_{00}^{(\ell+1)} = C_b + \chi K_{00}^{(\ell)}\,, \tag{5.63}$$

$$\delta K_{[1]}^{(\ell+1)} = \chi \delta K_{[1]}^{(\ell)}\,, \tag{5.64}$$

$$\delta\delta K_{[2]}^{(\ell+1)} = \chi \delta\delta K_{[2]}^{(\ell)}\,. \tag{5.65}$$

These are quite simple to solve. Just as the initialization hyperparameter C_W governed the exploding and vanishing kernel problem in §3.2, the constant susceptibility $\chi = A_2 C_W$ governs the same problem here:

- If $\chi > 1$, all quantities explode exponentially in ℓ towards a trivial fixed point at infinity.

- If $\chi < 1$, the fixed-point value of the kernel is given by $K^\star_{00} = \frac{C_b}{1-\chi}$ and all perturbations around the fixed point vanish exponentially with ℓ.

- If $C_W = 1/A_2$ and $C_b = 0$, then the network is at criticality. Not only does every perturbation stays constant,[11] but also any value of $K^\star_{00}$ serves as a nontrivial fixed point, i.e., there is a *line of nontrivial fixed points*.[12] In particular, the value of the fixed point is given by
$$K^\star_{00} = \frac{1}{A_2}\left(\frac{1}{n_0}\sum_{i=1}^{n_0} x^2_{i;0}\right) . \tag{5.66}$$

- If $C_W = 1/A_2$ and $C_b > 0$, then $\delta K^{(\ell)}_{[1]}$ and $\delta\delta K^{(\ell)}_{[2]}$ stay constant at this infinitesimal level of analysis. However, $K^{(\ell)}_{00}$ grows linearly towards a nontrivial fixed point at infinity, with the rate set by C_b. Since the kernel does not exhibit any exponential behavior, such a network is at criticality in a broad sense. This **semi-criticality** results in a *line of semi-critical initialization hyperparameters* parameterized by C_b in the hyperparameter plane spanned by (C_b, C_W).

In conclusion, this study generalizes the analysis carried out for deep linear networks in §3.2 and identifies
$$(C_b, C_W)^{\text{critical}} = \left(0, \frac{1}{A_2}\right) , \tag{5.67}$$
with $A_2 = (a_+^2 + a_-^2)/2$ as the critical initialization hyperparameters for scale-invariant activation functions.[13] For the `ReLU` activation function, this reproduces the *Kaiming initialization* $(C_b, C_W)^{\text{critical}} = (0, 2)$ [46].

---

[11] One caveat is in order. While the constancy of the preactivation norm $K^{(\ell)}_{[0]}$ and the parallel perturbation $K^{(\ell)}_{[1]}$ is exact, the constancy of $K^{(\ell)}_{[2]}$ is an artifact of our infinitesimal perturbation analysis. In fact, the finite-angle analysis of nonlinear scale-invariant activation functions in §5.5 describes how $K^{(\ell)}_{[2]}$ crosses over from near constancy for small ℓ to a power-law decay $\sim 1/\ell^2$ for large ℓ. In short, the preservation of the whole kernel matrix seen in §3.2 is a special property of the `linear` activation, and for nonlinear scale-invariant activation functions there is a slow power-law decay of some observables. This power-law behavior is quite benign compared to exponential behavior and is typical at criticality.

[12] For physicists, note that a similar line of fixed points often appears in scale-invariant field theories with exactly marginal deformations.

[13] We see here that our simplification of $C_b = 0$ for deep linear networks in §3 was completely warranted, *ex post facto*.

5.3 Universality beyond Scale-Invariant Activations

All of the activation functions treated in the last section shared a rather special property: scale invariance (5.58). This property gave rise to equal and kernel-independent parallel and perpendicular susceptibilities, $\chi_{\parallel}(K) = \chi$ and $\chi_{\perp}(K) = \chi$, all together enabling us to drastically simplify the criticality analysis for these activation functions.[14] Such an analysis showed that networks equipped with a scale-invariant activation function will behave similarly to each other under *representation group flow* at criticality.

In theoretical physics, systems at criticality that behave similarly under *renormalization group flow* are said to fall into the same **universality class**. The effective action describing such systems converge under the iterative coarse-graining procedure, such that at long-range scales these systems share the same underlying mathematical model or *effective theory*, independent of the microscopic details of the particular system. This phenomenon is known as **universality** [47].

This motivates the use of the same term, universality class, to characterize activation functions that share the same limiting behavior under representation group flow, thus furthering the connection between *RG flow and RG flow* that we began developing in §4.6. Activation functions that form a universality class will have an identical effective description after flowing through many layers, meaning that the effective theory describing the preactivation distribution becomes independent of the fine details of the particular activation function. The power of universality is that a *single* effective theory enables us to understand criticality for the many different activation functions within the same universality class.

Clearly, all the scale-invariant activation functions form a universality class. However, the simplifications that enabled us to easily analyze this *scale-invariant universality class*, e.g. the kernel-independence of the susceptibilities, do not hold for other activation functions. For activation functions such as the `sigmoid`, `tanh`, or `SWISH`, we'll need to develop a much more general algorithm to find critical initialization hyperparameters. In §5.3.1, we'll illustrate how this algorithm works, and then we'll analyze specific activation functions in §5.3.2, §5.3.3, and §5.3.4.

5.3.1 General Strategy

Let's start with some recollections. As discussed most recently in §5.1, for a generic choice of initialization hyperparameters C_b and C_W, the kernel recursion for a single-input x_0,

$$K_{00}^{(\ell+1)} = C_b + C_W g\left(K_{00}^{(\ell)}\right) , \tag{5.68}$$

admits a fixed-point solution satisfying

$$K_{00}^{\star} = C_b + C_W g\left(K_{00}^{\star}\right) , \tag{5.69}$$

[14] The kernel-independence property follows directly from the scale-invariance, as any dependence would have introduced a scale into the problem.

where the helper function

$$g\left(K\right) \equiv \left\langle \sigma(z)\sigma(z)\right\rangle_K \, , \tag{5.70}$$

is understood as a function of the kernel value K. Our goal is to find critical initialization hyperparameters whose associated fixed-point value $K^\star_{00} = K^\star_{00}\left(C_b, C_W\right)$ gives rise to $\chi_\parallel\left(K^\star_{00}\right) = \chi_\perp\left(K^\star_{00}\right) = 1$.

How do we actually find these critical values? Conceptually, the most obvious route – illustrated in Figure 5.1 for the `tanh` activation function – is the following procedure:

1. For each value of C_b and C_W, with $C_b \geq 0$ and $C_W \geq 0$, find a fixed-point value of the kernel $K^\star_{00} = K^\star_{00}\left(C_b, C_W\right)$, implicitly defined via $K^\star_{00} = C_b + C_W g_0\left(K^\star_{00}\right)$ with the constraint $K^\star_{00} \geq 0$.

2. With $K^\star_{00}\left(C_b, C_W\right)$, evaluate both $\chi_\parallel\left(K^\star_{00}\right)$ and $\chi_\perp\left(K^\star_{00}\right)$, scanning over values in the (C_b, C_W) plane until the criticality conditions $\chi_\parallel = 1$ and $\chi_\perp = 1$ are both met.

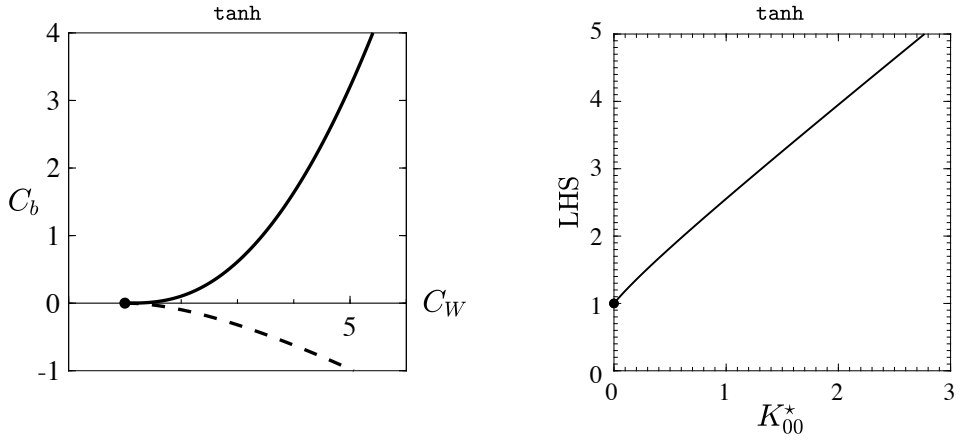

Figure 5.1: Two algorithms to pin down a nontrivial fixed point, illustrated here for the `tanh` activation function. **Left:** the lines defined by the conditions $\chi^\star_\perp = 1$ (solid) and $\chi^\star_\parallel = 1$ (dashed) are shown in the hyperparameter plane (C_W, C_b) for the `tanh` activation function. The intersection of these two lines gives the critical initialization hyperparameters $(C_W, C_b) = (1, 0)$. **Right:** the left-hand side of the condition (5.73) is plotted as a function of $K^\star_{00}$. The plotted line hits unity as $K^\star_{00} \to 0$.

This algorithm, however, is practically cumbersome to carry out for general activation functions, both numerically and analytically. In order to obtain a more implementation-friendly algorithm, let's reshuffle the logic a bit. First, note that for a candidate fixed-

point value $K^\star_{00}$, setting

$$C_W = \left[\left\langle \sigma'(z)\sigma'(z)\right\rangle_{K^\star_{00}}\right]^{-1}, \tag{5.71}$$

$$C_b = K^\star_{00} - \frac{\left\langle \sigma(z)\sigma(z)\right\rangle_{K^\star_{00}}}{\left\langle \sigma'(z)\sigma'(z)\right\rangle_{K^\star_{00}}}, \tag{5.72}$$

satisfies both the fixed-point equation $K^\star_{00} = C_b + C_W g_0\left(K^\star_{00}\right)$ as well as the first criticality condition $\chi_\perp\left(K^\star_{00}\right) = 1$. The second criticality condition $\chi_\parallel\left(K^\star_{00}\right) = 1$ then is tantamount to $\chi_\perp\left(K^\star_{00}\right)/\chi_\parallel\left(K^\star_{00}\right) = 1$, which is simply the following ratio of expectations

$$\left[\frac{2K^2\left\langle \sigma'(z)\sigma'(z)\right\rangle_K}{\left\langle \sigma(z)\sigma(z)\left(z^2-K\right)\right\rangle_K}\right]\Bigg|_{K=K^\star_{00}} = 1\,, \tag{5.73}$$

independent of the initialization hyperparameters C_W and C_b. Therefore, we can use the following simpler algorithm:

1. Scan over values of $K^\star_{00} \geq 0$ until (5.73) is satisfied.
2. Plug the resulting value of $K^\star_{00}$ into (5.71) and (5.72) to evaluate the critical initialization hyperparameters (and also make sure $C_b \geq 0$).

In Figure 5.1, the left-hand side of (5.73) is plotted as a function of $K^\star_{00}$ for the `tanh` activation function, which we see hits unity at $K^\star_{00} = 0$. Then, evaluating equations (5.71) and (5.72) in the limit $K^\star_{00} \to 0$ efficiently gives the critical initialization hyperparameters for `tanh`: $(C_W, C_b) = (1, 0)$.[15]

In passing, we note that scale-invariant activation functions trivially satisfy the condition (5.73) for any fixed-point value $K^\star_{00}$, since the susceptibilities are equal to the same kernel-independent constant, $\chi_\parallel(K) = \chi_\perp(K) = \chi$. It's easy to check that for this universality class, the above algorithm recovers the critical initialization hyperparameters (5.67) given in §5.2.

5.3.2 No Criticality: sigmoid, softplus, nonlinear monomials, etc.

For some activation functions, a nontrivial fixed point for the kernel does not exist. For example, consider the `sigmoid` activation function

$$\sigma(z) = \frac{1}{1+e^{-z}}\,. \tag{5.74}$$

The condition (5.73) is plotted for this activation in Figure 5.2. While this condition is satisfied at $K^\star_{00} = 0$, evaluating (5.72) in this limit yields $C_b = -\left(\frac{\sigma(0)}{\sigma'(0)}\right)^2 < 0$. Since the

[15] Even though the fixed-point value of the midpoint kernel is zero, this is a *nontrivial* fixed point. In particular, we will see in §5.3.3 that kernels with a nontrivial fixed point at $K^\star_{00} = 0$ form a *universality class*, characterized by a benign power-law decay in ℓ. In practice, the power-law behavior means that for any finite depth the kernel will remain finite.

variance of the bias cannot be negative, this is unphysical.[16] Thus, the `sigmoid` cannot be tuned to criticality and should not be used.[17]

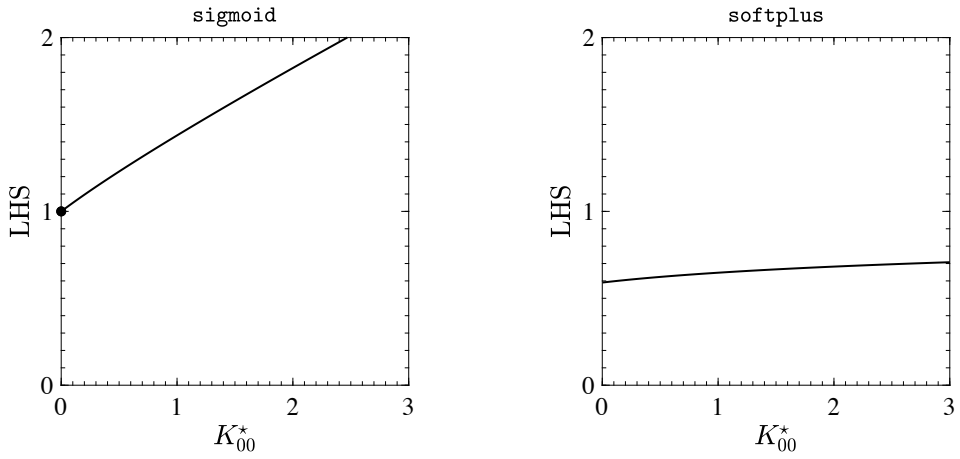

Figure 5.2: The left-hand side of the condition (5.73) is plotted as a function of $K^\star_{00}$ for the `sigmoid` activation function (left) and the `softplus` activation function (right). For the `sigmoid`, the plotted line hits unity as $K^\star_{00} \to 0$, but the associated critical initialization hyperparameters (C_b, C_W) are unphysical because $C_b < 0$. For the `softplus`, the plotted line does not hit unity. These activation functions cannot be tuned to criticality.

Next let's consider the `softplus` activation function

$$\sigma(z) = \log\left(1 + e^z\right) , \tag{5.75}$$

which, as a reminder, is a smooth approximation of the `ReLU`. Plotting the condition (5.73) in Figure 5.2, we see that it cannot be satisfied for any $K^\star_{00} \geq 0$. Thus, in contrast to the `ReLU`, the `softplus` cannot be tuned to criticality. This supports the lore in the community that the `ReLU` is superior to the `softplus`, despite their similarity and the `softplus`' smoothness.

As we will see in the next subsection, the real problem with these activation functions is that they do not cross zero at $z = 0$. There is an easy fix, namely, setting

$$\sigma(0) = 0 , \tag{5.76}$$

by an appropriate constant shift for each activation. With such a shift the `sigmoid` turns into the `tanh`, albeit with the preactivation and activation each scaled by a half. Such a scaled `tanh` indeed admits a critical initialization, which is easy to check after reading the discussion in the next subsection.

[16] The limiting value of $C_b = -\left(\frac{\sigma(0)}{\sigma'(0)}\right)^2$ hints that the conditions $\sigma(0) = 0$ and $\sigma'(0) \neq 0$ may be necessary constraints for an activation function to have a nontrivial fixed point.

[17] Similarly, as a non-smooth limit of a logistic function, the `perceptron` activation function is even worse and doesn't merit discussion.

With that in mind, let's see what happens for activation functions that cross zero nonlinearly. For simplicity, take any nonlinear monomial activation function

$$\sigma(z) = z^p\,, \quad p = 2, 3, 4, \ldots\,. \tag{5.77}$$

In this case, direct Gaussian integration translates the condition (5.73) into the constraint

$$\frac{p}{2p-1} = 1\,, \tag{5.78}$$

which cannot be satisfied for nonlinear monomials, since $p \neq 1$. Thus, such nonlinear monomials also shouldn't be used in deep networks. More importantly, in addition to $\sigma(0) = 0$, criticality seems to require the condition

$$\sigma'(0) \neq 0\,, \tag{5.79}$$

which we will investigate more generally in the next subsection.

The impossibility of criticality for all of the activation functions discussed in this subsection means that their use should be discouraged. While the problem is somewhat mitigated for shallow networks – since there are fewer layers for the exponential behavior to damage the signals – as networks become deeper and deeper, criticality becomes more and more essential.

5.3.3 $K^\star = 0$ Universality Class: tanh, sin, etc.

In §5.3.1, we learned through a numerical investigation that `tanh` has a nontrivial fixed point at $K^\star_{00} = 0$. In addition, in the last subsection §5.3.2, our analysis suggested that the conditions $\sigma(0) = 0$ and $\sigma'(0) \neq 0$ are important for any smooth activation to have a nontrivial fixed point.

In this subsection, we will connect these two observations. In particular, in the vicinity of $K^\star_{00} = 0$, we can analytically analyze the kernel recursions (5.46)–(5.48) by Taylor expanding around $K^\star_{00} = 0$ and directly integrating the Gaussian expectations. This analysis will show that the conditions $\sigma(0) = 0$ and $\sigma'(0) \neq 0$ are both necessary and sufficient for a smooth activation function to have a nontrivial fixed point at $K^\star_{00} = 0$, leading to the definition of our second universality class.

Let's use the following notation for the Taylor coefficients of any analytic activation function:

$$\sigma(z) = \sum_{p=0}^{\infty} \frac{\sigma_p}{p!} z^p\,. \tag{5.80}$$

Plugging this expansion into the definition of the helper function (5.70) and performing the Gaussian integral, we find

$$g(K) = \langle \sigma(z)\sigma(z) \rangle_K = \sigma_0^2 + \left(\sigma_1^2 + 2\sigma_0\sigma_2\right) K + O\left(K^2\right)\,. \tag{5.81}$$

From this we see that the fixed point of the recursion for the midpoint kernel

$$K^\star_{00} = C_b + C_W g(K^\star_{00})\,, \tag{5.82}$$

has a solution at $K^\star_{00} = 0$ if and only if $C_b = C_W \sigma_0^2 = 0$. Recalling that $C_W = 0$ violates the criticality conditions, we must pick $\sigma_0 = 0$. Henceforth we will assume that this choice has been made.

Continuing on with $\sigma_0 = 0$ and $C_b = 0$ in mind, inserting the expansion (5.80) into our expressions for the susceptibilities, (5.50) and (5.51), and performing the Gaussian integrals we find

$$C_W g(K) = \left(C_W \sigma_1^2\right)\left[K + a_1 K^2 + a_2 K^3 + O\left(K^4\right)\right] , \tag{5.83}$$

$$\chi_\parallel(K) = \left(C_W \sigma_1^2\right)\left[1 + 2a_1 K + 3a_2 K^2 + O\left(K^3\right)\right] , \tag{5.84}$$

$$\chi_\perp(K) = \left(C_W \sigma_1^2\right)\left[1 + b_1 K + O\left(K^2\right)\right] , \tag{5.85}$$

where here we have also expanded $g(K)$ to higher order in the kernel, and the coefficients a_1, a_2, and b_1 are given by the following combinations of Taylor coefficients of the activation function

$$a_1 \equiv \left(\frac{\sigma_3}{\sigma_1}\right) + \frac{3}{4}\left(\frac{\sigma_2}{\sigma_1}\right)^2 , \tag{5.86}$$

$$a_2 \equiv \frac{1}{4}\left(\frac{\sigma_5}{\sigma_1}\right) + \frac{5}{8}\left(\frac{\sigma_4}{\sigma_1}\right)\left(\frac{\sigma_2}{\sigma_1}\right) + \frac{5}{12}\left(\frac{\sigma_3}{\sigma_1}\right)^2 , \tag{5.87}$$

$$b_1 \equiv \left(\frac{\sigma_3}{\sigma_1}\right) + \left(\frac{\sigma_2}{\sigma_1}\right)^2 . \tag{5.88}$$

It's easy to check that, e.g., for `tanh` these coefficients take the following values $a_1 = -2$, $a_2 = 17/3$, $b_1 = -2$. Now, examining expansions (5.84) and (5.85), we see that to satisfy the criticality conditions $\chi_\parallel\left(K^\star_{00} = 0\right) = 1$ and $\chi_\perp\left(K^\star_{00} = 0\right) = 1$ we must set $C_W = 1/\sigma_1^2$. To ensure a finite variance, we also see that the activation function must have $\sigma_1 \neq 0$.

Thus, for any smooth activation function to have a nontrivial fixed point at $K^\star_{00} = 0$, it is necessary and sufficient that $\sigma(z)$ satisfy

$$\sigma_0 = 0 , \qquad \sigma_1 \neq 0 . \tag{5.89}$$

For such an activation, the critical initialization hyperparameters are then given by

$$(C_b, C_W)^{\text{critical}} = \left(0, \frac{1}{\sigma_1^2}\right) . \tag{5.90}$$

Just to emphasize this a bit, any activation with these conditions (5.89) initialized with (5.90) will have a nontrivial fixed point at $K^\star_{00} = 0$. The set of activation functions that vanish at the origin with a nonzero first derivative make up the $K^\star = 0$ *universality class*. The canonical class member is the `tanh` activation function, though there are obviously a very large number of members in this class, e.g. the `sin` activation function is a member too.

Having determined the critical initialization hyperparameters, let's now try to understand the behavior of the kernel for the $K^\star = 0$ universality class. We will see that

when tuned to criticality the activations satisfying (5.89) all behave similarly under RG flow, with the large-depth behavior of the kernel depending only on the first few Taylor coefficients of $\sigma(z)$.

Deep asymptotic analysis for the midpoint kernel

Recalling the expansion $K_{00}^{(\ell)} = K_{00}^{\star} + \Delta K_{00}^{(\ell)}$ around the fixed point and considering the expansion (5.83) for $g(K)$, the midpoint kernel recursion at $K_{00}^{\star} = 0$ criticality becomes

$$\Delta K_{00}^{(\ell+1)} = \Delta K_{00}^{(\ell)} + a_1 \left(\Delta K_{00}^{(\ell)}\right)^2 + a_2 \left(\Delta K_{00}^{(\ell)}\right)^3 + O\left(\left(\Delta K_{00}^{(\ell)}\right)^4\right) . \tag{5.91}$$

Since the whole point of criticality is to alleviate exponential behavior, we expect a gentler decay back to the $K_{00}^{(\ell)} = 0$ fixed point. With that in mind, let's plug a power-law ansatz $\Delta K_{00}^{(\ell)} \sim \left(\frac{1}{\ell}\right)^{p_0}$ into (5.91). Noting that $\left(\frac{1}{\ell+1}\right)^{p_0} = \frac{1}{\ell^{p_0}}\left[1 - \frac{p_0}{\ell} + O\left(\frac{1}{\ell^2}\right)\right]$ and matching the leading terms on both sides, we get a solution

$$\Delta K_{00}^{(\ell)} = \left[\frac{1}{(-a_1)}\right] \frac{1}{\ell} + \ldots . \tag{5.92}$$

Thus, the behavior at criticality is a mild power law decay, with a **critical exponent** $p_0 = 1$. Such an exponent is said to be *universal* for the $K^{\star} = 0$ universality class, since it is completely independent of the details of the particular activation function.

Importantly, for this asymptotic solution to be consistent, we must have $(-a_1) > 0$ to ensure the positivity of the kernel. If instead we had $(-a_1) < 0$, then the asymptotic solution (5.92) would be negative, making it invalid. In this case the fixed point would be unstable, exponentially repelling the kernel away from $K_{00}^{\star} = 0$.[18] We will see in the next subsection that `SWISH` and `GELU` activation functions exhibit such an instability near $K_{00}^{\star} = 0$.

Moreover, in the last subsection we suggested that an activation function that doesn't satisfy $\sigma(0) = 0$ could be potentially salvaged with a constant shift. In particular, perhaps the `softplus` could be saved by subtracting a constant $\log(2)$ so that $\sigma(0) = 0$? However, in this case we'd have $(-a_1) < 0$, and the kernel will get repelled from the only candidate nontrivial fixed point at $K_{00}^{\star} = 0$. And since $\chi_{\parallel}(K) > 1$ away from $K = 0$, the midpoint kernel will diverge exponentially. Thus, despite this attempt, we see that the `softplus` cannot be saved.

Returning to our solution (5.92), we can actually do quite a bit better than "..." for the subleading asymptotic analysis. As a first guess to improve our ansatz, let's include a subleading $1/\ell^2$ term in $\Delta K_{00}^{(\ell)}$. However, if we try to match terms on both sides of (5.91), we'd find that there's no way of canceling the $1/\ell^3$ terms. What we can do instead is to also add $\log(\ell)/\ell^2$ with an independent coefficient to our ansatz. This

[18] Generically, $(-a_1) < 0$ implies that $\chi_{\parallel} > 1$ away from $K_{00}^{\star} = 0$, which repels the midpoint kernel first with a power law and then exponentially. However, the semi-criticality that we discussed in §5.2 for scale-invariant activations was exceptional. For this universality class, $a_1 = 0$ and hence growth towards the fixed point at infinity is governed by a power law.

generates an additional $1/\ell^3$ term, allowing for a consistent solution. Generally for any of the observables $\mathcal{O}^{(\ell)}$ that we will consider, the correct **scaling ansatz** for the large-ℓ asymptotic expansion is of the form

$$\begin{aligned}\mathcal{O}^{(\ell)} &= \left(\frac{1}{\ell}\right)^{p_{\mathcal{O}}} \left[c_{0,0} + c_{1,1}\left(\frac{\log \ell}{\ell}\right) + c_{1,0}\left(\frac{1}{\ell}\right) + c_{2,2}\left(\frac{\log^2 \ell}{\ell^2}\right) + \ldots\right] \\ &= \left(\frac{1}{\ell}\right)^{p_{\mathcal{O}}} \left[\sum_{s=0}^{\infty}\sum_{q=0}^{s} c_{s,q}\left(\frac{\log^q \ell}{\ell^s}\right)\right], \end{aligned} \tag{5.93}$$

where the critical exponent $p_{\mathcal{O}}$ is expected to be universal for a given class, while the constants $c_{s,q}$ will depend on the details of a particular activation function. Carrying this process forward for $\mathcal{O}^{(\ell)} = \Delta K_{00}^{(\ell)}$, we can systematically determine the subleading behavior of the kernel perturbation as

$$\begin{aligned}\Delta K_{00}^{(\ell)} = &\left[\frac{1}{(-a_1)}\right]\frac{1}{\ell} + \left[\frac{-(a_2 - a_1^2)}{a_1^3}\right]\frac{\log\left(\frac{\ell}{\ell_0}\right)}{\ell^2} \\ &+ \left[\frac{-\left(a_2 - a_1^2\right)^2}{a_1^5}\right]\frac{\left[\log\left(\frac{\ell}{\ell_0}\right)\right]^2}{\ell^3} + \left[\frac{\left(a_2 - a_1^2\right)^2}{a_1^5}\right]\frac{\log\left(\frac{\ell}{\ell_0}\right)}{\ell^3} + O\left(\frac{1}{\ell^3}\right), \end{aligned} \tag{5.94}$$

and with enough effort this asymptotic expansion can be refined to arbitrary degree by including the higher-order corrections according to the scaling ansatz (5.93) described above.

Here, the constant ℓ_0 is undetermined by this large-ℓ asymptotic analysis and non-trivially depends on the input norm through

$$K_{00}^{(1)} = \frac{1}{\sigma_1^2}\frac{1}{n_0}\sum_{i=1}^{n_0} x_{i;0}^2\,, \tag{5.95}$$

which sets the initial condition (5.2) for the kernel recursion (5.1) when the rescaled weight variance is set to criticality, $C_W = 1/\sigma_1^2$. To get a sense of what this means, let's assume that $\chi_{\parallel}(K)$ is monotonically decreasing for $K \geq 0$ with $\chi_{\parallel}(0) = 1$ – as is true for `tanh` – and consider what happens when an input $x_{i;0}$ has a very large magnitude. Such a large-norm input will lead to a large value for the first-layer midpoint kernel, $K_{00}^{(1)} \gg 1$. In the range $0 < k_\sharp < K_{00}^{(\ell)}$, for some constant $k_\sharp$, the kernel $K_{00}^{(\ell)}$ will decay quicker than $\chi_{\parallel}(k_\sharp)^\ell$, with $\chi_{\parallel}(k_\sharp) < 1$, until it enters the power-law regime near $K_{00}^{\star} = 0$. The undetermined constant ℓ_0 is a remnant of this complicated crossover behavior, capturing the leading *data dependence* of the midpoint kernel.

Additionally, the asymptotic expansion for the midpoint kernel (5.94) has a nice interpretation under RG flow. While the critical exponent of the falloff $p_0 = 1$ is generic for the universality class, we see that the coefficients of the terms do depend on the details of the activation function, albeit only the first few Taylor coefficients. In fact, for larger and larger ℓ, the dependence is on fewer and fewer of the coefficients, with

the leading term only depending on a_1, (5.86). In this asymptotic limit, any activation function in the $K^\star = 0$ universality class with the same first three Taylor coefficients around zero will be completely indistinguishable. Thus, from the representation group flow perspective, one of the results of having a deeper network is to make the particular details of the activation function more and more irrelevant.

Lastly, let us note for all aspiring "activation designers" out there that we can engineer critical exponents other than $p_0 = 1$ by fine-tuning the Taylor coefficients of the activation function. For example, if we set $a_1 = 0$ by balancing σ_3 and σ_2, then the kernel approaches a $K^\star_{00} = 0$ nontrivial fixed point with a $1/\sqrt{\ell}$ power law decay so long as $(-a_2) > 0$. The need for such tuning indicates that the $\sim 1/\ell$ behavior is generic for activation functions in the $K^\star = 0$ universality class.[19]

Deep asymptotic analysis for parallel perturbations

Next, let's solve the $\delta K^{(\ell)}_{[1]}$ recursion for parallel perturbations. Plugging the expansion (5.84) for $\chi_\parallel(K)$ into the recursion (5.47), we get an algebraic equation

$$\delta K^{(\ell+1)}_{[1]} = \left[1 + 2a_1 \Delta K^{(\ell)}_{00} + 3a_2 \left(\Delta K^{(\ell)}_{00}\right)^2 + O\left(\left(\Delta K^{(\ell)}_{00}\right)^3\right)\right] \delta K^{(\ell)}_{[1]} \,. \tag{5.96}$$

Then, plugging in the large-ℓ solution for $\Delta K^{(\ell)}_{00}$ (5.94) and a large-ℓ asymptotic expansion for $\delta K^{(\ell)}_{[1]}$ based on our scaling ansatz (5.93), we can solve the resulting equation by matching the terms on both sides:

$$\delta K^{(\ell)}_{[1]} = \frac{\delta_\parallel}{\ell^2}\left[1 + \frac{2a_1\left(a_2 - a_1^2\right)}{a_1^3}\frac{\log\left(\frac{\ell}{\ell_0}\right)}{\ell} + O\left(\frac{1}{\ell}\right)\right] . \tag{5.97}$$

Inspecting our solution, we identify our second critical exponent for the $K^\star = 0$ universality class: $p_\parallel = 2$ corresponding to the $1/\ell^2$ falloff of $\delta K^{(\ell)}_{[1]}$. The particular value of this exponent is to be expected. As noted before, the parallel perturbation is just a difference of single-input kernels for two inputs with differing norms, $K^{(\ell)}_{[1]} = \left(K^{(\ell)}_{++} - K^{(\ell)}_{--}\right)/2$. The leading $1/\ell^2$ scaling occurs because the diagonal components $K^{(\ell)}_{++}$ and $K^{(\ell)}_{--}$ are governed by the same asymptotic behavior up to order $\log(\ell)/\ell^2$, including the same coefficients. Thus, the leading difference appears at order $1/\ell^2$, due to different input-dependent constants ℓ_+ and ℓ_- in expansions analogous to (5.94) for $K^{(\ell)}_{++}$ and $K^{(\ell)}_{--}$, with the undetermined constant $\delta_\parallel \propto \log(\ell_+/\ell_-)$. In this way, this constant explicitly carries the data dependence of the parallel perturbation.

[19] More precisely, we should have defined the $K^\star = 0$ universality class with the requirement $a_1 \neq 0$. This in turn would lead us to define a whole family of universality classes labeled by the degree of fine tuning of the a_1, a_2, etc., or equivalently labeled by the value of the critical exponent p_0.

Deep asymptotic analysis for perpendicular perturbations

Finally, let's conclude our analysis by solving the $\delta\delta K^{(\ell)}_{[2]}$ recursion for perpendicular perturbations. Let's begin by plugging the expansion (5.85) for $\chi_\perp(K)$ into the recursion (5.48). Since we want to focus on perpendicular perturbations with $\sum_{i=1}^{n_0} x_{i;0}\,\delta x_i = 0$, we will also turn off parallel perturbations by setting $\delta K^{(\ell)}_{[1]} = 0$. Putting this all together gives an algebraic equation

$$\delta\delta K^{(\ell+1)}_{[2]} = \left[1 + b_1 \Delta K^{(\ell)}_{00} + O\left(\left(\Delta K^{(\ell)}_{00}\right)^2\right)\right] \delta\delta K^{(\ell)}_{[2]} \,. \tag{5.98}$$

Plugging in the large-ℓ asymptotic solution for $\Delta K^{(\ell)}_{00}$ and solving the resulting equation with another large-ℓ asymptotic expansion for $\delta\delta K^{(\ell)}_{[2]}$ based on our scaling ansatz (5.93), we get

$$\delta\delta K^{(\ell)}_{[2]} = \frac{\delta^2}{\ell^{\frac{b_1}{a_1}}} \left[1 + \frac{b_1\left(a_2 - a_1^2\right)}{a_1^3} \frac{\log\left(\frac{\ell}{\ell_0}\right)}{\ell} + O\left(\frac{1}{\ell}\right)\right] , \tag{5.99}$$

where δ^2 is another unfixed constant undetermined by the large-ℓ solution, in this case related nontrivially to the magnitude of the difference of the inputs: $\sum_{i=1}^{n_0} \left(x_{i;+} - x_{i;-}\right)^2$. Here we see that the presumptive critical exponent, $p_\perp \equiv b_1/a_1$, depends mildly on the details of the activation function.

However, note that something nice happens for odd activation functions such as `tanh` and `sin`. In this case, we see from (5.86) and (5.88) that $a_1 = b_1$, giving us a bona fide critical exponent, $p_\perp = 1$, when restricting the universality class to odd activations. This means that perpendicular perturbations decay with the same power in ℓ as the midpoint kernel decays to the fixed point, $\sim 1/\ell$. Thus, at criticality the ratio $K^{(\ell)}_{[2]}/K^{(\ell)}_{[0]}$ is fixed at the leading order, preserving the angles between nearby perpendicular inputs. Importantly, this ensures that the relationship between input points is conserved under the RG flow, even if the signals propagate through a very deep network.

Furthermore, the milder falloff of the perpendicular perturbations suggests that they are in some sense more important than the parallel ones. This is because with enough depth the $K^{(\ell)}_{[1]}$ component will become subleading to the $K^{(\ell)}_{[0]}$ and the $K^{(\ell)}_{[2]}$ components, due to the $1/\ell^2$ scaling of the former compared to the $1/\ell$ scaling of the latter two. For this reason, we are going to ignore these parallel perturbations of the kernel going forward.

5.3.4 Half-Stable Universality Classes: SWISH, etc. and GELU, etc.

In this final subsection, we consider two other semi-popular activation functions in order to explore nontrivial fixed points away from zero, $K^\star_{00} \neq 0$.

- The `SWISH` activation function is defined as

$$\sigma(z) = \frac{z}{1 + e^{-z}} \,. \tag{5.100}$$

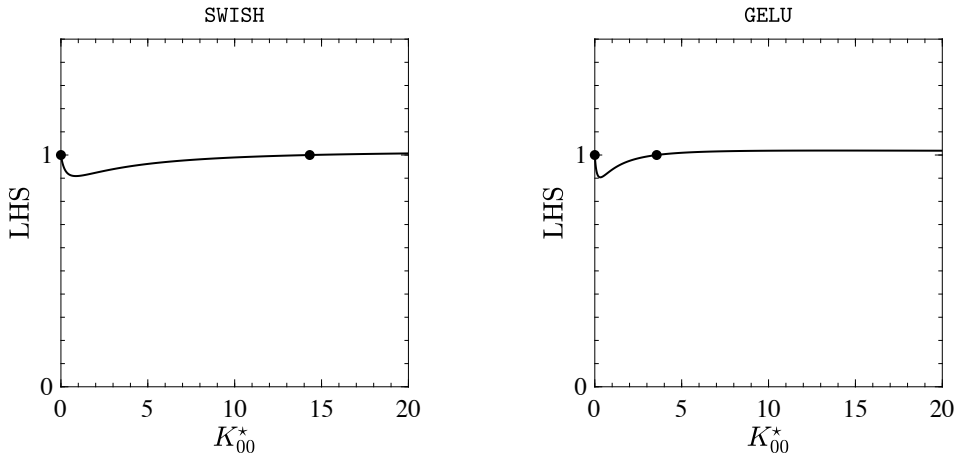

Figure 5.3: The left-hand side of the condition (5.73) is plotted as a function of $K^\star_{00}$ for the `SWISH` activation function (left) and the `GELU` activation function (right). For both activation functions, the plotted line hits unity (black dots) at $K^\star_{00} = 0$ as well as at a nonzero half-stable nontrivial fixed point $K^\star_{00} \neq 0$.

Similar to the intuition for the `softplus`, the `SWISH` is intended as a smooth version of the `ReLU`. Following our general algorithm in §5.3.1 for finding the critical initialization hyperparameters, we actually find two nontrivial fixed points for the kernel, see Figure 5.3. In particular, the condition (5.73) is met at $K^\star_{00} = 0$ with $(C_b, C_W) = (0, 4)$ and at $K^\star_{00} \approx 14.32017362$ with

$$(C_b, C_W) \approx (0.55514317, 1.98800468) \; . \tag{5.101}$$

For the $K^\star_{00} = 0$ nontrivial fixed point, one can check that $(-a_1) < 0$, and hence it's unstable. For the $K^\star_{00} \approx 14.3$ nontrivial fixed point, we expand the midpoint kernel recursion as $K^{(\ell)}_{00} = K^\star_{00} + \Delta K^{(\ell)}_{00}$, yielding

$$\Delta K^{(\ell+1)}_{00} = \Delta K^{(\ell)}_{00} + \tilde{a}_1 \left(\Delta K^{(\ell)}_{00}\right)^2 + O\left(\left(\Delta K^{(\ell)}_{00}\right)^3\right) , \tag{5.102}$$

with $(-\tilde{a}_1) \approx -2.84979219 \cdot 10^{-6}$.

Here, the large-ℓ asymptotic analysis around the finite fixed point is identical to the case of $K^\star_{00} = 0$, resulting in

$$\Delta K^{(\ell)}_{00} \sim \left[\frac{1}{(-\tilde{a}_1)}\right] \frac{1}{\ell} \, . \tag{5.103}$$

However, the interpretation is slightly different, because the fixed-point value $K^\star_{00} \approx 14.3$ is non-vanishing. In particular, this implies that when $K^{(\ell)}_{00} < K^\star_{00}$ the kernel is attracted to the fixed point, while when $K^{(\ell)}_{00} > K^\star_{00}$ the kernel is

repelled.[20] Hence, this fixed point is **half-stable**, and so the activation function is perhaps half-useful. In practice, however, $|\tilde{a}_1|$ is small enough that the `SWISH` behaves in an almost scale-invariant manner around $K_{00}^{(\ell)} \sim K_{00}^{\star} \approx 14.3$.

- The `GELU` activation is defined as

$$\sigma(z) = \frac{z}{2}\left[1 + \text{erf}\left(\frac{z}{\sqrt{2}}\right)\right] , \tag{5.104}$$

and as a reminder is another smoothed `ReLU`. Following our recipe for criticality, the condition (5.73) is again met twice, at $K_{00}^{\star} = 0$ with $(C_b, C_W) = (0, 4)$ and at $K_{00}^{\star} = \frac{3+\sqrt{17}}{2}$ with

$$(C_b, C_W) \approx (0.17292239, 1.98305826) , \tag{5.105}$$

see Figure 5.3. Similar to the `SWISH`, the fixed point at $K_{00}^{\star} = 0$ is unstable with $(-a_1) = -6/\pi < 0$, and the fixed point at $K_{00}^{\star} = \frac{3+\sqrt{17}}{2}$ is half-stable, in this case with $(-\tilde{a}_1) \approx (1.43626419) \cdot 10^{-4}$. Note that the sign of $\tilde{a}_1$ here differs from the sign for the `SWISH`. Thus, this time, when $K_{00}^{(\ell)} > K_{00}^{\star}$ the midpoint kernel is attracted to the fixed point, while when $K_{00}^{(\ell)} < K_{00}^{\star}$ it is repelled.[21] Note that the absolute value $|\tilde{a}_1|$ is bigger for the `GELU` than for the `SWISH`, meaning that it behaves less scale-invariantly and looks less like the `ReLU`.

Unlike the shifted `softplus` which admits only an unstable nontrivial fixed point at $K_{00}^{\star} = 0$, here the non-monotonicity of the `GELU` and `SWISH` activation functions gave rise to half-stable nontrivial fixed points at $K_{00}^{\star} \neq 0$. They are both representatives of *half-stable universality classes*. For both of these `ReLU`-like activations functions, the critical initialization hyperparameters for the $K_{00}^{\star} \neq 0$ half-stable nontrivial fixed points are very similar to the critical `ReLU` initialization $(C_b, C_W) = (0, 2)$; the activations in each of these classes really are just small perturbations of the `ReLU`. At the same time, the fact that there's a fixed point at a particular kernel value $K_{00}^{\star} \neq 0$ indicates – however weakly – the introduction of a particular scale. This is one way to see that these universality classes break scale invariance.

In summary, despite being `ReLU`-like and also smooth, both of the `SWISH` and `GELU` are inferior to the `ReLU` itself. If you want to use a smooth activation function, use `tanh`.

5.4 Fluctuations

Now that we fully understand how to tune infinite-width networks to criticality, let's back off this large-n limit to analyze the behavior of realistic networks. Specifically, we're

[20] With the half-critical initialization hyperparameters for the `SWISH` (5.101), there is a *trivial* fixed point at $K_{00}^{\star} \approx 14.5$ that exponentially attracts the midpoint kernel when $K_{00}^{(\ell)} > 14.3$.

[21] With the half-critical initialization hyperparameters for the `GELU` (5.105), there is a *trivial* fixed point at $K_{00}^{\star} \approx 3.2$ that exponentially attracts the midpoint kernel when $K_{00}^{(\ell)} < \frac{3+\sqrt{17}}{2} \approx 3.6$.

going to extend the finite-width analysis that we performed for deep linear networks in §3.3 to MLPs with nonlinear activation functions. Before diving in, let's review the motivation for carrying out such an analysis.

First, note that practitioners only use a single network rather than an ensemble of networks.[22] As we have discussed, sometimes a single instantiation will generically deviate from the mean. Therefore, in order to understand what *typically* happens in a single instantiation for an observable of interest, we have to compute not only the mean but also instantiation-to-instantiation fluctuations around the mean. As we explained in §3.3, such fluctuations are generically finite-width effects, controlled by the $1/n$-suppressed four-point vertex $V^{(\ell)}_{(\alpha_1\alpha_2)(\alpha_3\alpha_4)}$. If fluctuations are large, then a single instantiation can behave poorly, despite being sampled from an initialization distribution tune to criticality.

Second, we saw in §4.3 that the infinite-width ℓ-th-layer preactivation distribution factorizes as

$$p\Big(z^{(\ell)}_1, \ldots, z^{(\ell)}_{n_\ell} \Big| \mathcal{D}\Big) = p\Big(z^{(\ell)}_1 \Big| \mathcal{D}\Big) \cdots p\Big(z^{(\ell)}_{n_\ell} \Big| \mathcal{D}\Big) + O\Big(\frac{1}{n_\ell}\Big) \ , \tag{5.106}$$

where the distributions $p\Big(z^{(\ell)}_i \Big| \mathcal{D}\Big)$ on each neuron are given by statistically independent Gaussian distributions. (To emphasize the neural dependence here, we have included neural indices while suppressing sample indices.) Recalling our discussion of interactions and statistical independence in §1.3, this means that intralayer correlations among neurons are entirely finite-width phenomenon. Later, we will show how this lack of interactions connects to the fact that the representations of an infinite-width network cannot evolve during gradient-based learning. Thus, understanding these finite-width effects is a prerequisite to understanding how practical networks actually learn from input data.[23]

Third, finite-width corrections can modify the mean value of observables. As we saw in §4.5, at finite width all observables in principle receive an infinite series of subleading corrections. For instance, a possible finite-width NLO correction to the metric, $G^{\{1\}(\ell)}_{\alpha_1\alpha_2}$, can shift the infinite-width metric, $G^{\{0\}(\ell)}_{\alpha_1\alpha_2} \equiv K^{(\ell)}_{\alpha_1\alpha_2}$, a.k.a. the kernel. Such a finite-width correction could potentially ruin criticality, since our derivation of the critical initialization hyperparameters depended explicitly on the infinite-width fixed-point value of the kernel.[24]

There will be two main takeaways from this section.

- First, we will find that the leading finite-width fluctuations scale with the depth-to-width ratio of the network, L/n. We saw the importance of this *emergent scale* for the `linear` activation function in §3.3; here, we see that it persists very

[22] Actually in some cases practitioners can use ensembles of networks, though the computational cost of such models grows in proportion to the number of networks in the ensemble.

[23] We'll go into more detail about the role that these correlations play in the inductive bias of MLPs in §6 and then connect these interactions to representation learning in §11.

[24] In §5.4.1 we will show that the NLO metric $G^{\{1\}(\ell)}_{\alpha_1\alpha_2} = 0$ vanishes for the scale-invariant universality class, which is why we didn't discuss this type of correction for deep linear networks in §3.

generally for nonlinear activation functions. In the language of §4.6, this means that finite-width corrections are *relevant* under representation group flow and that deeper networks deviate more and more from the simple infinite-width limit. This emphasizes the importance of including such corrections when analyzing such networks and – taking into account the fact that overly deep networks suffer from overwhelming fluctuations – suggests that our perturbative effective theory works best in the regime where practical networks also work best.

- Second, the NLO metric $G^{\{1\}(\ell)}_{\alpha_1\alpha_2}$ is subdominant to the kernel $K^{(\ell)}_{\alpha_1\alpha_2}$ as long as an appropriate $O(1/n)$ correction is made to C_W. This means that the NLO metric vanishes in the interpolating limit – $n, L \to \infty$, with L/n fixed – and thus can safely be neglected for most wide networks of reasonable depths.

A single input, reloaded

In order to illustrate the important qualitative effects of finite width, we will again specialize to just a single input. The reason for this choice can be best understood by progressing through another twofold list:

(i) Once the two initialization hyperparameters, C_b and C_W, are tuned to criticality at leading order by the one- and two-input analysis of the kernel, the only additional tuning comes from the single-input analysis of the NLO metric $G^{\{1\}(\ell)}_{\alpha_1\alpha_2}$. Therefore, the multi-input solutions for the vertex and NLO metric do not add anything to the criticality analysis.

(ii) The most interesting part of the two-input vertex is a component that gives variance of the input-output Jacobian of the network. (As we described in footnote 9, the mean value of this Jacobian is captured by the $K^{(\ell)}_{[2]}$ component of the kernel.) However, the would-be analysis of this input-output variance will be subsumed by our analysis of the variance of the neural tangent kernel in §8, which more directly gives the variance of gradients relevant for training.

In the rest of this section we'll omit the $\alpha = 0$ sample indices, since such notation is unnecessarily cumbersome when considering only a single input. We'll also simplify things further by picking all the hidden-layer widths to be equal

$$n_1 = n_2 = \cdots = n_{L-1} \equiv n\,. \tag{5.107}$$

In addition to being a sensible choice, this means notationally that we don't have to carry around factors of $n_\ell/n_{\ell-1}$ everywhere. With these decisions in mind, the relevant

recursions from §4 become

$$K^{(\ell+1)} = C_b + C_W g\Big(K^{(\ell)}\Big)\,, \tag{5.108}$$

$$V^{(\ell+1)} = \chi_{\parallel}^2\Big(K^{(\ell)}\Big)\,V^{(\ell)} + C_W^2\left[\Big\langle \sigma^4(z)\Big\rangle_{K^{(\ell)}} - \Big\langle \sigma^2(z)\Big\rangle_{K^{(\ell)}}^2\right], \tag{5.109}$$

$$G^{\{1\}(\ell+1)} = \chi_{\parallel}\Big(K^{(\ell)}\Big)\,G^{\{1\}(\ell)} + \frac{1}{8}\,j\Big(K^{(\ell)}\Big)\,\frac{V^{(\ell)}}{\big(K^{(\ell)}\big)^2}\,, \tag{5.110}$$

where the helper function $g(K)$ and the parallel susceptibility $\chi_{\parallel}(K)$ were defined in (5.5) and (5.50), and we have defined another helper function

$$j(K) \equiv C_W \left\langle \sigma(z)\,\sigma(z)\left[\left(\frac{z^2}{K}\right)^2 - 6\left(\frac{z^2}{K}\right) + 3\right]\right\rangle_K . \tag{5.111}$$

These three recursions can be solved for each universality class by mirroring our bootstrap analysis of $K_{00}^{(\ell)}$, $\delta K_{[1]}^{(\ell)}$, $\delta\delta K_{[2]}^{(\ell)}$ in §5.2 and §5.3.

5.4.1 Fluctuations for the Scale-Invariant Universality Class

Recall from §5.2 that the scale-invariant universality class contains any activation function of the form

$$\sigma(z) = \begin{cases} a_+ z\,, & z \geq 0\,, \\ a_- z\,, & z < 0\,, \end{cases} \tag{5.112}$$

with the `ReLU` ($a_+ = 1, a_- = 0$) as the exemplar member to keep in mind. Also recall that for this class we evaluated the helper function as $g(K) = A_2 K$ and the parallel susceptibility as $\chi_{\parallel} = A_2 C_W \equiv \chi$, with the activation-dependent constant given by $A_2 \equiv (a_+^2 + a_-^2)/2$. The other terms in the new recursions (5.109) and (5.110) can similarly be evaluated by computing Gaussian integrals on the half-line, yielding

$$C_W^2\left[\Big\langle \sigma^4(z)\Big\rangle_K - \Big\langle \sigma^2(z)\Big\rangle_K^2\right] = C_W^2\left(3A_4 - A_2^2\right)K^2\,, \qquad j(K) = 0\,, \tag{5.113}$$

with a new activation-dependent constant

$$A_4 \equiv \frac{a_+^4 + a_-^4}{2}\,, \tag{5.114}$$

to pair with our other constant, A_2. With these expressions, the three recursions can be simplified as

$$K^{(\ell+1)} = C_b + \chi K^{(\ell)}\,, \tag{5.115}$$

$$V^{(\ell+1)} = \chi^2\left(\frac{3A_4}{A_2^2} - 1\right)\Big(K^{(\ell)}\Big)^2 + \chi^2\,V^{(\ell)}\,, \tag{5.116}$$

$$G^{\{1\}(\ell+1)} = \chi\,G^{\{1\}(\ell)}\,. \tag{5.117}$$

As a reminder, we already solved the kernel recursion in §5.2.

Things are now quite simple.

- First, remember from §4.1 that the first layer preactivation distribution is always exactly Gaussian, implying that the first-layer two-point correlator is simply given in terms of the first-layer kernel $K^{(1)}$ to all orders in n

$$\mathbb{E}\left[z_i^{(1)} z_j^{(1)}\right] = \delta_{ij} K^{(1)} \,. \tag{5.118}$$

  This means that the first-layer NLO metric must vanish $G^{\{1\}(1)} = 0$, and recursion (5.117) then tell us that the NLO metric will vanish in any subsequent layer. Thus, for activations in the scale-invariant universality class, we learn that the single-input metric does not get corrected at $O(1/n)$.

- Second, let's focus on criticality by setting $C_b = 0$ and $C_W = 1/A_2$. As discussed in §5.2, this setting of hyperparameters fixes the kernel to be an input-dependent layer-independent constant

$$K^{(\ell)} = K^\star \equiv \frac{1}{A_2}\left(\frac{1}{n_0}\sum_{i=1}^{n_0} x_i^2\right) \,. \tag{5.119}$$

  In particular, this means that the critical exponent for the single-input kernel is given by $p_0 = 0$. Setting $\chi = 1$ and substituting this expression into (5.116), we find a linearly growing solution for the four-point vertex

$$V^{(\ell)} = (\ell - 1)\left(\frac{3A_4}{A_2^2} - 1\right)(K^\star)^2 \,. \tag{5.120}$$

  By inspection, we identify another critical exponent for the scale-invariant universality class: assuming $V^{(\ell)} \sim (1/\ell)^{p_V}$, then $p_V = -1$. This exponent encodes the linear growth of the vertex under RG flow. Of particular note, the coefficient in front of (5.120) evaluates to $\left(\frac{3A_4}{A_2^2} - 1\right) = 2$ for `linear` activations in contrast to $=5$ for `ReLU` activations. Apparently the fluctuations in `ReLU` networks are significantly stronger than in deep linear networks. More generally, we conclude that the strength of such fluctuations is *not* universal.

- Third, let's revisit semi-criticality by setting $C_W = 1/A_2$, but setting the bias variance to an arbitrary positive constant, $C_b > 0$. As we saw in §5.2, in this case the kernel grows linearly towards a nontrivial fixed point at infinity, $K^{(\ell)} \sim \ell$, i.e., $p_0 = -1$. Plugging such a solution into the vertex recursion (5.116), we see that the four-point vertex grows cubicly $V^{(\ell)} \sim \ell^3$, i.e., $p_V = -3$. However, the appropriate dimensionless quantity – normalizing the vertex by the square of the kernel – still grows linearly in ℓ, i.e., $p_V - 2p_0 = -1$. Thus, even for semi-criticality the universal ℓ/n-scaling of the finite-width corrections is preserved.

5.4.2 Fluctuations for the $K^\star = 0$ Universality Class

Let's now consider the $K^\star = 0$ universality class. As a reminder, this class contains all smooth activation functions that satisfy $\sigma(0) = 0$ and $\sigma'(0) \neq 0$, with `tanh` as the

exemplar member to keep in mind. In §5.3.3, we determined that activations in this class have a nontrivial fixed point at $K^\star = 0$ and found that the associated critical initialization hyperparameters are given by $C_b = 0$ and $C_W = 1/\sigma_1^2$. For the rest of this subsection we will focus on such networks at criticality.

Mirroring our approach in §5.3.3 to solve the kernel recursions, we can evaluate the Gaussian expectations in the vertex recursion (5.109) and the NLO-metric recursion (5.110) by Taylor expanding the activation around $z = 0$ and explicitly computing the Gaussian integrals. Keeping in mind the criticality condition $C_W = 1/\sigma_1^2$, this gives the following expressions

$$\chi_\parallel(K) = 1 + 2a_1 K + 3a_2 K^2 + O\Big(K^3\Big) \,, \tag{5.121}$$

$$C_W^2 \left[\Big\langle \sigma^4(z)\Big\rangle_K - \Big\langle \sigma^2(z)\Big\rangle_K^2\right] = 2K^2 + \left(-52a_1 + 60b_1\right) K^3 + O\Big(K^4\Big) \,, \tag{5.122}$$

$$\frac{j(K)}{8K^2} = a_1 + 3a_2 K + O\Big(K^2\Big) \,. \tag{5.123}$$

Here, the expression for $\chi_\parallel(K)$ is simply reprinted from §5.3.3. Similarly, to limit the amount of time you have to flip back and forth, let us also reprint the large-ℓ asymptotic expansion of the kernel perturbation originally given by (5.94):

$$\begin{aligned}\Delta K^{(\ell)} = &\left[\frac{1}{(-a_1)}\right]\frac{1}{\ell} + \left[\frac{-(a_2 - a_1^2)}{a_1^3}\right]\frac{\log\left(\frac{\ell}{\ell_0}\right)}{\ell^2} \\ &+ \left[\frac{-\left(a_2 - a_1^2\right)^2}{a_1^5}\right]\frac{\left[\log\left(\frac{\ell}{\ell_0}\right)\right]^2}{\ell^3} + \left[\frac{\left(a_2 - a_1^2\right)^2}{a_1^5}\right]\frac{\log\left(\frac{\ell}{\ell_0}\right)}{\ell^3} + O\left(\frac{1}{\ell^3}\right) .\end{aligned} \tag{5.124}$$

Four-Point Vertex

Now, let's find a solution for the four-point vertex. Substituting in (5.121) and (5.122) into the single-input vertex recursion (5.109) gives an algebraic equation

$$\begin{aligned}V^{(\ell+1)} = V^{(\ell)}&\left[1 + 4a_1\Delta K^{(\ell)} + \left(6a_2 + 4a_1^2\right)\left(\Delta K^{(\ell)}\right)^2 + \ldots\right] \\ &+ 2\left(\Delta K^{(\ell)}\right)^2 + \left(-52a_1 + 60b_1\right)\left(\Delta K^{(\ell)}\right)^3 + \ldots \,.\end{aligned} \tag{5.125}$$

Using our scaling ansatz (5.93) for the large-ℓ asymptotic expansion

$$V^{(\ell)} = \left(\frac{1}{\ell}\right)^{p_V}\left[\# + \#'\frac{\log\ell}{\ell} + \frac{\#''}{\ell} + \ldots\right] \,, \tag{5.126}$$

and (5.124) for $\Delta K^{(\ell)}$ and then matching terms, we find

$$\begin{aligned}V^{(\ell)} = &\left[\frac{2}{3a_1^2}\right]\frac{1}{\ell} + \left[\frac{2(a_2 - a_1^2)}{3a_1^4}\right]\frac{\log\left(\frac{\ell}{\ell_0}\right)}{\ell^2} \\ &+ \left[\frac{5a_2 + a_1(82a_1 - 90b_1)}{3a_1^4}\right]\frac{1}{\ell^2} + O\left(\frac{\log^2(\ell)}{\ell^3}\right) ,\end{aligned} \tag{5.127}$$

where the constant scale ℓ_0 is same as the one in the $\Delta K^{(\ell)}$ expansion just above, again carrying the data dependence of the solution. We can also read off the critical exponent controlling the asymptotic falloff of the vertex for the $K^\star = 0$ universality class: $p_V = 1$.

Note that the value of the exponent $p_V = 1$ and the behavior of the four-point vertex $V^{(\ell)} \sim 1/\ell$ here is different from the value of the exponent $p_V = -1$ and the associated behavior $V^{(\ell)} \sim \ell$ that we found for the scale-invariant universality class. Also note that we saw this difference in the behavior of the kernel, $p_0 = 1$ vs. $p_0 = 0$, for the $K^\star = 0$ and scale-invariant classes, respectively. However, when instead considering the dimensionless quantity

$$\frac{V^{(\ell)}}{n\left(K^{(\ell)}\right)^2} \sim \frac{1}{n}\left(\frac{1}{\ell}\right)^{p_V - 2p_0} + \ldots \,, \tag{5.128}$$

we see that its scaling is consistent across both classes of activations:

$$p_V - 2p_0 = -1 \,. \tag{5.129}$$

Thus, this **scaling law** holds across different universality classes. As the normalized quantity (5.128) controls leading finite-width corrections to observables – this was discussed in detail in §3.3 – such a law means that these corrections are always *relevant* under representation group flow.

Concretely, the normalized vertex function is given by

$$\frac{V^{(\ell)}}{n\left(K^{(\ell)}\right)^2} = \left(\frac{3A_4}{A_2^2} - 1\right)\frac{\ell}{n} + O\left(\frac{1}{n}\right) \,, \tag{5.130}$$

for the scale-invariant universality class and

$$\frac{V^{(\ell)}}{n\left(K^{(\ell)}\right)^2} = \left(\frac{2}{3}\right)\frac{\ell}{n} + O\left(\frac{\log(\ell)}{n}\right) \,, \tag{5.131}$$

for the $K^\star = 0$ universality class. Of practical relevance, this means that `ReLU` networks and `tanh` networks of the same depth and width will have a mostly similar sensitivity to such corrections. However, the $O(1)$ coefficient of this quantity *does* depend on the particular activation function: $=5$ for `ReLU` and $=2/3$ for `tanh`. In Appendix A, we'll analyze this a bit more using tools from *information theory* and see how it can lead to a preferred aspect ratio, L/n, that is different for specific choices of activation functions.

NLO metric, bare

Next, let's solve the NLO-metric recursion (5.110). Substituting in (5.121) for $\chi_\parallel(K)$ and (5.123) for $j(K)$, we get

$$G^{\{1\}(\ell+1)} = G^{\{1\}(\ell)}\left[1 + 2a_1\Delta K^{(\ell)} + \ldots\right] + V^{(\ell)}\left[a_1 + 3a_2\Delta K^{(\ell)} + \ldots\right] \,. \tag{5.132}$$

As should now be familiar, let's assume a large-ℓ scaling ansatz

$$G^{\{1\}(\ell)} = \#\left(\frac{1}{\ell}\right)^{p_1} + \ldots \,, \tag{5.133}$$

with p_1 as the associated critical exponent. Bootstrapping (5.132) by substituting in our previous solutions – (5.124) for $\Delta K^{(\ell)}$ and (5.127) for $V^{(\ell)}$ – we then insert our ansatz for $G^{\{1\}(\ell)}$ (5.133) and match terms to find

$$G^{\{1\}(\ell)} = -\left[\frac{1}{3(-a_1)}\right] + O\Big(\frac{\log(\ell)}{\ell}\Big) \,. \tag{5.134}$$

This solution required us to set $p_1 = 0$ and gave a constant-in-ℓ leading contribution. Combining this with the kernel, we see that the finite-width-corrected two-point correlator

$$\mathbb{E}\left[z_i^{(\ell)} z_j^{(\ell)}\right] = \delta_{ij}\left[K^{(\ell)} + \frac{1}{n}G^{\{1\}(\ell)} + O\Big(1/n^2\Big)\right] , \tag{5.135}$$

is given by

$$K^{(\ell)} + \frac{1}{n}G^{\{1\}(\ell)} = \left[\frac{1}{(-a_1)}\right]\left(\frac{1}{\ell} - \frac{1}{3n}\right) + \ldots \,. \tag{5.136}$$

This result is to be contrasted with the scale-invariant universality class, where the NLO metric vanished identically.

For the NLO metric, the appropriate dimensionless quantity to consider is the ratio between the correction term and the infinite-width term in the two-point correlator (5.135)

$$\frac{1}{n}\frac{G^{\{1\}(\ell)}}{K^{(\ell)}} \sim \frac{1}{n}\left(\frac{1}{\ell}\right)^{p_1 - p_0} + \ldots \,, \tag{5.137}$$

with the exponent $p_1 - p_0$ controlling the relative importance of this NLO correction. In this case we see that $p_1 - p_0 = -1$, meaning that the above ratio scales with the depth-to-width ratio ℓ/n. This again illustrates the perturbative cutoff of our effective theory, $\ell \lesssim n$. However, in this particular case such a scaling turns out to be an artifact of not properly tuning the initialization hyperparameters C_W at finite width, as we will see next.

NLO metric, renormalized

In §5.3.3, we learned how to find the critical initialization hyperparameters for the $K^\star = 0$ universality class, fixing the hyperparameters C_b and C_W using the infinite-width recursions for the kernel components. However, in §4.5 we explained that all of the observables computed in a large-n expansion receive an infinite series of subleading corrections in $1/n$. This suggests that we should have allowed further fine-tuning of the initialization hyperparameters at criticality by considering large-n expansions

$$C_b^{(\ell)} = c_b^{(\ell)\{0\}} + \frac{c_b^{(\ell)\{1\}}}{n_{\ell-1}} + \frac{c_b^{(\ell)\{2\}}}{n_{\ell-1}^2} + \ldots \,, \tag{5.138}$$

$$C_W^{(\ell)} = c_W^{(\ell)\{0\}} + \frac{c_W^{(\ell)\{1\}}}{n_{\ell-1}} + \frac{c_W^{(\ell)\{2\}}}{n_{\ell-1}^2} + \ldots \,, \tag{5.139}$$

allowing us to adjust such hyperparameters order by order in $1/n$. Such an expansion could potentially give additional criticality conditions at each order in perturbation theory.

Considering the finite-width recursions (5.109) and (5.110), we see that such subleading tunings will not affect the leading order result for observables that depend on the four-point vertex, since the leading contributions to such observables are already at $O(1/n)$. However, these tunings do affect the solution for the NLO metric, because the NLO metric is itself subleading.

Concretely, there is an additional contribution to the NLO-metric recursion (5.110) coming from inserting the expansions (5.138) and (5.139) into the kernel recursion (5.108). The terms proportional to $c_b^{(\ell)\{1\}}$ or $c_W^{(\ell)\{1\}}$ are now subleading and thus contribute to the NLO metric recursion:

$$G^{\{1\}(\ell+1)} = \left[c_b^{(\ell)\{1\}} + c_W^{(\ell)\{1\}} g\left(K^{(\ell)}\right)\right] + \chi_{\parallel}^{(\ell)} G^{\{1\}(\ell)} + \frac{1}{8} j\left(K^{(\ell)}\right) \frac{V^{(\ell)}}{\left(K^{(\ell)}\right)^2} . \tag{5.140}$$

With this new "renormalized" perspective, we see that the analysis we did in the "bare" subsubsection before was just a particular choice of subleading corrections, $c_b^{(\ell)\{1\}} = c_W^{(\ell)\{1\}} = 0$. More generally, we really do have additional knobs to turn at this subleading order.

Substituting in (5.121) for $\chi_{\parallel}(K)$, (5.123) for $j(K)$, and (5.83) for $g(K)$, we find an algebraic equation

$$\begin{aligned} G^{\{1\}(\ell+1)} =& c_b^{(\ell)\{1\}} + c_W^{(\ell)\{1\}} \sigma_1^2 \left[K^{(\ell)} + a_1 \left(K^{(\ell)}\right)^2 + \ldots\right] \\ &+ G^{\{1\}(\ell)} \left[1 + 2a_1 K^{(\ell)} + \ldots\right] + V^{(\ell)} \left[a_1 + 3a_2 K^{(\ell)} + \ldots\right] . \end{aligned} \tag{5.141}$$

Plugging in the solution for the kernel (5.124) and vertex (5.127) – making sure to include the subleading-in-ℓ terms in both – inserting our large-ℓ scaling ansatz for $G^{\{1\}(\ell)}$ (5.133) and matching terms, we find that the tunings

$$c_b^{(\ell)\{1\}} = 0 , \qquad c_W^{(\ell)\{1\}} = \frac{2}{3} c_W^{(\ell)\{0\}} = \frac{2}{3\sigma_1^2} , \tag{5.142}$$

result in an asymptotically suppressed solution for the NLO metric

$$G^{\{1\}(\ell)} = \frac{2}{3}\left[\frac{3a_2 - a_1^2}{(-a_1)^3}\right] \frac{1}{\ell} + O\left(\frac{\log(\ell)}{\ell^2}\right) , \tag{5.143}$$

with a critical exponent $p_1 = 1$. Specifically, the tuning of $c_b^{(\ell)\{1\}}$ was required to suppress a linear growing $\sim \ell$ contribution, while the tuning of $c_W^{(\ell)\{1\}}$ cancels the constant $O(1)$ piece we found before in (5.134).

- In a sense, we got lucky before in our bare analysis: redoing this analysis without a $c_b^{(\ell)\{1\}} = 0$ tuning, the dimensionless ratio (5.137) grows quadratically with

depth and implies that the NLO metric dominates the kernel at $\ell \sim \sqrt{n}$. The fact that this subleading correction becomes parametrically large before reaching the ℓ/n perturbative cutoff of the effective theory really means that it's growing exponentially; $c_b^{(\ell)\{1\}} \neq 0$ eventually spoils criticality.

- In another sense, we got unlucky before: without the $c_W^{(\ell)\{1\}} = \frac{2}{3} c_W^{(\ell)\{0\}}$ tuning, the NLO metric is a leading ℓ/n correction. We see now that when properly handled, $p_1 - p_0 = 0$ and the dimensionless ratio (5.137) is $O(1)$ in depth at leading order. Such a correction is said to be *marginal* under the RG flow. This means that, while we'll always have to take into account the *relevant* four-point vertex corrections, we should be able to neglect NLO metric corrections as long as we respect the finite-width tunings (5.142).

Finally, the necessity of including such perturbative corrections to the critical initialization hyperparameters gives an alternate perspective on what can go wrong in practice when the network depth L approaches the network width n. Even for ensembles of such networks, the averaged quantities will require finer and finer tunings – e.g. (5.138) and (5.139) – in order for the effective theory describing the ensemble to reach criticality. For any reasonable value of n, such corrections will quickly become finer than the floating-point precision limit used to represent the hyperparameters. Thus, in practice it becomes essentially impossible to tune such large square networks to criticality.[25]

5.5 Finite-Angle Analysis for the Scale-Invariant Universality Class

In this section, we'll confront an important subtlety for activation functions in the scale-invariant universality class.

Recall that activation functions in this class take the form

$$\sigma(z) = \begin{cases} a_+ z \,, & z \geq 0 \,, \\ a_- z \,, & z < 0 \,, \end{cases} \tag{5.144}$$

and generally have a kink at the origin $z = 0$ (except for the *degenerate* member, the `linear` activation function, which has $a_+ = a_-$). In footnote 5 we first mentioned the existence of a subtlety after giving our δ expansions for the kernel (5.22)–(5.24), lightly questioning the validity of our expansions for non-smooth $\sigma(z)$. In footnote 11, we then described the main consequence of this subtlety. In particular, we claimed that for nonlinear scale-invariant activation functions the constant value – as a function of layer – of the perpendicular perturbation $\delta\delta K_{[2]}^{(\ell)}$ at criticality is an artifact of the perturbative δ expansion. To understand this claim properly, we'll need to work out the

[25] Note that this is an entirely different problem than the chaotic behavior at large depth that we described in §3.4 for deep linear networks. For the scale-invariant universality class, the NLO metric correction vanishes and therefore $c_W^{(\ell)\{1\}} = 0$.

full nonperturbative kernel recursion for activation functions in this class. This in turn will let us see the aforementioned correction to the asymptotic large-ℓ behavior of the kernel component $\delta\delta K^{(\ell)}_{[2]}$.

For this analysis, it will be sufficient to focus on two inputs $x_{i;\pm}$ of the same norm. In our previous setup, we assumed that both inputs were nearby such that their difference $\delta x_i \equiv (x_{i;+} - x_{i;-})$ was perturbatively small, $\delta x_i \ll 1$; here, we will make no assumptions at all about their difference. Given the symmetries of the network evolution, the individual norms of the two preactivations corresponding to these inputs will also be equal:

$$K_d^{(\ell)} \equiv \mathbb{E}\left[\frac{1}{n_\ell}\sum_{i=1}^{n_\ell}\left(z_{i;+}^{(\ell)}\right)^2\right] = \mathbb{E}\left[\frac{1}{n_\ell}\sum_{i=1}^{n_\ell}\left(z_{i;-}^{(\ell)}\right)^2\right] . \tag{5.145}$$

Geometrically this means that our preactivations live together on an n_ℓ-dimensional sphere with radius $\sqrt{n_\ell K_d^{(\ell)}}$, and algebraically this means that the parallel component vanishes $K^{(\ell)}_{[1]} = 0$, cf. (5.18). Going forward, we will call $K_d^{(\ell)}$ the **diagonal kernel**.[26]

The remaining dynamical variable is the polar angle between the preactivations. Therefore, we can decompose the two-input kernel matrix with the following parameterization:

$$K^{(\ell)}_{\alpha_1\alpha_2} = \begin{pmatrix} K^{(\ell)}_{++} & K^{(\ell)}_{+-} \\ K^{(\ell)}_{-+} & K^{(\ell)}_{--} \end{pmatrix} = K_d^{(\ell)} \begin{pmatrix} 1 & \cos\left(\psi^{(\ell)}\right) \\ \cos\left(\psi^{(\ell)}\right) & 1 \end{pmatrix} , \qquad \psi^{(\ell)} \in [0,\pi] . \tag{5.146}$$

The polar angle $\psi^{(\ell)}$ ranges from 0 – where the preactivations are coincident as $z_{i;+} = z_{i;-}$, making the kernel matrix degenerate – to π – where they're anti-correlated as $z_{i;+} = -z_{i;-}$. So far all we've done is fixed the norm of our two inputs to be equal and decomposed the kernel into a particular choice of coordinates; such a choice and parameterization can be applied to the analysis of any activation function. We'll now specialize to scale-invariant activation functions for which class it's possible to derive a nonperturbative recursion for the polar angle.

RG flow of the polar angle

The diagonal kernel follows the by-now familiar recursion for the single-input kernel (5.4)

$$K_d^{(\ell+1)} = C_b + C_W\, g\left(K_d^{(\ell)}\right) = C_b + A_2 C_W K_d^{(\ell)} , \tag{5.147}$$

where on the right-hand side we plugged in the explicit details for the scale-invariant universality class (5.63) and recalled $A_2 \equiv \left(a_+^2 + a_-^2\right)/2$. This part of the analysis carries

[26] Perturbatively, the *diagonal kernel* $K_d^{(\ell)}$ is equal to the *midpoint kernel* $K_{00}^{(\ell)}$ – the kernel for the midpoint input $x_{i;0} \equiv (x_{i;+} + x_{i;-})/2$ – at leading order in the δ expansion, cf. (5.22)–(5.24). Nonperturbatively, these two kernels are very different. To see this most vividly, consider two antipodal inputs $x_{i;+} = -x_{i;-}$. Then, the midpoint input is the zero vector $x_{i;0} = 0$, and the midpoint kernel in the first layer is given by $K_{00}^{(1)} = C_b^{(1)}$. In contrast, the diagonal kernel is given by either of $K_d^{(1)} = C_b^{(1)} + (C_W^{(1)}/n_0)\sum_{i=1}^{n_0} x_{i;\pm}^2$.

over from §5.2. We recall here that we can readily solve the recursion for any choice of initialization hyperparameters, and in particular criticality is attained by setting $C_b = 0$ and $A_2 C_W = 1$, where the diagonal kernel stays exactly constant: $K_d^{(\ell)} = K_d^{(1)} \equiv K_d^\star$.

With the evolution of the magnitude determined, we now need to find a recursion for the polar angle $\psi^{(\ell)}$. Plugging our new decomposition (5.146) into the full kernel recursion (5.1), the off-diagonal component of the recursion becomes

$$K_d^{(\ell+1)} \cos\Big(\psi^{(\ell+1)}\Big) = C_b + C_W \left\langle \sigma(z_+)\, \sigma(z_-) \right\rangle_{K^{(\ell)}} . \tag{5.148}$$

In this parameterization, the Gaussian expectation reads

$$\left\langle \sigma(z_+)\, \sigma(z_-) \right\rangle_{K^{(\ell)}} \equiv \frac{\int dz_+ dz_-\ \sigma(z_+)\, \sigma(z_-)\, e^{-\frac{1}{2}\sum_{\alpha_1,\alpha_2=\pm} K_{(\ell)}^{\alpha_1\alpha_2} z_{\alpha_1} z_{\alpha_2}}}{2\pi K_d^{(\ell)} \sin\big(\psi^{(\ell)}\big)} , \tag{5.149}$$

where the denominator comes from evaluating the determinant $\sqrt{\left|2\pi K^{(\ell)}\right|}$. To make further progress, we need to evaluate this painful integral.

Before working out the general case, let's focus on the `ReLU`. Setting $a_+ = 1$ and $a_- = 0$, we see that the argument of the Gaussian expectation is given by $\sigma(z_+)\,\sigma(z_-) = z_+ z_-$ when $z_+ > 0$ and $z_- > 0$ and vanishes otherwise. This means that the Gaussian expectation (5.149) is concentrated entirely in the first quadrant. In addition, noting that the integrand is invariant under parity $(z_+, z_-) \to (-z_+, -z_-)$, we can niftily substitute the integral over the first quadrant for half the integral over the first and third quadrants. This lets us rewrite the above Gaussian expectation as

$$\left\langle \sigma(z_+)\, \sigma(z_-) \right\rangle_{K^{(\ell)}} = \frac{\frac{1}{2}\int dz_+ dz_-\big|_{z_+ z_- > 0}\ z_+ z_-\, e^{-\frac{1}{2}\sum_{\alpha_1,\alpha_2=\pm} K_{(\ell)}^{\alpha_1\alpha_2} z_{\alpha_1} z_{\alpha_2}}}{2\pi K_d^{(\ell)} \sin\big(\psi^{(\ell)}\big)} . \tag{5.150}$$

The above actually turns out to be the only nifty step of the derivation; everything else is just a Herculean sequence of coordinate changes.

There are three coordinate changes in said sequence:

$$\begin{aligned} z_\pm =& \frac{u \pm w}{\sqrt{2}} \\ =& \sqrt{\frac{K_d^{(\ell)}\left[1 + \cos\big(\psi^{(\ell)}\big)\right]}{2}}\ x \pm \sqrt{\frac{K_d^{(\ell)}\left[1 - \cos\big(\psi^{(\ell)}\big)\right]}{2}}\ y \\ =& \sqrt{\frac{K_d^{(\ell)}\left[1 + \cos\big(\psi^{(\ell)}\big)\right]}{2}}\ r\cos(\phi) \pm \sqrt{\frac{K_d^{(\ell)}\left[1 - \cos\big(\psi^{(\ell)}\big)\right]}{2}}\ r\sin(\phi) . \end{aligned} \tag{5.151}$$

The first one diagonalizes the kernel so that the distribution factorizes $p(z_+, z_-) = p(u)p(w)$, the second one normalizes the coordinates with the kernel's eigenvalues, and

the last one exchanges Cartesian coordinates for polar coordinates.[27] Accordingly, this lets us rewrite the sum in the exponential in (5.150) as

$$\sum_{\alpha_1,\alpha_2=\pm} K_{(\ell)}^{\alpha_1\alpha_2} z_{\alpha_1} z_{\alpha_2} = \frac{u^2}{K_d^{(\ell)}\,[1+\cos(\psi^{(\ell)})]} + \frac{w^2}{K_d^{(\ell)}\,[1-\cos(\psi^{(\ell)})]} = x^2+y^2 = r^2\,, \tag{5.152}$$

while the product in the integrand becomes

$$z_+ z_- = \frac{K_d^{(\ell)} r^2}{2}\left[\cos(2\phi)+\cos\left(\psi^{(\ell)}\right)\right], \tag{5.153}$$

and the integral measure transforms as

$$dz_+ dz_- = K_d^{(\ell)} \sin\left(\psi^{(\ell)}\right) r\, dr\, d\phi\,. \tag{5.154}$$

Substituting (5.152)–(5.154) back into the Gaussian expectation (5.150), we get

$$\langle \sigma(z_+)\,\sigma(z_-)\rangle_{K^{(\ell)}} = \frac{K_d^{(\ell)}}{8\pi}\left[\int_0^\infty dr\; r^3 e^{-\frac{r^2}{2}}\right]\int_0^{2\pi} d\phi\Big|_{\cos(2\phi)+\cos\left(\psi^{(\ell)}\right)>0}\left[\cos(2\phi)+\cos\left(\psi^{(\ell)}\right)\right]. \tag{5.155}$$

The rest is now relatively straightforward. The radial integral can be evaluated by another change of the coordinate $s = r^2/2$:

$$\int_0^\infty dr\; r^3 e^{-\frac{r^2}{2}} = \int_0^\infty ds\; 2s\, e^{-s} = \Big[-2e^{-s}-2s\,e^{-s}\Big]\Big|_0^\infty = 2\,. \tag{5.156}$$

For the angle integral, note that any function of $\cos(2\phi)$ gives the same contribution from the four intervals $\widetilde{\phi} \equiv 2\phi \in [0,\pi],[\pi,2\pi],[2\pi,3\pi],[3\pi,4\pi]$. Further, within that first interval the constraint $\cos\left(\widetilde{\phi}\right) > -\cos\left(\psi^{(\ell)}\right)$ can be simply expressed as $\widetilde{\phi} < \pi - \psi^{(\ell)}$. Together, this lets us write

$$\begin{aligned}
&\int_0^{2\pi} d\phi\Big|_{\cos(2\phi)+\cos\left(\psi^{(\ell)}\right)>0}\left[\cos(2\phi)+\cos\left(\psi^{(\ell)}\right)\right] \qquad (5.157)\\
=&4\int_0^{\pi}\frac{d\widetilde{\phi}}{2}\Big|_{\cos(\widetilde{\phi})+\cos\left(\psi^{(\ell)}\right)>0}\left[\cos(\widetilde{\phi})+\cos\left(\psi^{(\ell)}\right)\right]\\
=&2\int_0^{\pi-\psi^{(\ell)}} d\widetilde{\phi}\left[\cos\left(\widetilde{\phi}\right)+\cos\left(\psi^{(\ell)}\right)\right] = 2\sin\left(\psi^{(\ell)}\right)+2\left(\pi-\psi^{(\ell)}\right)\cos\left(\psi^{(\ell)}\right).
\end{aligned}$$

[27] Unlike the perturbative calculations in (5.33) and (5.34), the diagonalization and normalization here are nonperturbatively exact. To reflect more on this, while we can always change coordinates as (5.151), we used the details of the `ReLU` in going from (5.149) to (5.150), establishing both the restricted domain of integration and the simplified form of the integrand, $\sigma(z_+)\,\sigma(z_-) \to z_+ z_-$, within that domain. For a general activation function, the resulting integral in the new coordinates (5.151) would still be difficult to evaluate, and we would have to resort to a perturbative expansion in $\psi^{(\ell)}$, ultimately analogous to the δ expansion, in order to make progress.

Inserting (5.156) and (5.157) into (5.155), we finally arrive at an expression for the Gaussian expectation of `ReLU` activations:

$$\langle \sigma(z_+)\, \sigma(z_-) \rangle_{K^{(\ell)}} = \frac{K_d^{(\ell)}}{2\pi} \left[\sin\!\left(\psi^{(\ell)}\right) + \left(\pi - \psi^{(\ell)}\right) \cos\!\left(\psi^{(\ell)}\right) \right] . \tag{5.158}$$

Now, let's work out the painful integral (5.149) for an arbitrary scale-invariant activation function (5.144). In general, there are contributions from the first quadrant proportional to a_+^2 and similar contributions from the third quadrant proportional to a_-^2, in both cases with the constraint $z_+ z_- > 0$ after our nifty trick. Then, there are also contributions from the second and fourth quadrants, both proportional to $a_+ a_-$ and with the constraint $z_+ z_- < 0$. Following a very similar sequence of steps as we did before for the `ReLU`, we can evaluate the Gaussian expectation (5.149) as

$$\begin{aligned} \langle \sigma(z_+)\, \sigma(z_-) \rangle_{K^{(\ell)}} &= \frac{K_d^{(\ell)}}{2\pi} (a_+^2 + a_-^2) \int_0^{\pi - \psi^{(\ell)}} d\widetilde{\phi} \left[\cos\!\left(\widetilde{\phi}\right) + \cos\!\left(\psi^{(\ell)}\right) \right] \\ &\quad + \frac{K_d^{(\ell)}}{2\pi} (2 a_+ a_-) \int_{\pi - \psi^{(\ell)}}^{\pi} d\widetilde{\phi} \left[\cos\!\left(\widetilde{\phi}\right) + \cos\!\left(\psi^{(\ell)}\right) \right] \\ &= \frac{K_d^{(\ell)}}{2\pi} (a_+ - a_-)^2 \left[\sin\!\left(\psi^{(\ell)}\right) - \psi^{(\ell)} \cos\!\left(\psi^{(\ell)}\right) \right] \\ &\quad + \left(\frac{a_+^2 + a_-^2}{2} \right) K_d^{(\ell)} \cos\!\left(\psi^{(\ell)}\right) . \end{aligned} \tag{5.159}$$

The full nonperturbative recursion for the off-diagonal part of the kernel (5.148) thus evaluates to

$$\begin{aligned} & K_d^{(\ell+1)} \cos\!\left(\psi^{(\ell+1)}\right) \\ =& C_b + C_W \left\{ \frac{K_d^{(\ell)}}{2\pi} (a_+ - a_-)^2 \left[\sin\!\left(\psi^{(\ell)}\right) - \psi^{(\ell)} \cos\!\left(\psi^{(\ell)}\right) \right] + \left(\frac{a_+^2 + a_-^2}{2} \right) K_d^{(\ell)} \cos\!\left(\psi^{(\ell)}\right) \right\} . \end{aligned} \tag{5.160}$$

One thing we notice here is that even though we evaluated the Gaussian expectation, we'll still have to deal with the fact the recursion is highly nonlinear in $\psi^{(\ell+1)}$.

While you're here and we have your attention, let's record the result for one additional nonperturbative Gaussian expectation for the scale-invariant universality class: $\langle \sigma'(z_+)\, \sigma'(z_-) \rangle_{K^{(\ell)}}$. The integral here is much simpler to evaluate than the undifferentiated one above since in each quadrant the argument of the expectation, $\sigma'(z_+)\, \sigma'(z_-)$, is constant. Following otherwise the same set of steps as above, in this case we find

$$\begin{aligned} \left\langle \sigma'(z_+)\, \sigma'(z_-) \right\rangle_{K^{(\ell)}} =& \frac{(a_+^2 + a_-^2)}{4\pi} \left[\int_0^\infty dr\; r e^{-\frac{r^2}{2}} \right] \int_0^{2\pi} d\phi \Big|_{\cos(2\phi) + \cos\left(\psi^{(\ell)}\right) > 0} \\ &+ \frac{2 a_+ a_-}{4\pi} \left[\int_0^\infty dr\; r e^{-\frac{r^2}{2}} \right] \int_0^{2\pi} d\phi \Big|_{\cos(2\phi) + \cos\left(\psi^{(\ell)}\right) < 0} \\ =& \left(\frac{a_+^2 + a_-^2}{2} \right) - \frac{\psi^{(\ell)}}{2\pi} (a_+ - a_-)^2 . \end{aligned} \tag{5.161}$$

We guess you guys aren't ready for that yet. But your future-selves are gonna love it.[28]

Criticality analysis of the polar angle

Having evaluated the recursion, let's now tune to criticality and work out the correct large-ℓ asymptotic behavior of the polar angle $\psi^{(\ell)}$. Working at scale-invariant criticality, with $C_b = 0$ and $A_2 C_W = 1$, and where the diagonal kernel is constant as $K_d^{(\ell)} = K_d^\star$, the off-diagonal recursion (5.160) simplifies to a decoupled recursion for the polar angle,

$$\cos\Big(\psi^{(\ell+1)}\Big) = \cos\Big(\psi^{(\ell)}\Big) + \rho\left[\sin\Big(\psi^{(\ell)}\Big) - \psi^{(\ell)}\cos\Big(\psi^{(\ell)}\Big)\right] . \tag{5.162}$$

Here, it was convenient to define a new constant,

$$\rho \equiv \frac{1}{\pi}\frac{(a_+ - a_-)^2}{(a_+^2 + a_-^2)} , \tag{5.163}$$

that encapsulates all of the details of the specific scale-invariant activation function. Roughly, ρ is a dimensionless measure of the kinkiness of the activation function at the origin, equal to zero for the `linear` activation function and $1/\pi$ for the `ReLU`. We see right away that the polar angle is exactly preserved for, and only for, $\rho = 0$. In particular, the preservation of the full two-input kernel matrix that we saw for the `linear` activation function in §3.2 doesn't extend to any other member of the universality class.

In order to determine the large-ℓ behavior of the polar angle $\psi^{(\ell)}$, we need a way to analyze the recursion (5.162). As we've been emphasizing, our main tool for analyzing such a nonlinear recursion is to find a fixed point and then linearize around it.[29] By inspection of the recursion, it's clear that $\psi^\star = 0$ is a fixed point. Thus, we should focus in on the small-angle regime: $\psi^{(\ell)} \ll 1$.

Taylor expanding the trigonometric functions in the recursion (5.162) around a vanishing polar angle, the linearized recursion becomes

$$\psi^{(\ell+1)} = \psi^{(\ell)}\sqrt{1 - \frac{2\rho}{3}\psi^{(\ell)} + O(\psi^2)} = \psi^{(\ell)} - \frac{\rho}{3}\Big(\psi^{(\ell)}\Big)^2 + O\Big(\psi^3\Big) . \tag{5.164}$$

To solve this recursion, we can use our scaling ansatz (5.93), which here reads

$$\psi^{(\ell)} = \left(\frac{1}{\ell}\right)^{p_\psi}\left[c_{0,0} + O\Big(\frac{\log\ell}{\ell}\Big)\right] , \tag{5.165}$$

with the critical exponent p_ψ governing the decay of the polar angle. Plugging this ansatz into our recursion (5.164) and matching the terms on both sides of the equation, we find a solution:

$$\psi^{(\ell)} = \left(\frac{3}{\rho}\right)\frac{1}{\ell} + O\Big(\frac{\log\ell}{\ell^2}\Big) . \tag{5.166}$$

[28] This result will turn out to be really useful in §10.3 when we investigate generalization error for the scale-invariant universality class at infinite width.

[29] Since we already nonperturbatively evaluated the Gaussian expectation (5.159) and fully took into account the lack of smoothness of the activation function – with the constant ρ (5.163) characterizing its kinkiness – at this point it's completely safe to employ a perturbative expansion.

From this we can read off the critical exponent, $p_\psi = 1$, which is universal excepting the degenerate `linear` limit of $\rho = 0$, for which we instead have $p_\psi = 0$.

In order to recast this result in the language of the rest of this chapter, let's project the two-input kernel (5.146) into the $\gamma^{[a]}_{\alpha\beta}$ representation using (5.20) and then insert (5.166):

$$K^{(\ell)}_{[0]} = K^{(\ell)}_d \left[\frac{1+\cos\left(\psi^{(\ell)}\right)}{2}\right] = K^\star_d + O\left(\frac{1}{\ell^2}\right) , \tag{5.167}$$

$$K^{(\ell)}_{[2]} = K^{(\ell)}_d \left[\frac{1-\cos\left(\psi^{(\ell)}\right)}{2}\right] = K^\star_d \left(\frac{9}{4\rho^2}\right)\frac{1}{\ell^2} + O\left(\frac{\log \ell}{\ell^3}\right) . \tag{5.168}$$

These solutions form the basis of what we claimed earlier in footnote 11. In particular, the perpendicular perturbation $K^{(\ell)}_{[2]}$ crosses over from being nearly constant for small depth $\ell \ll \ell_{\text{cross}}$ to power-law decaying $\sim 1/\ell^2$ for large depth $\ell \gg \ell_{\text{cross}}$.[30] This implies that the true critical exponent for scale-invariant perpendicular perturbations is $p_\perp = 2$.

Here, the crossover scale ℓ_{cross} is approximately given by

$$\ell_{\text{cross}} \sim \frac{3}{\rho\psi^{(\ell=1)}} \sim \frac{3}{2\rho}\sqrt{\frac{K^{(\ell=1)}_{[0]}}{K^{(\ell=1)}_{[2]}}} . \tag{5.169}$$

We get this by equating the small-depth constant answer, set by the first-layer condition, with the large-ℓ asymptotic answer given by (5.166); on the right-hand side of (5.169) we further wrote the polar angle $\psi^{(\ell=1)}$ in terms of the kernel components using (5.167) and (5.168). What we see is that the smaller this initial angle $\psi^{(\ell=1)}$ is – meaning that the closer the two inputs are to each other – the longer our original constant solution to the naive perpendicular recursion (5.65) is valid, and the longer it takes for the power-law regime to kick in.

The discussion above explains why our δ expansion failed to see the crossover: in such an analysis, by construction, $K^{(\ell=1)}_{[2]}$ is infinitesimally small. This means that the crossover scale (5.169) is pushed to infinity, invisible to perturbation theory. Here is another way to see it. For small separation of two inputs, we can rewrite the angle as

$$\psi^{(\ell)} \approx 2\sqrt{\frac{\delta\delta K^{(\ell)}_{[2]}}{K^\star_d}} + \ldots , \tag{5.170}$$

and hence the angle recursion (5.164) can be recast – upon a squaring and a rearrangement of terms – as

$$\delta\delta K^{(\ell+1)}_{[2]} = \delta\delta K^{(\ell)}_{[2]} - \frac{4\rho}{3\sqrt{K^\star_d}}\left(\delta\delta K^{(\ell)}_{[2]}\right)^{\frac{3}{2}} + \ldots . \tag{5.171}$$

[30] For deep linear networks where $\rho = 0$, the solution (5.168) is degenerate and doesn't apply. However, from our discussion just before we know that for such networks the polar angle remains constant at any depth.

Unfortunately, it's impossible to generate such a non-integer power, $3/2$, via a Taylor expansion. Given our our ansatz for the perpendicular perturbation $K^{(\ell)}_{[2]}$ (5.24), this explains why the correction term was invisible before. (There is no such issue for smooth activation functions; our Taylor expansion and subsequent analysis can be completely trusted for the $K^\star = 0$ universality class.)

The overall lesson here is that we should be very careful whenever encountering singular Gaussian expectations. In the future when we need to consider multiple inputs for nonlinear scale-invariant activation functions, we'll make sure to recall the results here.

Chapter 6

Bayesian Learning

> *. . . the mathematical rules of probability theory are not merely rules for calculating frequencies of 'random variables'; they are also the unique rules for conducting inference (i.e. plausible reasoning) of any kind, and we shall apply them in full generality to that end.*
>
> E. T. Jaynes, explaining the theme of his book [48].

In the previous three chapters, we've spent a considerable amount of spacetime analyzing the ensemble of wide neural networks at initialization. In particular, through the $1/n$ expansion and deep asymptotic analysis, we've obtained a rather thorough understanding of the interplay between the architecture, width, depth, and initialization hyperparameters that together define the effective distribution of preactivations.

In this study, we've paid very careful attention to the *deep* of *deep learning* to the total neglect of the *learning*. But this is a *deep learning book*, not just a *deep book*. Thus, in this chapter we will begin to learn about learning and – if the titles of our chapters are any real guide to their contents – will continue learning about learning for the rest of the book.

We'll begin on our learning quest with a discussion of *Bayesian inference*, as it provides a natural framework for thinking about learning in general. We'll first explain in §6.1 the Bayesian approach to probability, in which probabilities are reinterpreted to represent the strength of our beliefs about the world according to different hypotheses. There, we'll learn that the rules of Bayesian inference – really the rules of logic extended to probabilistic reasoning – pick out a logically consistent way of incorporating newly observed information into the probabilistic models representing our hypotheses.

From §6.2 on out, we'll see why this simple yet powerful framework enables us to analyze and then understand how deep neural networks learn from observed data.

In §6.2.1, we'll detail how Bayesian model fitting works for neural networks. First, we'll reinterpret our well-studied effective preactivation distribution as a *prior* distribution, encoding our initial beliefs about the model outputs before observing any data. With this as a starting point, the rules of Bayesian inference then imply a learning algo-

rithm for sharpening our beliefs so as to best fit our observations. The result of inference – the *posterior* distribution – further lets us make Bayesian predictions on novel inputs whose outputs we haven't observed but need to infer. This naturally segues into a discussion of practical implementations: first we'll discuss approximation methods – giving a Bayesian interpretation to the gradient-based learning methods that we'll explore in the epochs following this chapter – and then we'll discuss an exact method on which the rest of the current chapter will be based.

In §6.2.2, we'll expand our horizons by contemplating the ultimate question of Life, the Universe, and Everything: Bayesian model comparison. We'll explain how to use Bayesian *evidence* to select between different plausible hypotheses, organized according to different choices of hyperparameters and network architectures, in order to pick the best ones. Bayesian model comparison also gives us a quantitative means to address *inductive biases*, the often hidden assumptions built into deep learning models. As a bonus, we'll further see how *Occam's razor* is automatically incorporated in the rules of Bayesian inference applied to such model comparison. With these tools, we can really begin to address one of the fundamental questions we posed at the beginning of the book: why do some neural network models perform so well while others fail?

These abstract discussions are then followed by an onslaught of concrete calculations for infinite- and finite-width neural networks in §6.3 and §6.4, respectively.

Some of these calculations reinforce the themes of the previous chapter. We'll first show that Bayesian model comparison prefers critical initialization hyperparameters, giving additional evidence for the *principle of criticality* (§6.3.1). We'll also illustrate another role of finite-width interactions. Specifically, the accumulation of correlated *fluctuations* induces an inductive bias for *neural association*, leading to a propensity for Hebbian learning – a learning principle inspired by biological neurons (§6.4.1).

Some of these calculations contrast qualitatively different characteristics of infinite- and finite-width models that are trained with exact Bayesian learning. Analyzing the posterior distribution of network outputs, we'll see that correlations among different components of the output are nonzero at finite width only (§6.3.2 $\perp$ §6.4.2). The resulting expressions will also make it clear why – while theoretically quite tractable – exact Bayesian learning is impractical for any dataset of reasonable size. Next, analyzing the posterior distribution of hidden-layer representations, we'll see the absence/presence of representation learning at infinite/finite width (§6.3.3 $\perp$ §6.4.3). Overall, this contrasting will provide a valuable blueprint for when we later consider infinite- and finite-width models trained with gradient-based learning (§10 $\perp$ §∞).

6.1 Bayesian Probability

A Bayesian always starts with a **hypothesis** $\mathcal{H}$. Mathematically, a hypothesis is a mechanism for assigning numbers $p(A|\mathcal{H})$ to *statements* A about the world. These statements are logical propositions – such as "it will rain tomorrow" or "this image x contains a cat" or "the output value for this function $f(x)$ evaluated on an input x is z" – and these numbers $p(A|\mathcal{H})$ represent the relative plausibilities of those statements

according to the assumptions or model of the world summarized by the hypothesis $\mathcal{H}$. In the context of machine learning, $p(A|\mathcal{H})$ is often called a **probabilistic model**.

As this notation and discussion should make clear, these beliefs $p(A|\mathcal{H})$ are expressed in the language of probability. However, the *Bayesian* interpretation of the probability $p(A|\mathcal{H})$ subtlety differs from the *ensemble* interpretation that we gave in §2.3. Namely, rather than representing the statistics of a random variable – the relative frequency or chance observing A, given the conditions $\mathcal{H}$ – this probability instead constitutes the strength of our belief in the proposition A according to the assumptions $\mathcal{H}$.[1] Further, with such a Bayesian perspective all of probability theory and statistical inference can be uniquely derived as a consequence of logical constraints on these beliefs $p(A|\mathcal{H})$.[2] We'll next brief you through these constraints as they form the foundation of this chapter but, as we have been using probabilities for quite a while now in this book, let us be brief.

Formally, the first logical constraint is known as the **product rule**,

$$p(A, B|\mathcal{H}) = p(A|B, \mathcal{H})\, p(B|\mathcal{H}) = p(B|A, \mathcal{H})\, p(A|\mathcal{H})\,, \tag{6.1}$$

where $p(A, B|\mathcal{H})$ represents a *joint* belief in both A *and* B according to the hypothesis $\mathcal{H}$, while $p(A|B, \mathcal{H})$ represents a *conditional* belief in A according to $\mathcal{H}$ *given* that B has been observed. The second logical constraint is known as the **sum rule**,

$$p(A|\mathcal{H}) = \sum_B p(A, B|\mathcal{H})\,, \tag{6.2}$$

and relates the joint belief in A and B to a marginal belief in just A.[3] Here, the symbol $\sum_B$ represents a sum over all the logically possible values of a discrete variable B, or for a continuous variable it represents an integral.[4] This sum rule in particular implies the normalization condition if we assign $p(C|\mathcal{H}) \equiv 1$ for the statement C that holds with *absolute certainty* according to $\mathcal{H}$:

$$\sum_B p(B|\mathcal{H}) = \sum_B p(C, B|\mathcal{H}) = p(C|\mathcal{H}) = 1\,. \tag{6.3}$$

[1]The *ensemble* interpretation is often called **frequentist probability** when contrasted with **Bayesian probability**. In this book, we use the interpretation that is most appropriate for the particular problem under consideration: if we're instantiating models by randomly drawing parameters from an initialization distribution, it makes sense to analyze an ensemble; if we're making inferences based on a fixed hypothesis or comparing different hypotheses, it makes sense to adopt the Bayesian perspective.

[2]See Jaynes' book [48] for an extended development of this perspective for which our brief summary does not give justice.

[3]We essentially discussed this sum rule as (4.93) under §4.4 *Marginalization Rules*.

[4]Though (Bayesian) probably it's already clear if you've made it this deep in the book, as we cannot be (Bayesian) certain, let us clarify the meaning of the *statement* A inside the belief system $p(A|\mathcal{H})$. Sometimes a statement represents a *fixed* logical proposition, such as $A =$ "Schrödinger's cat is alive" with $p(A|\mathcal{H})$ encoding the plausibility of cat's aliveness. Sometimes a statement represents a binary *variable*, such as $B =$ "the livelihood of Schrödinger's cat" which takes values in $\{\text{dead}, \text{alive}\}$ with $p(B|\mathcal{H})$ giving the distribution over the two binary outcomes. More generally, the statement can represent observable outcomes $\mathcal{O}$ of experiments – a.k.a. *observables* – with $p(\mathcal{O}|\mathcal{H})$ encoding our relative belief in the plausibilities of the different outcomes, where such observables can take on a discrete or continuous spectrum of values. Prominent examples of such general observables for us include the model parameters θ and preactivations $z^{(\ell)}$.

With these rules in mind, after fixing a hypothesis a Bayesian then gathers information in order to refine the plausibilities of different beliefs. For instance, after *observing* A, we may want to *update* our beliefs about B. Such **Bayesian inference** can be accomplished by noting that an algebraic rearrangement of the product rule (6.1) tells us how our beliefs should change as we condition on additional information A:

$$p(B|A,\mathcal{H}) = \frac{p(A|B,\mathcal{H})\,p(B|\mathcal{H})}{p(A|\mathcal{H})}\,. \tag{6.4}$$

This rearrangement is so important that it's given its own name, **Bayes' rule**, and even each individual factor of the equation is named as well:

- The factor $p(B|\mathcal{H})$ is called the **prior** of B, thusly named because it quantifies our belief in B *a priori*; that is, it encodes our belief in B based entirely on our model $\mathcal{H}$ before we observe any additional information.
- The factor $p(B|A,\mathcal{H})$ is called the **posterior** of B given A, thusly named because it quantifies our belief in B *a posteriori* upon learning A; that is, it encodes how our model $\mathcal{H}$ updates its belief in B after observing A.
- The factor $p(A|B,\mathcal{H})$ is called the **likelihood**. We'll elaborate more on its name and interpretation later in §6.2.1 where we talk about model *fitting*.
- The factor $p(A|\mathcal{H})$ is called the **evidence** for $\mathcal{H}$. We'll elaborate more on its name and interpretation later in §6.2.2 where we talk about model *comparison*.

Note that the posterior is automatically normalized:

$$\sum_B p(B|A,\mathcal{H}) = \sum_B \frac{p(A|B,\mathcal{H})\,p(B|\mathcal{H})}{p(A|\mathcal{H})} = \sum_B \frac{p(A,B|\mathcal{H})}{p(A|\mathcal{H})} = \frac{p(A|\mathcal{H})}{p(A|\mathcal{H})} = 1\,. \tag{6.5}$$

More importantly, Bayes' rule is the only logically consistent way to update a set of beliefs after making observations.

6.2 Bayesian Inference and Neural Networks

The Bayesian framework for inference can be used for building, updating, and reasoning with powerful probabilistic models of the world. Let's now see how we can apply the Bayesian framework to deep learning, first for model fitting (§6.2.1) and then for model comparison (§6.2.2).

6.2.1 Bayesian Model Fitting

For neural networks, it's most natural to begin by discussing the prior distribution $p(\theta|\mathcal{H})$ of the model parameters $\theta_\mu = \left\{b_i^{(\ell)}, W_{ij}^{(\ell)}\right\}$. This prior lets us quantify our initial beliefs about the particular values of the model parameters that determine our neural-network

function approximator $f(x;\theta)$. The most common choice is to simply reinterpret the initialization distribution of the ensemble,

$$p(\theta|\mathcal{H}) \equiv \prod_{\ell=1}^{L} \left\{ \left[\prod_{i=1}^{n_\ell} p\Big(b_i^{(\ell)}\Big) \right] \left[\prod_{i=1}^{n_\ell} \prod_{j=1}^{n_{\ell-1}} p\Big(W_{ij}^{(\ell)}\Big) \right] \right\}, \tag{6.6}$$

as our Bayesian prior distribution. Here we recall that $p\Big(b_i^{(\ell)}\Big)$ and $p\Big(W_{ij}^{(\ell)}\Big)$ – given by (2.21) and (2.22) – are zero-mean Gaussian distributions with bias variance $C_b^{(\ell)}$ and weight variance $C_W^{(\ell)}/n_{\ell-1}$, respectively.

From the Bayesian perspective, these initialization hyperparameters are part of the hypothesis $\mathcal{H}$. This hypothesis $\mathcal{H}$ also contains our choice of architecture – MLP, CNN, transformer, etc. – as well as all the architecture hyperparameters within that architecture class – e.g. for MLPs we need to further select the depth L, the hidden-layer widths n_ℓ, and the activation function $\sigma(z)$. In short, $\mathcal{H}$ is for $\mathcal{H}$yperparameters.[5]

Here, we've taken familiar objects – the hyperparameters and the initialization distribution characterizing the frequency of potential network realizations – and interpreted them in the way of Bayes – as the hypothesis $\mathcal{H}$ and as the prior distribution $p(\theta|\mathcal{H})$ characterizing our initial beliefs about the value of the model parameters. Another familiar object, of course, is the distribution of ℓ-th-layer preactivations that we've spent last three chapters evaluating explicitly. To give that a Bayesian interpretation, let us first denote by $z_{\mathcal{D}}^{(\ell)} \equiv \left\{ z_{i;\delta}^{(\ell)} \right\}$ the set of ℓ-th-layer preactivations evaluated on inputs $x_{j;\delta} \in \mathcal{D}$ in some dataset $\mathcal{D}$. Then, the prior distribution over these ℓ-th-layer preactivations can be related to the prior distribution over the model parameters by

$$p\Big(z_{\mathcal{D}}^{(\ell)}\Big|\mathcal{H}\Big) = \int \left[\prod_{\mu=1}^{P} d\theta_\mu \right] p\Big(z_{\mathcal{D}}^{(\ell)}, \theta\Big|\mathcal{H}\Big) = \int \left[\prod_{\mu=1}^{P} d\theta_\mu \right] p\Big(z_{\mathcal{D}}^{(\ell)}\Big|\theta, \mathcal{H}\Big)\, p(\theta|\mathcal{H})\,, \tag{6.7}$$

where we've applied the sum rule (6.2) in the first equality and the product rule (6.1) in the second. This prior quantifies our initial beliefs about the different neural-network variables. More specifically, for a hidden layer ℓ, this distribution represents our beliefs about a particular *feature representation* of the input and, for the output layer L, this represents our initial beliefs about the behavior of the *function approximation* $f(x;\theta)$. More generally, for any neural-network observable $\mathcal{O} = \mathcal{O}(\theta)$, our prior beliefs

[5] To be strict, we should have always conditioned on $C_b^{(\ell)}$, $C_W^{(\ell)}$, and n_ℓ whenever we discussed the initialization distribution: $p(\theta) \to p\Big(\theta\Big|n_0, C_b^{(1)}, C_W^{(1)}, \ldots, n_{L-1}, C_b^{(L)}, C_W^{(L)}\Big)$. Thankfully we've so far left, and will continue to leave, this type of detailed dependence implicit for notational simplicity. However, to underscore the importance of the hypothesis for Bayesian inference, in this chapter we *(i)* will leave the conditioning on the overall hypothesis $\mathcal{H}$ explicit until the end of §6.3.1 and at the same time *(ii)* will move the dependence of a dataset $\mathcal{D}$ to an overall subscript of the preactivations. As a particular example, the prior distribution of the ℓ-layer preactivations $p\Big(z_{\mathcal{D}}^{(\ell)}\Big|\mathcal{H}\Big)$, defined next paragraph in (6.7), is equivalent to what we've been denoting as $p\Big(z^{(\ell)}\Big|\mathcal{D}\Big)$ outside of this chapter.

are determined by

$$p(\mathcal{O}|\mathcal{H}) = \int \left[\prod_{\mu=1}^{P} d\theta_\mu\right] p(\mathcal{O}|\theta,\mathcal{H})\; p(\theta|\mathcal{H})\,. \tag{6.8}$$

To better illustrate what these formal expressions represent, let us take the network output $z_{\mathcal{D}}^{(L)}$ as an observable. Then, the prior distribution for the output layer $p\Big(z_{\mathcal{D}}^{(L)}\Big|\mathcal{H}\Big)$, (6.7), is the same distribution as the output distribution induced by the initialization ensemble, (2.35), *if and only if* we also pick the conditional distribution of the outputs given the parameters to be deterministic:

$$p\Big(z_{\mathcal{D}}^{(L)}\Big|\theta,\mathcal{H}\Big) = \prod_{i=1}^{n_L}\prod_{\delta\in\mathcal{D}} \delta\Big(z_{i;\delta}^{(L)} - f_i(x_\delta;\theta)\Big)\,. \tag{6.9}$$

Here, $f_i(x_\delta;\theta)$ is an expression for the network output given in terms of the iteration equation that defines the MLP (2.5), while $z_{i;\delta}^{(L)}$ is interpreted as a random variable. The resulting prior distribution for the network outputs $p\Big(z_{\mathcal{D}}^{(L)}\Big|\mathcal{H}\Big)$ then characterizes our overall initial belief about the joint set of output values for a given set of inputs $\mathcal{D}$ according to the hypothesis $\mathcal{H}$, instead of characterizing the relative frequency of such output values at initialization across different realizations of the model parameters. That said, operationally, the formalism developed in the previous chapters can be directly brought to bear on calculating with these beliefs.

Importantly, note that the deterministic conditional distribution for the output (6.9) is a part of our hypothesis within the Bayesian framework: according to the hypothesis $\mathcal{H}$, given the model parameters θ, the outputs are *definitely* the ones computed by the function $f(x;\theta)$. Another common hypothesis is the *uncertain hypothesis*

$$p\Big(z_{\mathcal{D}}^{(L)}\Big|\theta,\mathcal{H}\Big) = \prod_{i=1}^{n_L}\prod_{\delta\in\mathcal{D}} \left\{\frac{1}{\sqrt{2\pi\sigma_\varepsilon^2}}\exp\left[-\frac{1}{2\sigma_\varepsilon^2}\Big(z_{i;\delta}^{(L)} - f_i(x_\delta;\theta)\Big)^2\right]\right\}, \tag{6.10}$$

which reduces to the deterministic hypothesis (6.9) in the limit of zero variance and absolute certainty: $\sigma_\varepsilon^2 \to 0$.[6]

[6]This hypothesis is equivalent to injecting random noise ε_i with mean zero and variance σ_ε^2 into the network output. This in turn is tantamount to shifting the last-layer biases as $b_i^{(L)} \to b_i^{(L)} + \varepsilon_i$, and hence we can easily incorporate this in our analysis by shifting the final bias variance as $C_b^{(L)} \to C_b^{(L)} + \sigma_\varepsilon^2$. You should keep in mind, however, that ε_i is separate from the bias $b_i^{(L)}$ and is *not* a part of the adjustable model parameters θ_μ; instead, this noise is intended to embody an intrinsic uncertainty present in our observation of the model's output.

Before moving on, let us also mention one other common hypothesis for the network output, the *categorical hypothesis*, defined for each input x by

$$p(i|\theta,\mathcal{H}) \equiv \frac{\exp\left[f_i(x;\theta)\right]}{\sum_{j=1}^{n_L}\exp\left[f_j(x;\theta)\right]}\,. \tag{6.11}$$

This distribution is also sometimes known as the *softmax*. Here, instead of considering a continuous

Having now thoroughly discussed the prior, let's next consider the posterior. As we gather more information A about the true behavior of our desired function $f(x)$, we should update our beliefs about our probabilistic model for $f(x;\theta)$. In order to incorporate this information in a logically consistent manner, we should use Bayes' rule. Specifically, to update our belief about the model parameters, Bayes' rule (6.4) instructs us to use

$$p(\theta|A,\mathcal{H}) = \frac{p(A|\theta,\mathcal{H})\,p(\theta|\mathcal{H})}{p(A|\mathcal{H})}\,. \tag{6.12}$$

Here, to find the posterior distribution $p(\theta|A,\mathcal{H})$, the prior distribution $p(\theta|\mathcal{H})$ gets multiplied by the likelihood $p(A|\theta,\mathcal{H})$ of the model parameters θ for the observation of A, and divided by the evidence $p(A|\mathcal{H})$. Consequently, with such a posterior distribution of the model parameters, our beliefs about any neural-network observable $\mathcal{O}$ shifts from our prior $p(\mathcal{O}|\mathcal{H})$ (6.8) to a posterior with the insertion of A,

$$p(\mathcal{O}|A,\mathcal{H}) = \int \left[\prod_{\mu=1}^{P} d\theta_\mu\right] p(\mathcal{O}|\theta,\mathcal{H})\; p(\theta|A,\mathcal{H})\,. \tag{6.13}$$

These two equations (6.12) and (6.13) uniquely determine how new information A can be incorporated to change our beliefs about the value of any neural-network observable.

For function approximation tasks, such information often comes in the form of some dataset $\mathcal{A}$ containing observed input-output pairs:

$$A \equiv \{(x_{j;\tilde{\alpha}}, y_{i;\tilde{\alpha}})\}\,|_{\tilde{\alpha}\in\mathcal{A}}\,. \tag{6.14}$$

Here, each input $x_{j;\tilde{\alpha}} \in \mathcal{A}$ is paired with its corresponding true output $y_{i;\tilde{\alpha}} \equiv f_i(x_{\tilde{\alpha}})$ recorded from our desired function $f(x)$.[7] With our observation of the true values $y_{\mathcal{A}} \equiv \{y_{i;\tilde{\alpha}}\}$, the likelihood and evidence are then given by the conditional belief $p(y_{\mathcal{A}}|\theta,\mathcal{H})$ and the belief $p(y_{\mathcal{A}}|\mathcal{H})$ for outputs, respectively. Such beliefs appeared before when considering the prior distribution of the outputs, (6.7) with $\ell = L$, but are now evaluated on the *fixed* values $y_{\mathcal{A}}$ associated with the given inputs $x_{\mathcal{A}}$.

To develop some intuition for what this means, let's again take the deterministic hypothesis (6.9). In this case, the likelihood is given by

$$p(A|\theta,\mathcal{H}) \equiv p(y_{\mathcal{A}}|\theta,\mathcal{H}) = \prod_{\tilde{\alpha}\in\mathcal{A}} \prod_{i=1}^{n_L} \delta\Big(y_{i;\tilde{\alpha}} - f_i(x_{\tilde{\alpha}};\theta)\Big)\,. \tag{6.15}$$

This likelihood quite explicitly restricts the model parameters to those *exactly* satisfying the constraints $f_i(x_{\tilde{\alpha}};\theta) = y_{i;\tilde{\alpha}}$ *fitting* our observations. Vice versa, the functions in our

distribution over the n_L output *values* $z_i^{(L)}$, we consider a discrete distribution over output *classes* i, such as `dog` or `cat` or `car`; then, for such classification tasks, each number $p(i|\theta,\mathcal{H})$ quantifies our belief about how likely the input x represents the class i. Functionally, the softmax can be thought of as a generalization of the logistic function (2.10) in the sense that it maps a vector of real numbers to a discrete probability distribution.

[7] For maximal disambiguation, in this chapter we'll use sample indices of the form $\tilde{\alpha}$ – the Greek letter alpha with a tilde on top – for elements of the dataset $\mathcal{A}$ corresponding to input-output pairs for which the *true* output values from $f(x)$ are observed.

set that do *not* satisfy these constraints are completely thrown away from the posterior distribution, deemed *unlikely*. Note what has just happened. Naively, $p(y_{\mathcal{A}}|\theta, \mathcal{H})$ represents our beliefs about the output values $y_{\mathcal{A}}$, given that we set the parameters of our model to θ. However, here we *first* observed the true output values $y_{\mathcal{A}}$ and *then* interpreted $p(y_{\mathcal{A}}|\theta, \mathcal{H})$ in terms of how likely the model parameters θ fit the observation A. This is the origin of the name "likelihood" and why the proper way to refer to it is "the likelihood of the model parameters θ for the observation A."

To develop even more intuition, it's customary to introduce the **negative log-likelihood** $\mathcal{L}_{\mathcal{A}}(\theta)$ – or *loss* – representation of the likelihood:

$$p(y_{\mathcal{A}}|\theta, \mathcal{H}) \equiv \exp\left[-\mathcal{L}_{\mathcal{A}}(\theta)\right] . \tag{6.16}$$

Here, by parameterizing the loss as a function of the parameters θ, we are emphasizing that it's the (negative log-)likelihood *of* the parameters.[8] For the uncertain hypothesis (6.10), the negative log-likelihood takes the form of the famous mean-squared-error or **MSE loss**:

$$\mathcal{L}_{\text{MSE}}(\theta) = \sum_{\tilde{\alpha}\in\mathcal{A}} \left\{ \frac{1}{2\sigma_\varepsilon^2} \left[f_i(x_{\tilde{\alpha}}; \theta) - y_{i;\tilde{\alpha}} \right]^2 + \frac{1}{2} \log\Bigl(2\pi\sigma_\varepsilon^2\Bigr) \right\} . \tag{6.17}$$

In particular, as the network outputs $f_i(x_{\tilde{\alpha}}; \theta)$ get closer to their target values $y_{i;\tilde{\alpha}}$, the MSE loss decreases and the likelihood increases.[9] As such, the loss is a natural measure of how well our model is approximating the true behavior of the function. Additionally, since the loss (6.17) involves an explicit sum over observations, as the number of observed input-output pairs $N_{\mathcal{A}}$ increases, the likelihood can dominate the prior; that is, if we gather enough information, eventually our prior beliefs can become entirely replaced by what we learned from our observations.

This is **Bayesian model fitting**: Bayesian inference (6.12) is used as a *learning algorithm* to increase the accuracy of a function approximation. It gives greater preference to the functions that better fit the constraints $f_i(x_{\tilde{\alpha}}; \theta) = y_{i;\tilde{\alpha}}$ and penalize the ones that don't. The posterior (6.12) is then updated to reflect a balance between this preference for fitting our observations and an adherence to our prior beliefs about the values the model parameters should take.

Ultimately, we want to use our fit Bayesian model to make **Bayesian predictions**. This is generically and abstractly embodied in (6.13). Specifically and concretely, for

[8] While the likelihood function – and therefore the loss – is considered auxiliary from the perspective of function approximation, from the perspective of Bayesian inference the form of the likelihood is considered to be part of the hypothesis, cf. the deterministic hypothesis (6.9) vs. the uncertain hypothesis (6.10).

[9] In the deterministic limit $\sigma_\varepsilon^2 \to 0$, the loss $\mathcal{L}_{\mathcal{A}}(\theta)$ would be infinite for functions that don't *exactly* fit all the constraints $f_i(x_{\tilde{\alpha}}; \theta) = y_{i;\tilde{\alpha}}$ and negative infinite for those that do. Thus, the uncertain hypothesis softens these hard-fitting constraints of the deterministic hypothesis by relaxing the Dirac delta function distribution to a Gaussian distribution with a finite variance σ_ε^2.

When we consider the categorical hypothesis (6.11), the negative log-likelihood of the softmax distribution gives the *cross-entropy loss*. We'll more systematically address the consequences of these different choices of loss functions in §10.

function approximation tasks we are most often interested in posterior beliefs about the network outputs $\mathcal{O} = z^{(L)}$, for which (6.13) reads

$$p\Big(z^{(L)}\Big|A,\mathcal{H}\Big) = \int \left[\prod_{\mu=1}^{P} d\theta_\mu\right] p\Big(z^{(L)}\Big|\theta,\mathcal{H}\Big)\; p(\theta|A,\mathcal{H})\,. \tag{6.18}$$

Once we have this distribution, then we can in particular use its mean as our prediction and its variance as our level of confidence. To compute any of these quantities, one way or another we need to perform a gigantic integral over the model parameters θ in order to properly weight our different beliefs. With that in mind, we'll now present two kinds of methods to tackle this model marginalization: *(i)* approximate methods based on saddle-point approximations and *(ii)* an exact method based on our effective theory approach.

Approximation methods for model marginalization: MAP and MLE

One way to tackle such a gigantic integral is to presume that the integral measure, given by the posterior distribution $p(\theta|A,\mathcal{H})$ (6.12), is very concentrated around its *mode*:

$$\theta^\star_{\text{MAP}} \equiv \arg\max_\theta\; p(\theta|A,\mathcal{H}) = \arg\max_\theta\; \left[p(y_\mathcal{A}|\theta,\mathcal{H})\, p(\theta|\mathcal{H})\right]\,. \tag{6.19}$$

This maximum is known as the **maximum a posteriori** (MAP) estimate. After such a maximazation, we can use the function $f(x;\theta^\star_{\text{MAP}})$ for tasks and more generally approximate the full posterior distribution $p(\mathcal{O}|A,\mathcal{H})$ (6.12) by the point estimate $\mathcal{O}\left(\theta^\star_{\text{MAP}}\right)$. This notion of approximating a probability distribution with single value of the random variable is known in statistics as a *point estimate* and in physics as a *saddle-point approximation*. Another commonly-used saddle is given by the maximum of the likelihood,

$$\theta^\star_{\text{MLE}} \equiv \arg\max_\theta\; p(y_\mathcal{A}|\theta,\mathcal{H})\,, \tag{6.20}$$

known as the **maximum likelihood estimation** (MLE) of the model parameters.

In terms of the negative log-likelihood $\mathcal{L}_\mathcal{A}(\theta)$, MLE is equivalent to the minimization of the loss

$$\theta^\star_{\text{MLE}} = \arg\min_\theta\, \mathcal{L}_\mathcal{A}(\theta)\;, \tag{6.21}$$

while MAP estimate (6.19) is a joint minimization of the loss and the negative log of the prior,

$$\theta^\star_{\text{MAP}} = \arg\min_\theta\, \left\{\mathcal{L}_\mathcal{A}(\theta) - \log\left[p(\theta|\mathcal{H})\right]\right\}\,. \tag{6.22}$$

In particular for a generic Gaussian prior of the form $p(\theta|\mathcal{H}) \propto \exp\big(-\sum_{\mu=1}^{P} a_\mu \theta_\mu^2\big)$, the negative-log prior acts as a *regularization* term of the form $\sum_{\mu=1}^{P} a_\mu\theta_\mu^2$ that has an effect of penalizing large parameter magnitudes. Since the loss grows extensively with the size of the dataset $\mathcal{A}$ while this regularization term stays constant, when we've made sufficiently many observations, we naively expect that the prior will be eventually overwhelmed by the likelihood and that the MAP and MLE estimates will become similar.

If we are to apply these approximation methods to wide neural networks, there are certain things we need to keep in mind.[10] First of all, there is actually no single optimal value for the maximum likelihood estimation $\theta^\star_{\text{MLE}}$. Instead, there are continuum of such optima, and we still have to consider a distribution over them. Importantly, such a distribution over maxima depends critically on how the maxima are obtained. For instance, it depends on the way you initialize model parameters θ_{init}, the learning algorithm used to estimate these maxima – such as gradient descent vs. *stochastic* gradient descent – and the *training hyperparameters* controlling the learning algorithm. The study of this ensemble over optima and its dependence on the initialization and training hyperparameters will more or less be the focus of the following chapters §7–§∞.[11]

Exact method for model marginalization: effective theory

For the prior (6.7), we know very well that it's possible to directly integrate out the model parameters through the use of a $1/n$ expansion. Such a gigantic marginalization was the focus of §4, and in writing (6.7) we already reinterpreted our effective preactivation distribution at initialization as our prior beliefs about the preactivations. For the posterior, the only hypothetical worry would be that we'd need to carry out entirely different sets of integrals. We'll show next that there is no such need. Thus, in a very real sense the most painstaking theoretical part of Bayesian inference has already been taken care of for us!

Let's continue to suppose that we've made some observations A of the true outputs $y_{i;\tilde{\alpha}} \equiv f_i(x_{\tilde{\alpha}})$ of our function $f(x)$ for a given set of inputs $x_{\mathcal{A}}$ in a subsample $\mathcal{A}$ as defined by (6.14). We now want to incorporate what we've learned from these observations in order to update our beliefs about the output values $z^{(L)}_{\mathcal{B}} \equiv \left\{ z^{(L)}_{i;\dot{\beta}} \right\}$ for a potentially different set of inputs $x_{j;\dot{\beta}} \in \mathcal{B}$ in another subsample $\mathcal{B}$.[12] Beginning with the joint prior

[10] In §10, we'll go through how all of this works in detail.

[11] Since those following chapters will unsentimentally drop our Bayesian lens, let's interpret these different methods with fresh Bayesian eyes here.

In the *impure* Bayesian approach – that is MLE – we have an initialization distribution $p(\theta_{\text{init}})$, but no prior distribution $p(\theta|\mathcal{H})$. By construction, the prior distribution does not enter into the estimate of the impure Bayesian (6.20), but the initialization distributions (2.21) and (2.22) enters into their code to give particular realizations of networks acting as the starting points for optimization and training. Thus, such an initialization distribution induces a distribution over the resulting MLE estimates.

In the *less impure* Bayesian approach – that is MAP – we have *both* an initialization distribution $p(\theta_{\text{init}})$ *and* a prior distribution $p(\theta|\mathcal{H})$. For the former, we again use the initialization distributions (2.21) and (2.22) to provide starting points for optimization; for the latter, we typically use a Gaussian prior $p(\theta|\mathcal{H}) \propto \exp\left(-\sum_{\mu=1}^{P} a_\mu \theta_\mu^2\right)$ which, as we said, serves as a regularization term when added to the optimization objective – the loss – as per (6.22).

In the *pure* Bayesian approach – which is the focus of the rest of this chapter – there is a prior distribution $p(\theta|\mathcal{H})$ but the initialization distribution $p(\theta_{\text{init}})$ isn't needed. Pure Bayesians always integrate. What we really did with (6.6) was pick a Gaussian prior over the parameters and then adopt the same conventions for the variances as we've been using for the initialization distribution (2.21) and (2.22). We'll see in the rest of the chapter why this is sensible.

[12] For maximal disambiguation, in this chapter we'll use sample indices of the form $\dot{\beta}$ – the Greek letter beta with a dot on top – for elements of the dataset $\mathcal{B}$ corresponding to input-output pairs for which

for the network outputs over the union of both subsamples $\mathcal{D} \equiv \mathcal{A} \cup \mathcal{B}$,

$$p\Big(z_{\mathcal{D}}^{(L)}\Big|\mathcal{H}\Big) \equiv p\Big(z_{\mathcal{A}}^{(L)}, z_{\mathcal{B}}^{(L)}\Big|\mathcal{H}\Big) \;, \tag{6.23}$$

we can set $z_{\mathcal{A}}^{(L)} \to y_{\mathcal{A}}$ and use the product rule (6.1) to condition our beliefs about $z_{\mathcal{B}}^{(L)}$ on the observed *true* values $y_{\mathcal{A}}$:

$$p\Big(y_{\mathcal{A}}, z_{\mathcal{B}}^{(L)}\Big|\mathcal{H}\Big) = p\Big(z_{\mathcal{B}}^{(L)}\Big|y_{\mathcal{A}}, \mathcal{H}\Big)\, p\Big(y_{\mathcal{A}}\Big|\mathcal{H}\Big) \;. \tag{6.24}$$

Then, rearranging terms like we are Reverend Thomas Bayes, we get

$$p\Big(z_{\mathcal{B}}^{(L)}\Big|y_{\mathcal{A}}, \mathcal{H}\Big) = \frac{p\Big(y_{\mathcal{A}}, z_{\mathcal{B}}^{(L)}\Big|\mathcal{H}\Big)}{p\Big(y_{\mathcal{A}}\Big|\mathcal{H}\Big)} \;. \tag{6.25}$$

Since this iteration of Bayes' rule is so important, let us be verbose and crystal clear about its interpretation: the denominator $p(y_{\mathcal{A}}|\mathcal{H})$ is the prior for the network outputs given the inputs $x_{\mathcal{A}}$ in the subsample $\mathcal{A}$, evaluated on the *fixed* observed values $y_{\mathcal{A}}$, hence it is just a *number*; the numerator $p\Big(y_{\mathcal{A}}, z_{\mathcal{B}}^{(L)}\Big|\mathcal{H}\Big)$ is the prior for the network outputs given the inputs $x_{\mathcal{D}}$ in the joint dataset $\mathcal{D} \equiv \mathcal{A} \cup \mathcal{B}$, evaluated on the *fixed* observed values $y_{\mathcal{A}}$ but with the network outputs $z_{\mathcal{B}}^{(L)}$ still *variable*, hence it is a *function* of the $z_{\mathcal{B}}^{(L)}$.[13] The numerator and denominator combine to make the posterior on the left-hand side, which is thus a function of the random variable $z_{\mathcal{B}}^{(L)}$ encoding our posterior beliefs about the plausible values of the network outputs $z_{\mathcal{B}}^{(L)}$ for the inputs $x_{\mathcal{B}}$ in $\mathcal{B}$, updated with our observations about the true values $y_{\mathcal{A}}$ of the outputs for the inputs $x_{\mathcal{A}}$ in $\mathcal{A}$. In this way, rather than performing Bayesian inference to learn about the model parameters as way of maintaining different beliefs about the different functions $f(x;\theta)$ in our flexible set, here we simply update our beliefs about the behavior of the function $f(x)$ directly.

In this presentation of Bayes' rule (6.25), the marginalization over all the model parameters already occurred in our transition from (6.6), the prior over the parameters, to (6.7), the prior over the preactivations. The resulting posterior (6.25) is in fact exactly equivalent to what you'd get by explicitly doing a marginalization over a posterior distribution of the model parameters, e.g. as in (6.13). To see why, consider the following

---

outputs values from $f(x)$ are *not* observed but instead to be *inferred*.

[13] The reason we say *given the inputs* here is that technically we should also be conditioning on $x_{\mathcal{A}}$ and $x_{\mathcal{B}}$ as well. In particular, while $y_{\mathcal{A}}$ is fixed and $z_{\mathcal{B}}^{(L)}$ is completely variable in the expression for the joint prior

$$p\Big(y_{\mathcal{A}}, z_{\mathcal{B}}^{(L)}\Big|\mathcal{H}\Big) \equiv p\Big(y_{\mathcal{A}}, z_{\mathcal{B}}^{(L)}\Big|x_{\mathcal{D}}, \mathcal{H}\Big) \;, \tag{6.26}$$

the full set of inputs $x_{\mathcal{D}} \equiv x_{\mathcal{A}} \cup x_{\mathcal{B}}$ determines the *data-dependent couplings* $g_{(L)}$ and $v_{(L)}$ – or equivalently the metric $G^{(L)}$ and the four-point vertex $V^{(L)}$ – that parameterize the output distribution. We will see how this works in more detail in the following sections.

set of manipulations:

$$
\begin{aligned}
p\Big(z_{\mathcal{B}}^{(L)}\Big|y_{\mathcal{A}},\mathcal{H}\Big) &= \int\left[\prod_{\mu=1}^{P} d\theta_\mu\right] p\Big(z_{\mathcal{B}}^{(L)},\theta\Big|y_{\mathcal{A}},\mathcal{H}\Big) = \int\left[\prod_{\mu=1}^{P} d\theta_\mu\right] p\Big(z_{\mathcal{B}}^{(L)}\Big|\theta,\mathcal{H}\Big)\; p(\theta|y_{\mathcal{A}},\mathcal{H}) \\
&= \int\left[\prod_{\mu=1}^{P} d\theta_\mu\right] p\Big(z_{\mathcal{B}}^{(L)}\Big|\theta,\mathcal{H}\Big)\left[\frac{p(y_{\mathcal{A}}|\theta,\mathcal{H})\,p(\theta|\mathcal{H})}{p(y_{\mathcal{A}}|\mathcal{H})}\right] \\
&= \frac{1}{p(y_{\mathcal{A}}|\mathcal{H})}\int\left[\prod_{\mu=1}^{P} d\theta_\mu\right] p\Big(y_{\mathcal{A}},z_{\mathcal{B}}^{(L)}\Big|\theta,\mathcal{H}\Big)\; p(\theta|\mathcal{H}) \\
&= \frac{1}{p(y_{\mathcal{A}}|\mathcal{H})}\int\left[\prod_{\mu=1}^{P} d\theta_\mu\right] p\Big(y_{\mathcal{A}},z_{\mathcal{B}}^{(L)},\theta\Big|\mathcal{H}\Big) = \frac{p\Big(y_{\mathcal{A}},z_{\mathcal{B}}^{(L)}\Big|\mathcal{H}\Big)}{p\Big(y_{\mathcal{A}}\Big|\mathcal{H}\Big)}\,.
\end{aligned}
\tag{6.27}
$$

The only nontrivial step is in the third line, where we reversed the factorization,

$$
p\Big(z_{\mathcal{A}}^{(L)},z_{\mathcal{B}}^{(L)}\Big|\theta,\mathcal{H}\Big) = p\Big(z_{\mathcal{A}}^{(L)}\Big|\theta,\mathcal{H}\Big)\; p\Big(z_{\mathcal{B}}^{(L)}\Big|\theta,\mathcal{H}\Big)\;, \tag{6.28}
$$

and evaluated at $z_{\mathcal{A}}^{(L)} \to y_{\mathcal{A}}$. This factorization (6.28) says that the network outputs are *conditionally independent*, given the parameters. This is a consequence of the fact that – for a fixed set of network parameters – the output on an example $x_{\tilde{\alpha}}$ is entirely independent from the output evaluated on any other example $x_{\hat{\beta}}$, which is manifestly true for all the hypotheses that we mentioned. (If it were not, neural networks would be pretty useless in practice.) The use of Bayes' rule for the model parameters in the square brackets in the second line also makes manifest the connection between *Bayesian model fitting* (6.12) on the one hand and *Bayesian prediction* (6.13) on the other hand.

As we already alluded to, this exact method for model marginalization is closely connected with our effective theory approach to understanding neural networks. In particular, while the model parameters are always part of the definition of our neural networks, we've always had to integrate them out in the process of determining the distribution over the network outputs. In this way, our effective theory of deep learning has always worked directly with the entire ensemble of network functions implied by the initialization distribution of the parameters (6.6) rather than with any *particular* network. Up until now, we've motivated this ensemble approach via the *principle of typicality*, in which we use the ensemble to analyze how a typical realization is likely to behave.[14] Here we have a slightly different interpretation: rather than trying to make the ensemble describe a typical network, we actually want to consider the posterior predictions across the full set of potential networks, each weighted according to our posterior beliefs about how plausible those predictions are.

Now, after a brief detour into Bayesian model comparison, much of the focus of §6.3 and §6.4 will be the explicit evaluation of these Bayesian predictions (6.25) for infinite- and finite-width MLPs, respectively.

[14] In §8 and onwards, we'll see how this principle is manifested in neural networks *trained* via gradient-based learning.

6.2.2 Bayesian Model Comparison

In the context of Bayesian model fitting and Bayesian prediction, the *evidence* $p(y_{\mathcal{A}}|\mathcal{H})$ has thus far played essentially no role. In the context of our approximation methods, MAP and MLE and their respective maximizations (6.19) and (6.20), the value of the argument maximization is strictly independent of the evidence, since it doesn't depend on the model parameters. In the context of our exact method for Bayesian prediction, the evidence is simply the normalization factor of the posterior, which is trivial for us to compute.

To actually see the role of the evidence in action, *you mustn't be afraid to dream a little bigger, darling.* That is, rather than being fixated on a single hypothesis $\mathcal{H}$, we instead consider a multitude of different hypotheses $\mathcal{H}_a$ as possible explanations for our data. This is the essence of **Bayesian model comparison**: using the *evidence* to weigh the plausibility of different probabilistic models as explanations for all of our observations. In the context of deep learning, this corresponds to comparing our relative beliefs in the different modeling choices encapsulated in each $\mathcal{H}_a$ – i.e. comparing different hyperparameter settings – and determining which modeling choice provides the best description of our observations $y_{\mathcal{A}}$.

To begin, let us again use Bayes' rule – this time on the evidence – to invert the conditioning as

$$p(\mathcal{H}_a|y_{\mathcal{A}}) = \frac{p(y_{\mathcal{A}}|\mathcal{H}_a)\, p(\mathcal{H}_a)}{p(y_{\mathcal{A}})}\,. \tag{6.29}$$

In this form, the posterior $p(\mathcal{H}_a|y_{\mathcal{A}})$ on the left-hand side encodes our updated beliefs in the plausibility of the different hypotheses $\mathcal{H}_a$ – the different hyperparameters settings – given our observation $y_{\mathcal{A}}$, while the prior $p(\mathcal{H}_a)$ on the right-hand side encodes our initial beliefs about these hypotheses. Amusingly, the *old evidence* $p(y_{\mathcal{A}}|\mathcal{H}_a)$ for the hypothesis $\mathcal{H}_a$ from our Bayesian model fitting now appears as the *new likelihood* $p(y_{\mathcal{A}}|\mathcal{H}_a)$ of the hypothesis $\mathcal{H}_a$ for the observation $y_{\mathcal{A}}$ in the context of Bayesian model comparison. Lastly, the *new evidence* $p(y_{\mathcal{A}})$ is just a normalization factor that we can safely ignore.[15]

To see how the model comparison works, let's use (6.29) to compare two different hypothesis, $\mathcal{H}_1$ and $\mathcal{H}_2$, in order to determine which is a better fit for our observations. Since our *relative beliefs* are all that matter, let's take the ratio of the two posteriors,

$$\frac{p(\mathcal{H}_1|y_{\mathcal{A}})}{p(\mathcal{H}_2|y_{\mathcal{A}})} = \left[\frac{p(y_{\mathcal{A}}|\mathcal{H}_1)}{p(y_{\mathcal{A}}|\mathcal{H}_2)}\right] \frac{p(\mathcal{H}_1)}{p(\mathcal{H}_2)}\,, \tag{6.30}$$

[15] Unless, of course, we aren't afraid to dream even bigger. If we did – *narrator:* they won't – we'd need to introduce a *meta-hypothesis*, $\mathcal{G}$, that encodes our prior beliefs about different hyperparameter configurations $p(\mathcal{H}_a|\mathcal{G})$. This is sometimes called *Bayesian hierarchical modeling*. In this case, Bayesian model comparison in terms of this even grander evidence $p(y_{\mathcal{A}}) \to p(y_{\mathcal{A}}|\mathcal{G})$ in principle involves integrating overall *all* the probabilistic models as $p(y_{\mathcal{A}}|\mathcal{G}) = \sum_a p(y_{\mathcal{A}}|\mathcal{H}_a)\; p(\mathcal{H}_a|\mathcal{G})$, i.e. any and all hypotheses $\mathcal{H}_a$ that are encoded by $\mathcal{G}$. The distinction between the meta-hypothesis $\mathcal{G}$ and hypotheses $\mathcal{H}_a$ is somewhat arbitrary, however; for instance, we could put into $\mathcal{G}$ our overall choice of architecture – e.g. MLP, CNN, transformer – and then let $\mathcal{H}_a$ index the different settings of $\mathcal{H}$yperparameters. Then, recursing again, a Bayesian model comparison over $\mathcal{G}$ would be a weighted evaluation of the best architecture for the data, taking into account all possible settings of the hyperparameters for those architectures.

from which we see that the irrelevant normalization factor $p(y_{\mathcal{A}})$ simply drops out. Here, the ratio in the square brackets is sometimes given the name the **Bayes' factor**, which in turn multiplies the ratio of our prior beliefs. In particular, the Bayes' factor contains all of the observation dependence and characterizes how we should update our relative prior beliefs in each hypothesis given the new data $y_{\mathcal{A}}$. A ratio greater than one indicates that the model specified by hypothesis $\mathcal{H}_1$ is favored, while a ratio less than one indicates that the model specified by hypothesis $\mathcal{H}_2$ is favored. In this way, the old evidence – i.e. the new likelihood – $p(y_{\mathcal{A}}|\mathcal{H}_a)$ can be very useful, indeed.

Occam's razor

In order to further elaborate on the mechanism behind Bayesian model comparison (6.30), let us pick up **Occam's razor** [49], which is the famous *principle of sparsity*. It says that we should favor the simplest hypothesis that fits all the observations. In the context of machine learning and parameterized probabilistic modeling, this principle is often intended as a heuristic that guides us to favor models with fewer parameters, all else being equal. The intuitive explanation for this heuristic is that models with more parameters have greater flexibility to fit the observed data, making them more likely to *overfit* and less likely to *generalize* to explain new observations.[16]

Naively, Bayesian model comparison (6.29) seems to give us a very natural way to implement this razor: we can *subjectively* adjust the ratio of our prior beliefs $p(\mathcal{H}_1)/p(\mathcal{H}_2)$ to explicitly favor the simpler hypothesis, a priori penalizing more complicated models. However, as MacKay [50] points out:

> *Coherent [Bayesian] inference embodies Occam's Razor automatically and quantitatively.*

That is, Occam's razor is *objectively* built into Bayesian model comparison (6.30) through the Bayes' factor.[17]

To understand why, note that the prior distribution $p\left(z_{\mathcal{A}}^{(L)}\middle|\mathcal{H}_a\right)$ needs to be normalized. This means that for a given hypothesis $\mathcal{H}_a$ to be complicated enough to explain an overwhelmingly wide variety of *potential* observations $z_{\mathcal{A}}^{(L)}$, it must have small support on

[16] It's natural to wonder here how to interpret this overfitting in light of the fact that we've actually integrated out all our parameters! (In the machine learning literature, such ensembles are sometimes called *non-parametric models*, though we really do not like such terminology, given the following explanation.) The culprit for this potential confusion is the overloaded usage of the word *parameter*. To illustrate this with the extreme, let's consider the infinite-width limit. Despite formally starting with an infinite number of model parameters – giving a model that is naively very *overparameterized*, to say the least – the effective theory of the output distribution is completely characterized by the kernel $K^{(L)}$, which can be described by a finite number of *data-dependent couplings* $\sim N_{\mathcal{D}}^2$. Thus, from the macroscopic perspective of Bayesian model comparison, it's these couplings that control the model complexity and not what we usually call the parameters, the tunable weights and biases. We will discuss this further and in greater detail in Epilogue ε, and in particular we'll highlight how the $1/n$ expansion for finite-width networks leads to a sequence of effective theories with increasing complexity.

[17] See MacKay's excellent exposition [50] for further details and examples, with a particular emphasis on (pre-deep-learning-era) neural networks.

any *particular* observation $y_{\mathcal{A}}$. Hence the evidence $p(y_{\mathcal{A}}|\mathcal{H}_a)$ for such a hypothesis will be small regardless of which actual observation we make. In contrast, if the hypothesis is very simple, the prior $p\left(z_{\mathcal{A}}^{(L)}\middle|\mathcal{H}_a\right)$ will make a constrained set of predictions, but make them strongly, by concentrating its support on only a few plausible outcomes. Thus, the simplest models that still correctly predict the observation $y_{\mathcal{A}}$ are naturally preferred by the Bayes' factor $p(y_{\mathcal{A}}|\mathcal{H}_1)/p(y_{\mathcal{A}}|\mathcal{H}_2)$ alone. In addition, the more observations we make that are correctly predicted, the more the Bayes' factor will amplify this preference for simpler models that still fit.[18]

Since the Bayes' factor automatically and objectively implements Occam's razor, there's no need to subjectively express a preference for simpler models using the prior over hypotheses $p(\mathcal{H}_a)$. This means that for a discrete set of hypothesis $\{\mathcal{H}_a\}$, we can choose the prior distribution to be uniform, giving equal a priori preference to any particular hypothesis $\mathcal{H}_a$ regardless of their complexity. With this choice our Bayesian model comparison is completely characterized by the Bayes' factor:

$$\frac{p(\mathcal{H}_1|y_{\mathcal{A}})}{p(\mathcal{H}_2|y_{\mathcal{A}})} = \frac{p(y_{\mathcal{A}}|\mathcal{H}_1)}{p(y_{\mathcal{A}}|\mathcal{H}_2)} \,. \tag{6.31}$$

Thus, we should really think of Occam's razor as the *inductive bias* of Bayesian inference applied to model comparison.

Inductive Bias

Given our last statement, we should clarify about something that we've been informally referring to since §2.1 but now are finally ready to formally address: **inductive bias**.

Way back in §2.1, inductive biases were introduced as something implicit that is built into a neural network architecture in order that the set of functions $\{f(x;\theta)\}$ may better represent the properties of a particular dataset $\mathcal{D}$ and the function approximation task at hand. From the Bayesian perspective, inductive biases represent the a priori assumptions made about the desired function $f(x)$ before any observations are made. More broadly, both hypotheses and learning algorithms may have their own set of inductive biases; e.g. we've just pointed out that Occam's razor is an inductive bias of Bayesian inference.

Throughout §6.3 and §6.4, we'll encounter various inductive biases while performing concrete calculations for infinite- and finite-width MLPs. Here, let's consider a very simple example for illustration: suppose that a Bayesian firmly believes with *absolute certainty* that a statement B is false such that their hypothesis $\mathcal{H}_{\overline{B}}$ assigns an a priori probability of zero to this belief as $p(B|\mathcal{H}_{\overline{B}}) = 0$; then, via Bayes' rule (6.4), there's no way that the posterior on B can be updated to be anything other than zero, even if the Bayesian gathers some new information A that would serve as positive evidence for B. In this case, $\mathcal{H}_{\overline{B}}$ is clearly a bad hypothesis; its inductive bias is leading to an absurdly stubborn set of beliefs. Alternatively, if B turns out to be actually false, $\mathcal{H}_{\overline{B}}$ is a good hypothesis because it can then assign more probability to other plausibly true

[18] This is analogous to the way the likelihood factor will dominate the prior as observations accumulate when Bayesian model fitting.

statements. As this *gedanken inference* illustrates, the advantage and disadvantage of an inductive bias depends on the ground truth.

Returning to our initial example in §2.1 of the inductive bias of different neural-network architectures, the advantage of one architecture over another is a highly data- and task-dependent question. In principle, we could use Bayesian model comparison (6.30) to directly compare these different architectures – MLPs, CNNs, and transformers – for different sets of observations $y_{\mathcal{A}}$ if only we knew how to compute the evidence $p(y_{\mathcal{A}}|\mathcal{H})$ for those architectures.[19] The formalism of our effective theory of deep learning as laid out in the earlier chapters is precisely a blueprint for computing such factors for different architectures as a function of a particular dataset. We encourage you to give it a try.

6.3 Bayesian Inference at Infinite Width

In this section, we'll give three lessons on Bayesian learning in the infinite-width limit. First, we'll calculate the evidence $p(y_{\mathcal{A}}|\mathcal{H})$ and see that Bayesian model comparison prefers criticality for sufficiently deep networks (§6.3.1). Then, we'll calculate the posterior distribution for the network outputs $p\Big(z_{\mathcal{B}}^{(L)}\Big|y_{\mathcal{A}},\mathcal{H}\Big)$ and see that different output components are completely independent in this limit (§6.3.2). Finally, we'll calculate the posterior distribution of preactivations in the *penultimate* layer $p\Big(z_{\mathcal{D}}^{(L-1)}\Big|y_{\mathcal{A}},\mathcal{H}\Big)$ and show that it's identical to the penultimate prior distribution $p\Big(z_{\mathcal{D}}^{(L-1)}\Big|\mathcal{H}\Big)$, thus implying that such infinitely-wide networks lack representation learning (§6.3.3).

Before we begin, let's start with some reminiscence, recast through the lens of our new Bayesian glasses. In the infinite-width limit, the prior distribution over the network outputs is given by a simple zero-mean Gaussian distribution

$$p\Big(z_{\mathcal{D}}^{(L)}\Big|\mathcal{H}\Big) = \frac{1}{\sqrt{|2\pi K|^{n_L}}}\exp\left(-\frac{1}{2}\sum_{i=1}^{n_L}\sum_{\delta_1,\delta_2\in\mathcal{D}} K^{\delta_1\delta_2} z_{i;\delta_1}^{(L)} z_{i;\delta_2}^{(L)}\right) , \tag{6.32}$$

with the variance $K_{\delta_1\delta_2} \equiv K^{(L)}_{\delta_1\delta_2} = K^{(L)}(x_{\delta_1}, x_{\delta_2})$ given by the kernel at the output layer – here with the layer index dropped – depending explicitly on pairs of inputs x_{δ_1} and x_{δ_2} from the dataset $\mathcal{D}$ and implicitly on the $\mathcal{H}$yperparameters C_b and C_W. Also recall that, as per our *general relativistic* conventions, the matrix $K^{\delta_1\delta_2}$ is the *inverse* of the covariance matrix $K_{\delta_1\delta_2}$

$$\sum_{\delta_2\in\mathcal{D}} K^{\delta_1\delta_2} K_{\delta_2\delta_3} = \delta^{\delta_1}_{\ \delta_3} , \tag{6.33}$$

[19] Recall from §2.1 that CNNs (2.8) are designed to capitalize on the fact that computer vision data organizes useful information in a spatially-local translationally-invariant manner. Incorporating this property into the architecture design is an inductive bias of the CNN; in particular, the assumption is that a cat is still a `cat`, even if it's shifted *up up down down left right left right BA*. The advantage of such an inductive bias as compared to MLPs should be directly encoded in a Bayes' factor $p\big(y_{\mathcal{A}}\big|\mathcal{H}_{\text{CNN}}\big)\,/p\big(y_{\mathcal{A}}\big|\mathcal{H}_{\text{MLP}}\big)$. This ratio should presumably be greater than one for any dataset with desired outputs $y_{\mathcal{A}}$ for which the assumption of spatial locality is a useful inductive bias.

where we are entertained by – but also apologize for – the collision of sample indices δ_1, δ_2 with the overall Kronecker delta, and further recall that $|2\pi K|$ is the determinant of the $N_{\mathcal{D}}$-by-$N_{\mathcal{D}}$ matrix $(2\pi K)_{\delta_1\delta_2}$.

6.3.1 The Evidence for Criticality

As we elaborated on in the last section, the evidence is just the prior distribution for the network outputs evaluated on the observed true output values $y_{i;\tilde{\alpha}}$ given the inputs $x_{i;\tilde{\alpha}}$ in the subsample $\mathcal{A}$:

$$p(y_{\mathcal{A}}|\mathcal{H}) = \frac{1}{\sqrt{\left|2\pi\widetilde{K}\right|^{n_L}}} \exp\left(-\frac{1}{2}\sum_{i=1}^{n_L}\sum_{\tilde{\alpha}_1,\tilde{\alpha}_2\in\mathcal{A}} \widetilde{K}^{\tilde{\alpha}_1\tilde{\alpha}_2} y_{i;\tilde{\alpha}_1} y_{i;\tilde{\alpha}_2}\right) . \tag{6.34}$$

Here we've put tildes both on the sample indices $\tilde{\alpha}$ and on the kernel as well, $\widetilde{K}_{\tilde{\alpha}_1\tilde{\alpha}_2}$, in order to indicate that it's an $N_{\mathcal{A}}$-by-$N_{\mathcal{A}}$ submatrix built from the pairs of inputs $(x_{\tilde{\alpha}_1}, x_{\tilde{\alpha}_2})$ in the subsample $\mathcal{A}$ of size $N_{\mathcal{A}}$. Importantly, this means that the inverse $\widetilde{K}^{\tilde{\alpha}_1\tilde{\alpha}_2}$ is taken with respect to the samples in the set $\mathcal{A}$ *only*,

$$\sum_{\tilde{\alpha}_2\in\mathcal{A}} \widetilde{K}^{\tilde{\alpha}_1\tilde{\alpha}_2}\widetilde{K}_{\tilde{\alpha}_2\tilde{\alpha}_3} = \delta^{\tilde{\alpha}_1}_{\ \tilde{\alpha}_3} , \tag{6.35}$$

and in particular that $\widetilde{K}^{\tilde{\alpha}_1\tilde{\alpha}_2} \neq K^{\tilde{\alpha}_1\tilde{\alpha}_2}$. In other words, $\widetilde{K}^{\tilde{\alpha}_1\tilde{\alpha}_2}$ is *not* the same as the inverse of the kernel $K^{\delta_1\delta_2}$ on the whole dataset $\mathcal{D}$ (6.33) evaluated on the sample indices $\delta_1 = \tilde{\alpha}_1$ and $\delta_2 = \tilde{\alpha}_2$; if you'd like, flip ahead and cf. (6.53). Accordingly, the determinant $\left|2\pi\widetilde{K}\right|$ is also computed from this $N_{\mathcal{A}}$-by-$N_{\mathcal{A}}$ submatrix. The usefulness of this notation and the essentialness of this distinction will become clearer when we consider the posterior in §6.3.2.

Before we analyze the evidence (6.34) in detail, we should establish our space of hypotheses. Considering MLP architectures in the infinite-width limit, there's only three hyperparameters of relevance, the bias variance and rescaled weight variance C_b and C_W, and the depth L. In principle, each combination of these three hyperparameters is a different hypothesis. However, in the asymptotic limit of large depth $L \gg 1$, we know from our discussion in §3.2 and our analysis in §5 that generically the kernel recursion will either exponentially lead to a *trivial fixed point* at zero $K^\star = 0$ or at infinity $K^\star = \infty$, or slowly approach a *nontrivial fixed point* at criticality.[20] Thus, for deep networks Bayesian model comparison essentially reduces to the comparison of three different hypotheses, $\mathcal{H}_0$, $\mathcal{H}_\infty$ and $\mathcal{H}_{\text{critical}}$, corresponding to the two trivial fixed points and the one nontrivial fixed point, respectively.

[20] Yes, we know, for some activation functions there exist hyperparameter settings that lead to trivial fixed points at nonzero values of the kernel $K^\star \neq 0$. We'll eventually consider – and make a case against – such hypotheses as well, though only in a future footnote, 23, and only after first considering the details of the two-input evidence.

Having established our space of hypotheses, let's first see how Bayesian model comparison works when we have only a single input x. In this case the kernel is just a scalar, and the evidence is simply given by

$$p(y|\mathcal{H}) = \frac{1}{\left(2\pi\widetilde{K}\right)^{\frac{n_L}{2}}} \exp\left(-\frac{1}{2\widetilde{K}}\sum_{i=1}^{n_L} y_i^2\right) . \tag{6.36}$$

Here, the output norm $\sum_{i=1}^{n_L} y_i^2$ is fixed by a given function approximation task.[21] Thus all the dependence on the hyperparameters $\mathcal{H}$ is encoded in a single number: $\widetilde{K}$.

Let's start with $\mathcal{H}_\infty$, for which $\widetilde{K} \to \infty$. In this case, the argument of the exponential in (6.36) vanishes and thus the exponential evaluates to unity, while the normalization factor in front vanishes. Therefore, the evidence will vanish polynomially:

$$p(y|\mathcal{H}_\infty) = \lim_{\widetilde{K}\to\infty} \frac{1}{\left(2\pi\widetilde{K}\right)^{\frac{n_L}{2}}} = 0 . \tag{6.37}$$

In fact, in this limit the output distribution becomes an (unnormalizable) uniform distribution over all possible output norms. Next, let's consider $\mathcal{H}_0$ with $\widetilde{K} \to 0$. In this case, while the normalization factor grows polynomially, the argument in the exponent approaches negative infinity. Thus, the evidence approaches zero exponentially quickly:

$$p(y|\mathcal{H}_0) = \lim_{\widetilde{K}\to 0} \exp\left[-\frac{1}{2\widetilde{K}}\sum_{i=1}^{n_L} y_i^2 + O\Big(\log\widetilde{K}\Big)\right] = 0 . \tag{6.38}$$

Indeed, recalling (2.30), the evidence (6.36) in this limit becomes a Dirac delta function,

$$p(y|\mathcal{H}_0) = \prod_{i=1}^{n_L} \delta(y_i) , \tag{6.39}$$

which is a fairly useless hypothesis unless all of the true outputs are the zero vector. Therefore, for generic nonzero and finite output values, the maximal evidence should lie between these two extrema. Specifically, seen as a function of $\widetilde{K}$, the evidence (6.36) peaks at $\widetilde{K} = \widetilde{K}^{(L)}(x,x) \equiv \sum_{i=1}^{n_L} y_i^2/n_L$. Our remaining hypothesis, criticality $\mathcal{H}_{\text{critical}}$, comes the closest to realizing this maximum.

To reiterate, for a single input we just need the kernel $\widetilde{K}$ to be of order one. For deep neural networks, this is precisely the condition that we imposed in order to avoid the *exploding and vanishing kernel problem* for a single input, which we satisfied with the parallel susceptibility condition $\chi_\parallel(K^\star) = 1$. Physically, the exploding kernel gives a very flat distribution spread over a big range of output norms, yielding insubstantial evidence for any particular output norm; the vanishing kernel gives sharp support for

[21] Many common datasets for *classification* tasks employ "one-hot" true outputs in which all but one component y_i of a particular output are zero, and the remaining single component – corresponding to the *correct* class – is equal to one. For such datasets, the output norm is trivial $\sum_{i=1}^{n_L} y_i^2 = 1$.

the zero norm (6.39) and no support anywhere else. Clearly the Bayes' factor (6.31) will prefer any hypothesis that gives more focused support over reasonable output norms. In the language of our Occam's razor discussion, $\mathcal{H}_\infty$ is too complex, predicting every possible norm, while $\mathcal{H}_0$ is too simple, predicting only one particular norm. The only hypothesis that gives a finite and nonzero $\widetilde{K}$ in the deep asymptotic regime is $\mathcal{H}_{\text{critical}}$, whereat the initialization hyperparameters are tuned to satisfy $\chi_\parallel(K^\star) = 1$.[22]

Now that we see how this works, let's extend our analysis of the evidence to two inputs, with $\tilde{\alpha} = \pm$. Intuitively, we expect to find the perpendicular susceptibility condition $\chi_\perp(K^\star) = 1$ and thus demonstrate a conclusive preference for the criticality hypothesis $\mathcal{H}_{\text{critical}}$. To rediscover $\chi_\perp(K^\star) = 1$, it will be sufficient to consider the case where both inputs have the same norm

$$\sum_{i=1}^{n_0} x_{i;+}^2 = \sum_{i=1}^{n_0} x_{i;-}^2 \,. \tag{6.40}$$

Then, recalling our decomposition into the $\gamma^{[a]}_{\tilde{\alpha}_1\tilde{\alpha}_2}$ basis (5.15), we can write the kernel as

$$\widetilde{K}_{\tilde{\alpha}_1\tilde{\alpha}_2} = \begin{pmatrix} \widetilde{K}_{[0]} + \widetilde{K}_{[2]} & \widetilde{K}_{[0]} - \widetilde{K}_{[2]} \\ \widetilde{K}_{[0]} - \widetilde{K}_{[2]} & \widetilde{K}_{[0]} + \widetilde{K}_{[2]} \end{pmatrix} , \tag{6.41}$$

where we've used the fact that $\widetilde{K}_{[1]} = 0$ when both inputs have the same norm (6.40).

In this basis, the determinant is given by $\left|2\pi\widetilde{K}\right| = 16\pi^2\widetilde{K}_{[0]}\widetilde{K}_{[2]}$, and the inverse of the kernel is given by

$$\widetilde{K}^{\tilde{\alpha}_1\tilde{\alpha}_2} = \frac{1}{4\widetilde{K}_{[0]}\widetilde{K}_{[2]}} \begin{pmatrix} \widetilde{K}_{[0]} + \widetilde{K}_{[2]} & -\widetilde{K}_{[0]} + \widetilde{K}_{[2]} \\ -\widetilde{K}_{[0]} + \widetilde{K}_{[2]} & \widetilde{K}_{[0]} + \widetilde{K}_{[2]} \end{pmatrix} , \tag{6.42}$$

which in turn lets us evaluate the argument of the exponential in (6.34) as

$$\begin{aligned} \sum_{i=1}^{n_L} \sum_{\tilde{\alpha}_1,\tilde{\alpha}_2=\pm} \widetilde{K}^{\tilde{\alpha}_1\tilde{\alpha}_2} y_{i;\tilde{\alpha}_1} y_{i;\tilde{\alpha}_2} =& \sum_{i=1}^{n_L} \frac{1}{4\widetilde{K}_{[0]}\widetilde{K}_{[2]}} \left[\widetilde{K}_{[2]} \left(y_{i;+} + y_{i;-}\right)^2 + \widetilde{K}_{[0]} \left(y_{i;+} - y_{i;-}\right)^2\right] \\ =& \frac{\mathbb{Y}_{[0]}}{\widetilde{K}_{[0]}} + \frac{\mathbb{Y}_{[2]}}{\widetilde{K}_{[2]}} \,, \end{aligned} \tag{6.43}$$

where in the last equality we introduced the components

$$\mathbb{Y}_{[0]} \equiv \sum_i^{n_L} \left(\frac{y_{i;+} + y_{i;-}}{2}\right)^2 , \qquad \mathbb{Y}_{[2]} \equiv \sum_i^{n_L} \left(\frac{y_{i;+} - y_{i;-}}{2}\right)^2 . \tag{6.44}$$

[22] N.B. polynomially vanishing kernels give finite evidence for all practical depths. To be very pedantic about this, for such kernels – for instance, for the `tanh` – for absurdly deep networks the truly Bayesian-optimal C_W would be ever so slightly above its critical value.

All together, this gives a simple expression for the two-input evidence,

$$p(y_+, y_-|\mathcal{H}) = \left(16\pi^2 \widetilde{K}_{[0]} \widetilde{K}_{[2]}\right)^{-\frac{n_L}{2}} \exp\left(-\frac{\mathbb{Y}_{[0]}}{2\widetilde{K}_{[0]}} - \frac{\mathbb{Y}_{[2]}}{2\widetilde{K}_{[2]}}\right) \tag{6.45}$$
$$= \left[\left(4\pi \widetilde{K}_{[0]}\right)^{-\frac{n_L}{2}} \exp\left(-\frac{\mathbb{Y}_{[0]}}{2\widetilde{K}_{[0]}}\right)\right] \times \left[\left(4\pi \widetilde{K}_{[2]}\right)^{-\frac{n_L}{2}} \exp\left(-\frac{\mathbb{Y}_{[2]}}{2\widetilde{K}_{[2]}}\right)\right] .$$

Now, let's consider a generic pair of input-output pairs (x_+, y_+) and (x_-, y_-) for which both the average and the difference of the true outputs, $\mathbb{Y}_{[0]}$ and $\mathbb{Y}_{[2]}$ (6.44), are nonzero and of order one. Then, running the same argument as we did for the single-input evidence, we prefer a hypothesis that comes as close as possible to having both $\widetilde{K}_{[0]} \approx \mathbb{Y}_{[0]}/n_L = O(1)$ – from maximizing the object in the first square brackets of (6.45) – and $\widetilde{K}_{[2]} \approx \mathbb{Y}_{[2]}/n_L = O(1)$ – from maximizing the object in the second square brackets of (6.45). And, as we learned in §5, to keep both $\widetilde{K}_{[0]}$ *and* $\widetilde{K}_{[2]}$ of order one, we need to set both the critical parallel susceptibility condition $\chi_\parallel(K^\star) = 1$ *and* the critical perpendicular susceptibility condition $\chi_\perp(K^\star) = 1$.[23] Therefore, with this *evidence for criticality*, Bayesian model comparison demonstrates a full preference for $\mathcal{H}_{\text{critical}}$.[24]

[23]Finally, let's consider the trivial fixed points with nonzero kernel values $K^\star \neq 0$. (This can occur, e.g., for the $K^\star = 0$ universality class, for which there exists fixed points $K^\star$ that have $\chi_\perp(K^\star) = 1$ but $\chi_\parallel(K^\star) < 1$.) For this analysis, we need to relax the same-norm condition (6.40) and consider the most general form of the two-input kernel. Projecting the kernel into the $\gamma^{[a]}_{\tilde{\alpha}_1\tilde{\alpha}_2}$ basis (5.15) as

$$\widetilde{K}_{\tilde{\alpha}_1\tilde{\alpha}_2} = \begin{pmatrix} \widetilde{K}_{[0]} + \widetilde{K}_{[1]} + \widetilde{K}_{[2]} & \widetilde{K}_{[0]} - \widetilde{K}_{[2]} \\ \widetilde{K}_{[0]} - \widetilde{K}_{[2]} & \widetilde{K}_{[0]} - \widetilde{K}_{[1]} + \widetilde{K}_{[2]} \end{pmatrix} , \tag{6.46}$$

we can similarly use (5.20) to decompose the *output matrix*, $\mathbb{Y}_{\tilde{\alpha}_1\tilde{\alpha}_2} \equiv \sum_{i=1}^{n_L} y_{i;\tilde{\alpha}_1} y_{i;\tilde{\alpha}_2}$, into components

$$\mathbb{Y}_{[0]} = \sum_i \left(\frac{y_{i;+} + y_{i;-}}{2}\right)^2 , \quad \mathbb{Y}_{[1]} = \frac{\sum_i y^2_{i;+} - \sum_i y^2_{i;-}}{2} , \quad \mathbb{Y}_{[2]} = \sum_i \left(\frac{y_{i;+} - y_{i;-}}{2}\right)^2 . \tag{6.47}$$

Then, a quick calculations shows that the evidence evaluates to

$$p(y_+, y_-|\mathcal{H}) = \left[4\pi^2(4\widetilde{K}_{[0]}\widetilde{K}_{[2]} - \widetilde{K}^2_{[1]})\right]^{-\frac{n_L}{2}} \exp\left[-\frac{(4\widetilde{K}_{[0]}\mathbb{Y}_{[2]} + 4\widetilde{K}_{[2]}\mathbb{Y}_{[0]} - 2\widetilde{K}_{[1]}\mathbb{Y}_{[1]})}{2(4\widetilde{K}_{[0]}\widetilde{K}_{[2]} - \widetilde{K}^2_{[1]})}\right] . \tag{6.48}$$

Now, we see from this expression that a hypothesis with $\widetilde{K}_{[1]}\mathbb{Y}_{[1]} > 0$ has improved evidence compared to the one with non-positive $\widetilde{K}_{[1]}\mathbb{Y}_{[1]}$. In particular, if a fixed point is trivial then the parallel perturbation $\widetilde{K}_{[1]}$ always vanishes exponentially, even if the fixed-point value of the kernel is non-vanishing $K^\star \neq 0$. Thus, such a hypothesis will be disfavored compared to $\mathcal{H}_{\text{critical}}$, completing our argument.

It should be noted that for this distinction to matter, we must have a nonzero $\mathbb{Y}_{[1]}$, meaning $\sum_i y^2_{i;+} \neq \sum_i y^2_{i;-}$. For networks used as generic function approximators – or for tasks where the network outputs are general and used downstream for other tasks – this may matter. For deep-learning tasks where all the true outputs have the same norm, this may not matter.

[24]Technically, what we've shown here is a preference for criticality in the Bayesian prior distribution. In §9.4, we'll also find a natural preference for criticality in the initialization distribution, by showing that such a tuning is necessary for controlling the exploding and vanishing *gradient* problem that arises with gradient-based learning.

Programming note: since conditioning on $\mathcal{H}$ is so deeply ingrained in our minds by now, for notational simplicity we'll re-start the suppression of this conditioning from here on out.

6.3.2 Let's Not Wire Together

Now, let's work out the full posterior distribution (6.25) at infinite width.[25] As we already have an expression for the evidence $p(y_{\mathcal{A}})$ (6.34) in the denominator, let's focus on the joint distribution $p\Big(y_{\mathcal{A}}, z_{\mathcal{B}}^{(L)}\Big)$ in the numerator. Recall also that to discuss the posterior we need to partition the data into two subsamples, $\mathcal{D} \equiv \mathcal{A} \cup \mathcal{B}$, one for which we have observed the true output values $y_{\mathcal{A}}$ and the other for which we are going to infer the output values.

With such a data partitioning in mind, we can write out the joint distribution as

$$\begin{aligned} &p\Big(y_{\mathcal{A}}, z_{\mathcal{B}}^{(L)}\Big) \\ =&\frac{1}{\sqrt{|2\pi K|^{n_L}}} \exp\Bigg[-\frac{1}{2} \sum_{i=1}^{n_L} \Bigg(\sum_{\tilde{\alpha}_1,\tilde{\alpha}_2 \in \mathcal{A}} K^{\tilde{\alpha}_1\tilde{\alpha}_2} y_{i;\tilde{\alpha}_1} y_{i;\tilde{\alpha}_2} + \sum_{\tilde{\alpha}_1 \in \mathcal{A}, \dot{\beta}_2 \in \mathcal{B}} K^{\tilde{\alpha}_1\dot{\beta}_2} y_{i;\tilde{\alpha}_1} z^{(L)}_{i;\dot{\beta}_2} \\ &\qquad + \sum_{\dot{\beta}_1 \in \mathcal{B}, \tilde{\alpha}_2 \in \mathcal{A}} K^{\dot{\beta}_1\tilde{\alpha}_2} z^{(L)}_{i;\dot{\beta}_1} y_{i;\tilde{\alpha}_2} + \sum_{\dot{\beta}_1,\dot{\beta}_2 \in \mathcal{B}} K^{\dot{\beta}_1\dot{\beta}_2} z^{(L)}_{i;\dot{\beta}_1} z^{(L)}_{i;\dot{\beta}_2} \Bigg) \Bigg], \end{aligned} \tag{6.49}$$

where $K^{\tilde{\alpha}_1\tilde{\alpha}_2}$, $K^{\tilde{\alpha}_1\dot{\beta}_2}$, $K^{\dot{\beta}_1\tilde{\alpha}_2}$, and $K^{\dot{\beta}_1\dot{\beta}_2}$ are the blocks of

$$K^{\delta_1\delta_2} \equiv \begin{pmatrix} K^{\tilde{\alpha}_1\tilde{\alpha}_2} & K^{\tilde{\alpha}_1\dot{\beta}_2} \\ K^{\dot{\beta}_1\tilde{\alpha}_2} & K^{\dot{\beta}_1\dot{\beta}_2} \end{pmatrix}, \tag{6.50}$$

which is the inverse of the whole $N_{\mathcal{D}}$-by-$N_{\mathcal{D}}$ kernel matrix,

$$K_{\delta_1\delta_2} = \begin{pmatrix} \widetilde{K}_{\tilde{\alpha}_1\tilde{\alpha}_2} & K_{\tilde{\alpha}_1\dot{\beta}_2} \\ K_{\dot{\beta}_1\tilde{\alpha}_2} & K_{\dot{\beta}_1\dot{\beta}_2} \end{pmatrix}. \tag{6.51}$$

To make progress, we need to relate the submatrices in the inverse (6.50) to the submatrices in the kernel decomposition (6.51), since, recalling

$$K_{\delta_1\delta_2} \equiv \frac{1}{n_L} \sum_{i}^{n_L} \mathbb{E}\left[z_i^{(L)}(x_{\delta_1})\, z_i^{(L)}(x_{\delta_2}) \right] + O\Big(\frac{1}{n}\Big), \tag{6.52}$$

it's these blocks that are naturally defined in terms of the data.[26]

[25] The form of this distribution was first worked out by Williams in [51] for one-hidden-layer networks.

[26] As we explained before, the over-tilde on $\widetilde{K}_{\tilde{\alpha}_1\tilde{\alpha}_2}$ indicates that it's a submatrix of the kernel evaluated on samples in the set $\mathcal{A}$, only. The inverse of that block was defined explicitly in (6.35) and is symbolized as $\widetilde{K}^{\tilde{\alpha}_1\tilde{\alpha}_2}$. Also note that the symmetry of the full kernel, $K_{\delta_1\delta_2} = K_{\delta_2\delta_1}$, endows a similar set of symmetries on the submatrices: $\widetilde{K}_{\tilde{\alpha}_1\tilde{\alpha}_2} = \widetilde{K}_{\tilde{\alpha}_2\tilde{\alpha}_1}$, $K_{\dot{\beta}_1\dot{\beta}_2} = K_{\dot{\beta}_2\dot{\beta}_1}$, and $K_{\dot{\beta}\tilde{\alpha}} = K_{\tilde{\alpha}\dot{\beta}}$.

Explicitly inverting $K_{\delta_1\delta_2}$ according to the inverse formula (6.33), we find that the submatrices of (6.50) can be defined in terms of the blocks of the kernel (6.51) and the inverse submatrix $\widetilde{K}^{\tilde{\alpha}_1\tilde{\alpha}_2}$ on $\mathcal{A}$ as

$$K^{\tilde{\alpha}_1\tilde{\alpha}_2} \equiv \widetilde{K}^{\tilde{\alpha}_1\tilde{\alpha}_2} + \sum_{\tilde{\alpha}_3,\tilde{\alpha}_4\in\mathcal{A}} \sum_{\dot{\beta}_3,\dot{\beta}_4\in\mathcal{B}} \widetilde{K}^{\tilde{\alpha}_1\tilde{\alpha}_3} K_{\tilde{\alpha}_3\dot{\beta}_3} \mathbb{K}^{\dot{\beta}_3\dot{\beta}_4} K_{\dot{\beta}_4\tilde{\alpha}_4} \widetilde{K}^{\tilde{\alpha}_4\tilde{\alpha}_2} \,, \tag{6.53}$$

$$K^{\tilde{\alpha}_1\dot{\beta}_2} \equiv -\sum_{\tilde{\alpha}_3\in\mathcal{A}} \sum_{\dot{\beta}_3\in\mathcal{B}} \widetilde{K}^{\tilde{\alpha}_1\tilde{\alpha}_3} K_{\tilde{\alpha}_3\dot{\beta}_3} \mathbb{K}^{\dot{\beta}_3\dot{\beta}_2} \,, \tag{6.54}$$

$$K^{\dot{\beta}_1\tilde{\alpha}_2} \equiv -\sum_{\tilde{\alpha}_3\in\mathcal{A}} \sum_{\dot{\beta}_3\in\mathcal{B}} \mathbb{K}^{\dot{\beta}_1\dot{\beta}_3} K_{\dot{\beta}_3\tilde{\alpha}_3} \widetilde{K}^{\tilde{\alpha}_3\tilde{\alpha}_2} \,, \tag{6.55}$$

$$K^{\dot{\beta}_1\dot{\beta}_2} \equiv \mathbb{K}^{\dot{\beta}_1\dot{\beta}_2} \,, \tag{6.56}$$

where we've had to introduce (and name a posteori) the *posterior covariance*,

$$\mathbb{K}_{\dot{\beta}_1\dot{\beta}_2} \equiv K_{\dot{\beta}_1\dot{\beta}_2} - \sum_{\tilde{\alpha}_3,\tilde{\alpha}_4\in\mathcal{A}} K_{\dot{\beta}_1\tilde{\alpha}_3} \widetilde{K}^{\tilde{\alpha}_3\tilde{\alpha}_4} K_{\tilde{\alpha}_4\dot{\beta}_2} \,. \tag{6.57}$$

The expression for (6.56) is defined implicitly by taking the inverse of (6.57):

$$\sum_{\dot{\beta}_2\in\mathcal{B}} \mathbb{K}^{\dot{\beta}_1\dot{\beta}_2} \mathbb{K}_{\dot{\beta}_2\dot{\beta}_3} = \delta^{\dot{\beta}_1}_{\ \dot{\beta}_3} \,. \tag{6.58}$$

Since these are essential relations, let us check all the components of the inverse formula (6.33), one by one. Firstly, considering the $\delta^{\delta_1}_{\ \delta_3} \to \delta^{\tilde{\alpha}_1}_{\ \tilde{\alpha}_3}$ component, we see

$$\begin{aligned} \sum_{\delta_2\in\mathcal{D}} K^{\tilde{\alpha}_1\delta_2} K_{\delta_2\tilde{\alpha}_3} &= \sum_{\tilde{\alpha}_2\in\mathcal{A}} K^{\tilde{\alpha}_1\tilde{\alpha}_2} K_{\tilde{\alpha}_2\tilde{\alpha}_3} + \sum_{\dot{\beta}_2\in\mathcal{B}} K^{\tilde{\alpha}_1\dot{\beta}_2} K_{\dot{\beta}_2\tilde{\alpha}_3} \\ &= \sum_{\tilde{\alpha}_2\in\mathcal{A}} \widetilde{K}^{\tilde{\alpha}_1\tilde{\alpha}_2} \widetilde{K}_{\tilde{\alpha}_2\tilde{\alpha}_3} = \delta^{\tilde{\alpha}_1}_{\ \tilde{\alpha}_3} \,, \end{aligned} \tag{6.59}$$

where in the first line we decomposed the sum over $\delta_2 \in \mathcal{D}$ into separate sums over $\tilde{\alpha}_2 \in \mathcal{A}$ and over $\dot{\beta}_2 \in \mathcal{B}$ according to our partitioning $\mathcal{D} = \mathcal{A} \cup \mathcal{B}$, then in going to the second line we plugged in our expressions for the inverse blocks (6.53) and (6.54), and finally in the last step we used the fact that $\widetilde{K}^{\tilde{\alpha}_1\tilde{\alpha}_2}$ is the inverse of the submatrix $\widetilde{K}_{\tilde{\alpha}_1\tilde{\alpha}_2}$ (6.35). Secondly, considering the $\delta^{\delta_1}_{\ \delta_3} \to \delta^{\dot{\beta}_1}_{\ \dot{\beta}_3}$ component, we see

$$\sum_{\delta_2\in\mathcal{D}} K^{\dot{\beta}_1\delta_2} K_{\delta_2\dot{\beta}_3} = \sum_{\tilde{\alpha}_2\in\mathcal{A}} K^{\dot{\beta}_1\tilde{\alpha}_2} K_{\tilde{\alpha}_2\dot{\beta}_3} + \sum_{\dot{\beta}_2\in\mathcal{B}} K^{\dot{\beta}_1\dot{\beta}_2} K_{\dot{\beta}_2\dot{\beta}_3} \tag{6.60}$$

$$= \sum_{\dot{\beta}_2\in\mathcal{B}} \mathbb{K}^{\dot{\beta}_1\dot{\beta}_2} \left(K_{\dot{\beta}_2\dot{\beta}_3} - \sum_{\tilde{\alpha}_3,\tilde{\alpha}_2\in\mathcal{A}} K_{\dot{\beta}_2\tilde{\alpha}_3} \widetilde{K}^{\tilde{\alpha}_3\tilde{\alpha}_2} K_{\tilde{\alpha}_2\dot{\beta}_3} \right) = \sum_{\dot{\beta}_2\in\mathcal{B}} \mathbb{K}^{\dot{\beta}_1\dot{\beta}_2} \mathbb{K}_{\dot{\beta}_2\dot{\beta}_3} = \delta^{\dot{\beta}_1}_{\ \dot{\beta}_3} \,,$$

where as before in the first line we decomposed the sum over $\delta_2 \in \mathcal{D}$ into separate sums over $\tilde{\alpha}_2 \in \mathcal{A}$ and over $\dot{\beta}_2 \in \mathcal{B}$ according to our partitioning $\mathcal{D} = \mathcal{A} \cup \mathcal{B}$, then in going to

the second line we plugged in our expressions for the inverse blocks (6.55) and (6.56), and finally, identifying the expression in the parenthesis as the definition of the posterior covariance $\mathbb{K}_{\dot\beta_1\dot\beta_2}$ (6.57), we get the final result since $\mathbb{K}^{\dot\beta_1\dot\beta_2}$ is the inverse of the posterior covariance (6.58). Lastly, we consider the off-diagonal block:

$$\begin{aligned}\sum_{\delta_2\in\mathcal{D}} K^{\tilde\alpha_1\delta_2}K_{\delta_2\dot\beta_3} &= \sum_{\tilde\alpha_2\in\mathcal{A}} K^{\tilde\alpha_1\tilde\alpha_2}K_{\tilde\alpha_2\dot\beta_3} + \sum_{\dot\beta_2\in\mathcal{B}} K^{\tilde\alpha_1\dot\beta_2}K_{\dot\beta_2\dot\beta_3} \qquad (6.61)\\ &= \sum_{\tilde\alpha_2\in\mathcal{A},\dot\beta_2\in\mathcal{B}} \widetilde{K}^{\tilde\alpha_1\tilde\alpha_2}K_{\tilde\alpha_2\dot\beta_2}\left(\delta^{\dot\beta_2}_{\ \dot\beta_3} + \sum_{\tilde\alpha_3,\tilde\alpha_4\in\mathcal{A}}\sum_{\dot\beta_4\in\mathcal{B}} \mathbb{K}^{\dot\beta_2\dot\beta_4}K_{\dot\beta_4\tilde\alpha_4}\widetilde{K}^{\tilde\alpha_4\tilde\alpha_3}K_{\tilde\alpha_3\dot\beta_3} - \sum_{\dot\beta_4\in\mathcal{B}}\mathbb{K}^{\dot\beta_2\dot\beta_4}K_{\dot\beta_4\dot\beta_3}\right)\\ &= \sum_{\tilde\alpha_2\in\mathcal{A},\dot\beta_2\in\mathcal{B}} \widetilde{K}^{\tilde\alpha_1\tilde\alpha_2}K_{\tilde\alpha_2\dot\beta_2}\left(\delta^{\dot\beta_2}_{\ \dot\beta_3} - \sum_{\dot\beta_4\in\mathcal{B}}\mathbb{K}^{\dot\beta_2\dot\beta_4}\mathbb{K}_{\dot\beta_4\dot\beta_3}\right) = 0\,,\end{aligned}$$

Here, we follow the same pattern as before, *(i)* decomposing the sum according to the partitioning $\mathcal{D}=\mathcal{A}\cup\mathcal{B}$, *(ii)* plugging in expressions for inverse blocks (6.53) and (6.54), and *(iii)* using the posterior covariance (6.57) and the inverse equation (6.58). Everything checks out.

Now that we have some confidence in our inversions, let's plug our expressions for these submatrices (6.53)–(6.56) into the joint prior (6.49). Since the posterior (6.25) is only a function of the outputs $z^{(L)}_{\mathcal{B}}$, we can make things easier by limiting our focus to the $z^{(L)}_{\mathcal{B}}$ dependence only, ignoring the $y_{\mathcal{A}}$ terms independent of $z^{(L)}_{\mathcal{B}}$ and ignoring the normalization factor:

$$\begin{aligned}p\Big(y_{\mathcal{A}}, z^{(L)}_{\mathcal{B}}\Big) \propto \exp\Bigg[&-\frac{1}{2}\sum_{i=1}^{n_L}\sum_{\dot\beta_1,\dot\beta_2\in\mathcal{B}} \mathbb{K}^{\dot\beta_1\dot\beta_2} z^{(L)}_{i;\dot\beta_1} z^{(L)}_{i;\dot\beta_2} \qquad (6.62)\\ &+\sum_{i=1}^{n_L}\sum_{\dot\beta_1\in\mathcal{B},\tilde\alpha_1\in\mathcal{A}} z^{(L)}_{i;\dot\beta_1}\left(\sum_{\tilde\alpha_2\in\mathcal{A},\dot\beta_2\in\mathcal{B}} \mathbb{K}^{\dot\beta_1\dot\beta_2}K_{\dot\beta_2\tilde\alpha_2}\widetilde{K}^{\tilde\alpha_2\tilde\alpha_1}\right) y_{i;\tilde\alpha_1}\Bigg]\,.\end{aligned}$$

At this point you know what to do: completing the square – as should be your second nature by now – and ignoring the new $z^{(L)}_{\mathcal{B}}$-independent additive constant in the exponential, you get

$$\begin{aligned}p\Big(y_{\mathcal{A}}, z^{(L)}_{\mathcal{B}}\Big) \propto \exp\Bigg[&-\frac{1}{2}\sum_{i=1}^{n_L}\sum_{\dot\beta_1,\dot\beta_2\in\mathcal{B}} \mathbb{K}^{\dot\beta_1\dot\beta_2}\left(z^{(L)}_{i;\dot\beta_1} - \sum_{\tilde\alpha_3,\tilde\alpha_4\in\mathcal{A}} K_{\dot\beta_1\tilde\alpha_3}\widetilde{K}^{\tilde\alpha_3\tilde\alpha_4}y_{i;\tilde\alpha_4}\right) \qquad (6.63)\\ &\times\left(z^{(L)}_{i;\dot\beta_2} - \sum_{\tilde\alpha_5,\tilde\alpha_6\in\mathcal{A}} K_{\dot\beta_2\tilde\alpha_5}\widetilde{K}^{\tilde\alpha_5\tilde\alpha_6}y_{i;\tilde\alpha_6}\right)\Bigg]\,.\end{aligned}$$

This distribution (6.63) is still Gaussian, with a variance given by the posterior covariance $\mathbb{K}_{\dot\beta_1\dot\beta_2}$ and a *nonzero* posterior mean:

$$m^{\infty}_{i;\dot\beta} \equiv \sum_{\tilde\alpha_1,\tilde\alpha_2\in\mathcal{A}} K_{\dot\beta\tilde\alpha_1}\widetilde{K}^{\tilde\alpha_1\tilde\alpha_2}y_{i;\tilde\alpha_2}\,. \qquad (6.64)$$

Here, the superscript ∞ is used to remind us that we're in the infinite-width limit. Finally, we realize that the posterior distribution (6.25) is proportional to the joint prior (6.63),

$$p\Big(z_{\mathcal{B}}^{(L)}\Big|y_{\mathcal{A}}\Big) \propto p\Big(y_{\mathcal{A}}, z_{\mathcal{B}}^{(L)}\Big) \ , \tag{6.65}$$

and that the posterior distribution is automatically normalized (6.5) as a function of the variable $z_{\mathcal{B}}^{(L)}$. Thus, computing the normalization factor for (6.63) – or really just writing it down, since at this point you know by heart how to normalize any Gaussian distribution – we get the posterior at infinite width:

$$p\Big(z_{\mathcal{B}}^{(L)}\Big|y_{\mathcal{A}}\Big) = \frac{1}{\sqrt{|2\pi\mathbb{K}|^{n_L}}} \exp\left[-\frac{1}{2}\sum_{i=1}^{n_L}\sum_{\dot\beta_1,\dot\beta_2\in\mathcal{B}} \mathbb{K}^{\dot\beta_1\dot\beta_2}\left(z_{i;\dot\beta_1}^{(L)} - m_{i;\dot\beta_1}^{\infty}\right)\left(z_{i;\dot\beta_2}^{(L)} - m_{i;\dot\beta_2}^{\infty}\right)\right] . \tag{6.66}$$

The *posterior mean* $m_{i;\dot\beta}^{\infty}$ represents our updated belief about the expected network output for the input $x_{j;\dot\beta} \in \mathcal{B}$ after incorporating information about the true outputs $y_{\mathcal{A}}$ for all the inputs $x_{j;\tilde\alpha} \in \mathcal{A}$; as such, it is explicitly a function of the true input-output pairs $x_{\mathcal{A}}$ and $y_{\mathcal{A}}$ in the subsample $\mathcal{A}$, as we see in (6.64). Importantly, our expected predictions were a priori zero – indicating an inductive bias towards vanishing outputs on average – and now a posteriori our predictions are shifted to something nonzero. Such a nonzero posterior mean is a signature that learning is (finally!) happening. In addition, the posterior covariance $\mathbb{K}_{\dot\beta_1\dot\beta_2}$ encodes the confidence interval: the smaller the covariance is, the more sharply peaked the posterior is around its mean, and the more confident the model is about its predictions.

Practically speaking, note that in order to compute the mean prediction $m_{i;\dot\beta_1}^{\infty}$ according to its definition (6.64), we'd in principle need to invert – and then represent – the $N_{\mathcal{A}}$-by-$N_{\mathcal{A}}$ submatrix $\widetilde{K}_{\tilde\alpha_1\tilde\alpha_2}$. As the size of our observations $N_{\mathcal{A}}$ grows, the computational cost of such an inversion grows very fast.[27] This hidden catch is why – though theoretically quite elegant – (at least any naive implementation of) Bayesian learning is not practical for large datasets. Instead, for this reason we will essentially need to rely on approximation methods for model fitting, such as MLE (6.20). We'll comment more on this next chapter (§7).

Theoretically *and* practically speaking, there is another serious issue with the infinite-width posterior mean. Looking at its expression (6.64), we see that the mean prediction on the output component i is entirely independent from the observations $y_{j;\alpha}$ that we made on the other components with $j \neq i$. Thus, our updated best estimate of these different output components are entirely uncorrelated, though in principle observations of different components j may contain very useful information about a given component

[27]For instance, the computational cost of *Gauss-Jordan elimination* scales as $\sim N_{\mathcal{A}}^3$ and requires us to represent the $N_{\mathcal{A}} \times N_{\mathcal{A}}$-dimensional inverse in memory. Things can be improved a bit by realizing that to compute the posterior mean we only really require the *matrix-vector product* of the inverse with the observations: $\sum_{\tilde\alpha_2\in\mathcal{A}} \widetilde{K}^{\tilde\alpha_1\tilde\alpha_2} y_{i;\tilde\alpha_2}$. However, such an improvement is still not really sufficient for Bayesian learning to compete practically with gradient-based learning for large datasets $\mathcal{A}$.

i.[28] In fact, we see from (6.66) that the posterior distribution actually factorizes as

$$p\Big(z^{(L)}_{i;\mathcal{B}}, z^{(L)}_{j;\mathcal{B}}\Big|y_{i;\mathcal{A}}, y_{j;\mathcal{A}}\Big) = p\Big(z^{(L)}_{i;\mathcal{B}}\Big|y_{i;\mathcal{A}}\Big)\, p\Big(z^{(L)}_{j;\mathcal{B}}\Big|y_{j;\mathcal{A}}\Big)\ , \qquad (i \neq j)\,, \tag{6.67}$$

meaning that the different output components are entirely statistically independent.[29]

We can trace this independence back to a similar property of the infinite-width prior distribution

$$p\Big(z^{(L)}_{i;\mathcal{A}}, z^{(L)}_{j;\mathcal{A}}\Big) = p\Big(z^{(L)}_{i;\mathcal{A}}\Big)\, p\Big(z^{(L)}_{j;\mathcal{A}}\Big)\ , \qquad (i \neq j), \tag{6.68}$$

a property that we've recognized for a while now, see e.g. (5.106). Thus, with Bayesian learning output features do not *wire together*: recalling our discussion of inductive bias before (§6.2.2), we see that the prior endows on the posterior an absurdly stubborn set of beliefs, namely that the components of the output are completely independent with *absolute certainty*. Such an inductive bias is incurable by any amount of learning, irregardless of how large the set of observations $\mathcal{A}$ are; the inductive bias of this prior can never be overwhelmed in the infinite width limit.

Luckily, this state of affiars is completely curable – for both learning algorithms, Bayesian learning and gradient-based learning – by backing off of the infinite-width limit and working with finite-width networks ... the actual kind of networks that are used in practice.

6.3.3 Absence of Representation Learning

Considering the independence of the different components of the output in the posterior, a natural follow-up question is whether or not Bayesian learning at infinite width enables representation learning. Here, we will show decisively that it does *not*.

As a representative avatar of this question, let's compute the *posterior* distribution of preactivations in the penultimate layer $\ell = L-1$ on the full set of samples $\mathcal{D}$, given observations $y_{\mathcal{A}}$:

$$p\Big(z^{(L-1)}_{\mathcal{D}}\Big|y_{\mathcal{A}}\Big) = \frac{p\Big(y_{\mathcal{A}}\Big|z^{(L-1)}_{\mathcal{D}}\Big)\, p\Big(z^{(L-1)}_{\mathcal{D}}\Big)}{p(y_{\mathcal{A}})}\,. \tag{6.69}$$

This is an application of Bayes' rule (6.4), following from applying the product rule (6.1) to the joint distribution $p\Big(y_{\mathcal{A}}, z^{(L-1)}_{\mathcal{D}}\Big)$ between the observations $y_{\mathcal{A}}$ and the penultimate preactivations $z^{(L-1)}_{\mathcal{D}}$. Here, the likelihood $p\Big(y_{\mathcal{A}}\Big|z^{(L-1)}_{\mathcal{D}}\Big)$ is the conditional distribution $p\Big(z^{(L)}_{\mathcal{A}}\Big|z^{(L-1)}_{\mathcal{D}}\Big)$ evaluated on our set of observations $z^{(L)}_{\mathcal{A}} \to y_{\mathcal{A}}$.

[28] The concept of *knowledge distillation* [52] is predicated on this principle of correlations among the output components. For example, if a network is trying to classify images of hand-written digits, a certain example of a "2" may be more "7"-like or more "3"-like. Such feature information is quite useful, especially if the output of the network is used downstream for some other task.

[29] To be FAIR, the issue is with the infinite-width limit itself, as different output components are also decorrelated for infinite-width networks trained with gradient-based learning (§10).

We already know the form of this conditional distribution, as it is the same object (4.69) that we needed in order to work out the layer-to-layer RG flow of the preactivations. In general, this distribution involves the *stochastic* metric $\widehat{G}^{(L)}_{\tilde{\alpha}_1\tilde{\alpha}_2} = \widehat{G}^{(L)}_{\tilde{\alpha}_1\tilde{\alpha}_2}\left(z^{(L-1)}_{\mathcal{D}}\right)$. However, in the infinite-width limit the metric is entirely *deterministic* $\widehat{G}^{(L)}_{\tilde{\alpha}_1\tilde{\alpha}_2} \to G^{(L)}_{\tilde{\alpha}_1\tilde{\alpha}_2}$, with no dependence at all on the penultimate-layer preactivations $z^{(L-1)}_{\mathcal{D}}$. Thus, the likelihood at infinite width – swapping the deterministic metric for the kernel – is given by

$$p\left(y_{\mathcal{A}}\middle|z^{(L-1)}_{\mathcal{D}}\right) = \frac{1}{\sqrt{\left|2\pi\widetilde{K}^{(L)}\right|^{n_L}}} \exp\left(-\frac{1}{2}\sum_{i=1}^{n_L}\sum_{\tilde{\alpha}_1,\tilde{\alpha}_2\in\mathcal{A}} \widetilde{K}^{\tilde{\alpha}_1\tilde{\alpha}_2}_{(L)} y_{i;\tilde{\alpha}_1} y_{i;\tilde{\alpha}_2}\right) = p(y_{\mathcal{A}}) \ , \quad (6.70)$$

and our expression for the posterior of the penultimate layer (6.69) reduces to the prior:

$$p\left(z^{(L-1)}_{\mathcal{D}}\middle|y_{\mathcal{A}}\right) = p\left(z^{(L-1)}_{\mathcal{D}}\right) \ . \quad (6.71)$$

Since the posterior equals the prior, our observation of $y_{\mathcal{A}}$ had no consequence on the penultimate-layer representation; thus, we conclude that there is no representation learning at infinite width.

This lack of representation learning stems from the lack of interlayer correlation in the joint distribution $p\left(z^{(\ell)}_{\mathcal{D}}, z^{(\ell+1)}_{\mathcal{D}}\right)$ at infinite width, and thus it persists for all hidden layers with $\ell < L$. This is another bad inductive bias of the infinite-width hypotheses: regardless of the set of observations $y_{\mathcal{A}}$ that we make, there's no amount of new information that will allow the network to update its representations in the hidden layers $\ell < L$.

This state of affairs is somewhat tragic as the whole point of having many layers – in fact, the main motivation given for deep learning on the whole – is the learning of complex representations in those hidden layers. As we will see next, we can solve this lack of representation learning – as well as the lack of *wiring together* in the output – by backing off the infinite-width limit and looking at finite-width effects.[30]

6.4 Bayesian Inference at Finite Width

In this section, we'll give three lessons on Bayesian learning at finite width. To begin, we'll show that finite-width neural networks are automatically endowed with an inductive bias for neural association due to non-Gaussian interactions between neurons, leading to a natural predisposition towards Hebbian learning (§6.4.1). With that in mind, we'll in turn demonstrate how such learning works by first calculating the mean of the posterior distribution for the network outputs $p\left(z^{(L)}_{\mathcal{B}}\middle|y_{\mathcal{A}}\right)$ – showing how *intralayer* neural

[30] In §10, will also show the same lack of representation learning occurs for the ensemble of infinite-width networks that are (theoretically) trained with gradient-based learning. This issue is also resolved (practically) in §11 by going to finite width.

interactions in the prior give rise to nontrivial correlations among the components of the output (§6.4.2) – and then calculating the posterior distribution of preactivations in the penultimate layer $p\left(z_{\mathcal{B}}^{(L-1)}\middle|y_{\mathcal{A}}\right)$ – showing how *interlayer* interactions give rise to a nonzero shift between prior and posterior, thus signaling the presence of representation learning at finite width (§6.4.3).

6.4.1 Hebbian Learning, Inc.

In this subsection, we'll see that finite-width neural networks have an inductive bias that facilitates *neural association*. To explain **Hebbian learning**, let's begin first with a few words from our honorary guest speaker, Donald Hebb:

> *The general idea is an old one, that any two cells or systems of cells that are repeatedly active at the same time will tend to become "associated," so that activity in one facilitates activity in the other.*
>
> Donald Hebb, in his 1949 classic *The Organization of Behavior* [53].

(*Applause.*)

> *Thank you very much.*
>
> Donald Hebb, apocryphal.

While Hebb was originally thinking about biological neurons, Hebbian learning has become a popular guiding principle for systems of artificial neurons as well. We've actually already seen this inductive bias for neural association any of the numerous times we've discussed the presence of neural interactions in the finite-width prior distribution. To make this manifest, we're now going to explicitly determine the neural influence of one preactivation on another in our effective preactivation distribution at initialization.

Concretely, let's suppose that a single input x is fed into a network, and we've checked that at layer ℓ the value of the first preactivation $z_1^{(\ell)} = \check{z}_1^{(\ell)}$ is larger than typical; given this atypical value $\check{z}_1^{(\ell)}$, we can then ask whether the second preactivation $z_2^{(\ell)}$ is likely to be atypically large. This kind of neural association or influence is encoded in the conditional distribution

$$p\left(z_2^{(\ell)}\middle|\check{z}_1^{(\ell)}\right) = \frac{p\left(\check{z}_1^{(\ell)}, z_2^{(\ell)}\right)}{p\left(\check{z}_1^{(\ell)}\right)} \,. \tag{6.72}$$

Note that at infinite width $p\left(z_2^{(\ell)}\middle|\check{z}_1^{(\ell)}\right) = p\left(z_2^{(\ell)}\right)$ due to the factorization of the prior on neurons (5.106), and so we see right away that there is a complete absence of neural association in such a limit.

To compute this association for finite-width networks, recall from §4.4 the action

representation (4.97) for a distribution over m neurons

$$p(z_1, \ldots, z_m) \propto \exp\left(-\frac{g_m}{2} \sum_{i=1}^{m} z_i^2 + \frac{v}{8} \sum_{i,j=1}^{m} z_i^2 z_j^2 \right) , \tag{6.73}$$

where we have temporarily dropped layer indices from the variables and couplings. Here, the quadratic coupling g_m is given implicitly by the expression (4.102),

$$\frac{1}{g_m} = G^{(\ell)} - \frac{(m+2)}{2n_{\ell-1}} \frac{V^{(\ell)}}{G^{(\ell)}} + O\left(\frac{1}{n^2}\right) , \tag{6.74}$$

and we have emphasized the dependence of the coupling on m; similarly, the quartic coupling is given by (4.103),

$$v = \frac{1}{n_{\ell-1}} \frac{V^{(\ell)}}{\left(G^{(\ell)}\right)^4} + O\left(\frac{1}{n^2}\right) , \tag{6.75}$$

which is independent of m to this order in $1/n$. Evaluating the action representation (6.73) on $m = 1$ and $m = 2$ neurons and plugging the resulting distributions into our expression for the conditional distribution (6.72), we get

$$p(z_2|\check{z}_1) \propto \exp\left[-\frac{g_2}{2} z_2^2 + \frac{v}{8} \left(z_2^4 + 2 z_2^2 \check{z}_1^2 \right) \right] , \tag{6.76}$$

where, similar to the last section, for such a conditional distribution we only need to keep track of the terms in the action that depend on z_2.

Now that we have a conditional distribution, let's evaluate some conditional expectations. Since this distribution is manifestly even in z_2, i.e. invariant under a sign flip $z_2 \leftrightarrow -z_2$, all the odd-point correlators vanish, including the conditional mean. This means that the first nontrivial observable is the two-point correlator or conditional variance:

$$\begin{aligned} \int dz_2 \; p(z_2|\check{z}_1) \, z_2^2 =& \frac{\int dz_2 \; \exp\left[-\frac{g_2}{2} z_2^2 + \frac{v}{8} \left(z_2^4 + 2 z_2^2 \check{z}_1^2 \right) \right] z_2^2}{\int dz_2 \; \exp\left[-\frac{g_2}{2} z_2^2 + \frac{v}{8} \left(z_2^4 + 2 z_2^2 \check{z}_1^2 \right) \right]} \\ =& \frac{\int dz_2 \; e^{-\frac{g_2 z_2^2}{2}} \left[z_2^2 + \frac{v}{8} \left(z_2^6 + 2 z_2^4 \check{z}_1^2 \right) + O(v^2) \right]}{\int dz_2 \; e^{-\frac{g_2 z_2^2}{2}} \left[1 + \frac{v}{8} \left(z_2^4 + 2 z_2^2 \check{z}_1^2 \right) + O(v^2) \right]} \\ =& \frac{g_2^{-1} + \frac{v}{8} \left(15 g_2^{-3} + 6 g_2^{-2} \check{z}_1^2 \right)}{1 + \frac{v}{8} \left(3 g_2^{-2} + 2 g_2^{-1} \check{z}_1^2 \right)} + O\left(v^2\right) \\ =& g_2^{-1} + \frac{v}{2} g_2^{-2} \left(3 g_2^{-1} + \check{z}_1^2 \right) + O\left(v^2\right) . \end{aligned} \tag{6.77}$$

Above, on the first line we used (6.76) in the numerator and at the same time computed its normalization in the denominator, on the second line we expanded both the

numerator and denominator in v, on the third line we computed the single-variable Gaussian integrals, and on the final line we expanded the denominator in v. Plugging in our expressions for the quadratic coupling (6.74) and the quartic coupling (6.75) and reimplementing layer indices, we find

$$\int dz_2^{(\ell)}\; p\left(z_2^{(\ell)}\middle|\check{z}_1^{(\ell)}\right)\left(z_2^{(\ell)}\right)^2 = G^{(\ell)} + \frac{1}{2}\left[\left(\check{z}_1^{(\ell)}\right)^2 - G^{(\ell)}\right]\left[\frac{V^{(\ell)}}{n_{\ell-1}\left(G^{(\ell)}\right)^2}\right] + O\left(\frac{1}{n^2}\right) . \tag{6.78}$$

In passing, note for later that this result holds for any distinct pair of neurons by replacing neural indices as $1, 2 \to i_1, i_2$, with $i_1 \neq i_2$.

This conditional variance (6.78) embodies some really interesting physics. If the observed value $\left(\check{z}_1^{(\ell)}\right)^2$ is larger/smaller than its expected value $\mathbb{E}\left[\left(z_1^{(\ell)}\right)^2\right] = G^{(\ell)}$, then the variance of $z_2^{(\ell)}$ will itself be larger/smaller than is typical. Thus, $z_1^{(\ell)}$ and $z_2^{(\ell)}$ correlate their atypical firing.[31] This effect is proportional to the normalized four-point vertex in the second square brackets of (6.78), which as we know from (5.128) and (5.129) is proportional to ℓ/n across our universality classes when at criticality. In other words, deeper layers have an inductive bias to build more neural associations. Moreover, the presence of these associations is mediated by the interactions in the effective action induced at finite width *only*. As we will soon show, nontrivial representation learning is a direct descendant of such associations.

Note that this result should be interpreted as a *propensity* for atypicality rather than a *guarantee*. Since the conditional variance (6.78) applies to any pair of neurons, conditioned on a particular neuron i_* having a larger/smaller norm than expected, then all of the other neurons with $i \neq i_*$ are more likely to have a larger/smaller norm, though not all will. In a given realization of a network in practice, the ones that happen to have a larger/smaller norm are the ones that are more likely to develop a correlation with i_* as learning progresses.

Hebbian learning is often summarized by the following slogan: *neurons that fire together, wire together*. What we see here is that conditioned on an atypical firing $\check{z}_1$, another preactivation, e.g. z_2, is much more likely to have an atypical firing itself. This propensity of finite-width networks to *fire together* is an inductive bias of our prior beliefs before Bayesian learning as well as of our initialization distribution before gradient-based learning. To understand the *wire together* part, let's now consider the Bayesian posterior.[32]

[31] You may or may not recall from footnote 8 in §1.2 that having a nontrivial connected four-point correlator serves as a measure of the potential for outliers. In statistics, for single-variable distributions this is called the *excess kurtosis*; here, we see a multi-neuron generalization (which apparently can be called the *cokurtosis*). In particular, observing an outlying value $z_1^{(\ell)} = \check{z}_1^{(\ell)}$ implies that we are more likely to see outlying values for $z_2^{(\ell)}$ as well. At the end of Appendix A, we'll provide an information-theoretic reformulation of this phenomenon that will also shed further light on how deep a network should be in order to best take advantage of it.

[32] For some models of artificial neurons – such as the Hopfield network – Hebbian learning is often added in *by hand*. For instance, one learning rule for such networks that explicitly implements the

6.4.2 Let's Wire Together

Let's start with some more reminiscing through our now well-adjusted Bayesian lens. Recall from (4.80) that the prior distribution over peractivations is nearly-Gaussian at large-but-finite width:

$$p\Big(z_{\mathcal{D}}^{(L)}\Big) \propto \exp\Bigg[-\frac{1}{2}\sum_{j=1}^{n_L}\sum_{\delta_1,\delta_2\in\mathcal{D}} g^{\delta_1\delta_2} z_{j;\delta_1}^{(L)} z_{j;\delta_2}^{(L)} \tag{6.79}$$
$$+\frac{1}{8}\sum_{j,k=1}^{n_L}\sum_{\delta_1,\ldots,\delta_4\in\mathcal{D}} v^{(\delta_1\delta_2)(\delta_3\delta_4)} z_{j;\delta_1}^{(L)} z_{j;\delta_2}^{(L)} z_{k;\delta_3}^{(L)} z_{k;\delta_4}^{(L)} + \ldots \Bigg] .$$

As a reminder, the quadratic coupling $g^{\delta_1\delta_2} \equiv g_{(L)}^{\delta_1\delta_2}$ (4.81) and the quartic coupling $v^{(\delta_1\delta_2)(\delta_3\delta_4)} \equiv v_{(L)}^{(\delta_1\delta_2)(\delta_3\delta_4)}$ (4.82) depend explicitly on groups of inputs from the dataset $\mathcal{D}$ and implicitly on the $\mathcal{H}$yperparameters C_b and C_W, the widths n_ℓ, and the depth L. As a consequence of the nonzero *intralayer* interaction between different output preactivations in the prior, there will be non-vanishing correlations between the components of the network outputs in the posterior.

As we did at infinite width, we'll start with the prior distribution (6.79) and then obtain the posterior distribution $p\Big(z_{\mathcal{B}}^{(L)}\Big|y_{\mathcal{A}}\Big) \propto p\Big(y_{\mathcal{A}}, z_{\mathcal{B}}^{(L)}\Big)$ by plugging in our observations $z_{i;\tilde{\alpha}}^{(L)} \to y_{i;\tilde{\alpha}}$ and keeping track of the dependence on the remaining variables $z_{i;\dot{\beta}}^{(L)}$. For the quadratic term in the action, with exactly the same set of manipulations as we did in the infinite-width limit (§6.3.2), replacing the inverse kernel with the quadratic coupling at finite width $K^{\delta_1\delta_2} \to g^{\delta_1\delta_2}$, we find

$$\frac{1}{2}\sum_{j=1}^{n_L}\sum_{\delta_1,\delta_2\in\mathcal{D}} g^{\delta_1\delta_2} z_{j;\delta_1}^{(L)} z_{j;\delta_2}^{(L)}\Big|_{z_{i;\tilde{\alpha}}^{(L)}=y_{i;\tilde{\alpha}}} \tag{6.80}$$
$$= \text{constant} + \frac{1}{2}\sum_{j=1}^{n_L}\sum_{\dot{\beta}_1,\dot{\beta}_2\in\mathcal{B}} \mathbb{G}^{\dot{\beta}_1\dot{\beta}_2}\left(z_{j;\dot{\beta}_1}^{(L)} - m_{j;\dot{\beta}_1}\right)\left(z_{j;\dot{\beta}_2}^{(L)} - m_{j;\dot{\beta}_2}\right) ,$$

with the *naive* posterior mean

$$m_{i;\dot{\beta}} \equiv \sum_{\tilde{\alpha}_1,\tilde{\alpha}_2\in\mathcal{A}} g_{\dot{\beta}\tilde{\alpha}_1}\widetilde{g}^{\tilde{\alpha}_1\tilde{\alpha}_2} y_{i;\tilde{\alpha}_2} , \tag{6.81}$$

and the *naive* posterior covariance

$$\mathbb{G}_{\dot{\beta}_1\dot{\beta}_2} \equiv g_{\dot{\beta}_1\dot{\beta}_2} - \sum_{\tilde{\alpha}_3,\tilde{\alpha}_4\in\mathcal{A}} g_{\dot{\beta}_1\tilde{\alpha}_3}\widetilde{g}^{\tilde{\alpha}_3\tilde{\alpha}_4} g_{\tilde{\alpha}_4\dot{\beta}_2} . \tag{6.82}$$

Hebbian principle is updating the weights connecting two neurons i and j as $W_{ij} \propto z_i(x) z_j(x)$ when observing activities $z_i(x)$ and $z_j(x)$ for a given input x.

In contrast, any finite-width feed-forward neural network should automatically incorporate Hebbian learning *by nature*. To underscore this point further, in §∞ we'll perform an analogous computation for a gradient-descent update. Since the prior has the same form as the initialization distribution, we expect that all learned finite-width networks will inc. the Hebbian learning principle automatically, regardless of whether that learning is Bayesian or gradient-based.

We say *naive* here because there are additional corrections we need to consider coming from the the quartic term in the action. Let see explicitly how this works for the posterior mean.

Given the observed true outputs $y_{i;\tilde{\alpha}}$ and the quadratic term (6.80) centered at the naive posterior mean $m_{i;\dot{\beta}}$, it is natural to center ourselves at

$$\Phi_{i;\delta} \equiv \Big(y_{i;\tilde{\alpha}} \ , \ m_{i;\dot{\beta}} \Big) = \Big(y_{i;\tilde{\alpha}} \ , \ \sum_{\tilde{\alpha}_1,\tilde{\alpha}_2 \in \mathcal{A}} g_{\dot{\beta}\tilde{\alpha}_1} \widetilde{g}^{\tilde{\alpha}_1\tilde{\alpha}_2} y_{i;\tilde{\alpha}_2} \Big) \,, \tag{6.83}$$

and define a fluctuating variable $w_{i;\dot{\beta}} \equiv z^{(L)}_{i;\dot{\beta}} - m_{i;\dot{\beta}}$ so that we can plug the decomposition

$$z^{(L)}_{i;\delta} = \Big(z^{(L)}_{i;\tilde{\alpha}} \ , \ z^{(L)}_{i;\dot{\beta}} \Big) \to \Big(y_{i;\tilde{\alpha}} \ , \ m_{i;\dot{\beta}} + w_{i;\dot{\beta}} \Big) = \Phi_{i;\delta} + \Big(0, \ w_{i;\dot{\beta}} \Big) \ , \tag{6.84}$$

into the action (6.79), thus making the partitioning into subsamples $\mathcal{D} = \mathcal{A} \cup \mathcal{B}$ manifest. In terms of this fluctuation, the quadratic term (6.80) takes the form

$$\text{constant} + \frac{1}{2} \sum_{j=1}^{n_L} \sum_{\dot{\beta}_1,\dot{\beta}_2 \in \mathcal{B}} \mathbb{G}^{\dot{\beta}_1\dot{\beta}_2} w_{j;\dot{\beta}_1} w_{j;\dot{\beta}_2} \,, \tag{6.85}$$

and the quartic term can be evaluated as

$$\begin{aligned}
\mathcal{Q}(w) &\equiv \left[\frac{1}{8} \sum_{j,k=1}^{n_L} \sum_{\delta_1,\ldots,\delta_4 \in \mathcal{D}} v^{(\delta_1\delta_2)(\delta_3\delta_4)} z^{(L)}_{j;\delta_1} z^{(L)}_{j;\delta_2} \ z^{(L)}_{k;\delta_3} z^{(L)}_{k;\delta_4} \right] \Bigg|_{z^{(L)}_{i;\tilde{\alpha}} = \Phi_{i;\tilde{\alpha}}; \ z^{(L)}_{i;\dot{\beta}} = \Phi_{i;\dot{\beta}} + w_{i;\dot{\beta}}} \\
&= \text{constant} + \frac{4}{8} \sum_{j} \sum_{\dot{\beta}_1 \in \mathcal{B}} w_{j;\dot{\beta}_1} \left(\sum_{k} \sum_{\delta_1,\delta_2,\delta_3 \in \mathcal{D}} v^{(\dot{\beta}_1\delta_1)(\delta_2\delta_3)} \Phi_{j;\delta_1} \Phi_{k;\delta_2} \Phi_{k;\delta_3} \right) \\
&\quad + \frac{2}{8} \sum_{j} \sum_{\dot{\beta}_1,\dot{\beta}_2 \in \mathcal{B}} w_{j;\dot{\beta}_1} w_{j;\dot{\beta}_2} \left(\sum_{k} \sum_{\delta_1,\delta_2 \in \mathcal{D}} v^{(\dot{\beta}_1\dot{\beta}_2)(\delta_1\delta_2)} \Phi_{k;\delta_1} \Phi_{k;\delta_2} \right) \\
&\quad + \frac{4}{8} \sum_{j,k} \sum_{\dot{\beta}_1,\dot{\beta}_2 \in \mathcal{B}} w_{j;\dot{\beta}_1} w_{k;\dot{\beta}_2} \left(\sum_{\delta_1,\delta_2 \in \mathcal{D}} v^{(\dot{\beta}_1\delta_1)(\dot{\beta}_2\delta_2)} \Phi_{j;\delta_1} \Phi_{k;\delta_2} \right) \\
&\quad + \frac{4}{8} \sum_{j,k} \sum_{\dot{\beta}_1,\dot{\beta}_2,\dot{\beta}_3 \in \mathcal{B}} w_{j;\dot{\beta}_1} w_{j;\dot{\beta}_2} w_{k;\dot{\beta}_3} \left(\sum_{\delta_1 \in \mathcal{D}} v^{(\dot{\beta}_1\dot{\beta}_2)(\dot{\beta}_3\delta)} \Phi_{k;\delta_1} \right) \\
&\quad + \frac{1}{8} \sum_{j,k} \sum_{\dot{\beta}_1,\ldots,\dot{\beta}_4 \in \mathcal{B}} w_{j;\dot{\beta}_1} w_{j;\dot{\beta}_2} \ w_{k;\dot{\beta}_3} w_{k;\dot{\beta}_4} \left(v^{(\dot{\beta}_1\dot{\beta}_2)(\dot{\beta}_3\dot{\beta}_4)} \right) \,.
\end{aligned} \tag{6.86}$$

Given all these expressions, we can finally determine the true posterior mean by

computing the following expectation:

$$\begin{aligned}\int dz_{\mathcal{B}}^{(L)} p\Big(z_{\mathcal{B}}^{(L)}\Big|y_{\mathcal{A}}\Big)\, z_{i;\dot\beta}^{(L)} =& \int dz_{\mathcal{B}}^{(L)} \frac{p\Big(y_{\mathcal{A}}, z_{\mathcal{B}}^{(L)}\Big)}{p(y_{\mathcal{A}})} z_{i;\dot\beta}^{(L)} \\ =& m_{i;\dot\beta} + \int dw_{\mathcal{B}}\, \frac{p(y_{\mathcal{A}},\ m_{\mathcal{B}} + w_{\mathcal{B}})}{p(y_{\mathcal{A}})} w_{i;\dot\beta} \\ =& m_{i;\dot\beta} + \frac{\Big\langle\!\Big\langle w_{i;\dot\beta} e^{\mathcal{Q}(w)} \Big\rangle\!\Big\rangle_{\mathbb{G}}}{\big\langle\!\big\langle e^{\mathcal{Q}(w)} \big\rangle\!\big\rangle_{\mathbb{G}}} \\ =& m_{i;\dot\beta} + \frac{\Big\langle\!\Big\langle w_{i;\dot\beta}\, [1 + \mathcal{Q}(w)] \Big\rangle\!\Big\rangle_{\mathbb{G}}}{\big\langle\!\big\langle 1 + \mathcal{Q}(w) \big\rangle\!\big\rangle_{\mathbb{G}}} + O\Big(v^2\Big) \\ =& m_{i;\dot\beta} + \Big\langle\!\Big\langle w_{i;\dot\beta} \mathcal{Q}(w) \Big\rangle\!\Big\rangle_{\mathbb{G}} + O\Big(v^2\Big)\ , \end{aligned} \tag{6.87}$$

where on the first line we used Bayes' rule for the posterior (6.25), on the second line we inserted our decomposition (6.84) in two places, on the third line we separated out the quartic term in order to rewrite the posterior expectation as a Gaussian expectation with respect to the naive posterior covariance $\mathbb{G}$ divided by the distribution's normalization, on the fourth line we expanded the exponential, and on the final line we used the fact that the fluctuation has zero mean $\Big\langle\!\Big\langle w_{i;\dot\beta} \Big\rangle\!\Big\rangle_{\mathbb{G}} = 0$ in Gaussian expectation. We can now evaluate the remaining Gaussian expectation by plugging in our expression for the quartic term (6.86) and making Wick contractions:

$$\begin{aligned} & m_{i;\dot\beta} + \Big\langle\!\Big\langle w_{i;\dot\beta} \mathcal{Q}\,(w) \Big\rangle\!\Big\rangle_{\mathbb{G}} \\ =& m_{i;\dot\beta} + \frac{1}{2} \sum_{\dot\beta_1 \in \mathcal{B}} \mathbb{G}_{\dot\beta\dot\beta_1} \left(\sum_k \sum_{\delta_1,\delta_2,\delta_3 \in \mathcal{D}} v^{(\dot\beta_1\delta_1)(\delta_2\delta_3)} \Phi_{i;\delta_1} \Phi_{k;\delta_2} \Phi_{k;\delta_3} \right) \\ & + \frac{1}{2} \sum_{\dot\beta_1,\dot\beta_2,\dot\beta_3 \in \mathcal{B}} \Big(n_L \mathbb{G}_{\dot\beta_1\dot\beta_2} \mathbb{G}_{\dot\beta\dot\beta_3} + 2 \mathbb{G}_{\dot\beta\dot\beta_1} \mathbb{G}_{\dot\beta_2\dot\beta_3} \Big) \left(\sum_{\delta_1 \in \mathcal{D}} v^{(\dot\beta_1\dot\beta_2)(\dot\beta_3\delta_1)} \Phi_{i;\delta_1} \right) . \end{aligned} \tag{6.88}$$

Thus, we see that the naive posterior mean (6.81) is further corrected by a number of v-dependent terms.

To extract some physics from this complicated expression, note from the definition of Φ (6.83) that the i-th component of $\Phi_{i;\delta}$ depends on the i-th component of our observation $y_{i;\tilde\alpha}$. This in particular means that the term above $\propto \sum_k \Phi_{i;\delta_1} \Phi_{k;\delta_2} \Phi_{k;\delta_3}$ does incorporate information from all of the components of the observed true outputs. In other words, information from the k-th component of the observed outputs successfully influences the posterior mean prediction on the i-th component for $i \neq k$. This means that at finite width we have a dependence among the components of the posterior outputs

$$p\Big(z_{i;\mathcal{B}}^{(L)}, z_{k;\mathcal{B}}^{(L)}\Big|y_{i;\mathcal{A}}, y_{k;\mathcal{A}}\Big) \neq p\Big(z_{i;\mathcal{B}}^{(L)}\Big|y_{i;\mathcal{A}}\Big)\ p\Big(z_{k;\mathcal{B}}^{(L)}\Big|y_{k;\mathcal{A}}\Big)\ . \tag{6.89}$$

This property of the posterior distribution descends from the nontrivial *fire-together* inductive bias $p\Big(z_i^{(L)}\Big|z_k^{(L)}\Big)$ present in the finite-width prior as discussed in §6.4.1. The dependence among the components of the posterior outputs (6.89) is a signature of our posterior beliefs' learning to *wire together*, and we will see a further manifestation of this when we again consider representation learning in the next section.

Before we move on, we should address practical matters. Practically speaking, it is even more computationally infeasible to evaluate the finite-width predictions of Bayesian learning (6.87) than it was at infinite width. In particular, evaluating the quartic coupling involves first *representing* the four-point vertex – a $N_{\mathcal{A}}\times N_{\mathcal{A}}\times N_{\mathcal{A}}\times N_{\mathcal{A}}$-dimensional tensor – and then multiply contracting it with inverse kernels. Thus, both the cost of computation and the memory requirements of Bayesian learning grow terrifyingly quickly with our observations, i.e. with size of our dataset $\mathcal{A}$. However, please *Don't Panic*: we are getting ever closer to the point where we can show you how gradient-based learning resolves all these practical difficulties at finite width.

6.4.3 Presence of Representation Learning

The fact that the individual components of the finite-width posterior mean prediction can incorporate information from our observations of the other components is suggestive of the idea that these observations might also be used to build up representations in the hidden layers. Here we will show that such *representation learning* actually does occur at finite width as a direct consequence of the nonzero *interlayer* interactions.

Analogous to our parallel subsection at infinite width (§6.3.3), we can investigate representation learning by considering the posterior distribution in the penultimate layer $\ell = L-1$ on the full set of samples $\mathcal{D}$, given observations $y_{\mathcal{A}}$. In particular, to show how the *features* of the penultimate-layer representation evolve, our goal will be to compute the change in the expectation of a penultimate-layer observable $\mathcal{O}\Big(z_{\mathcal{D}}^{(L-1)}\Big)$ taken with respect to the posterior as compared to the expectation taken with respect to the prior

$$\overline{d\mathcal{O}} \equiv \int dz_{\mathcal{D}}^{(L-1)} p\Big(z_{\mathcal{D}}^{(L-1)}\Big|y_{\mathcal{A}}\Big)\,\mathcal{O}\Big(z_{\mathcal{D}}^{(L-1)}\Big) - \int dz_{\mathcal{D}}^{(L-1)} p\Big(z_{\mathcal{D}}^{(L-1)}\Big)\,\mathcal{O}\Big(z_{\mathcal{D}}^{(L-1)}\Big)\,, \tag{6.90}$$

where $p\Big(z_{\mathcal{D}}^{(L-1)}\Big)$ and $p\Big(z_{\mathcal{D}}^{(L-1)}\Big|y_{\mathcal{A}}\Big)$ are the prior and posterior distributions, respectively. This expectation difference was strictly zero in the infinite-width limit since the penultimate-layer posterior was exactly equal to the penultimate-layer prior (6.71). A non-vanishing difference in contrast will mean that the penultimate-layer preactivations are being updated after making observations $y_{\mathcal{A}}$. Such an *update* is a direct avatar of representation learning.

As before, by Bayes' rule we can write the posterior distribution of the penultimate preactivations $z_{\mathcal{D}}^{(L-1)}$ given our observations $y_{\mathcal{A}}$ as

$$p\Big(z_{\mathcal{D}}^{(L-1)}\Big|y_{\mathcal{A}}\Big) = \frac{p\Big(y_{\mathcal{A}}\Big|z_{\mathcal{D}}^{(L-1)}\Big)\,p\Big(z_{\mathcal{D}}^{(L-1)}\Big)}{p(y_{\mathcal{A}})}\,. \tag{6.91}$$

Just as before, the likelihood $p\Big(y_{\mathcal{A}}\Big|z_{\mathcal{D}}^{(L-1)}\Big)$ is the conditional distribution $p\Big(z_{\mathcal{A}}^{(L)}\Big|z_{\mathcal{D}}^{(L-1)}\Big)$ evaluated on our set of observations $z_{\mathcal{A}}^{(L)} \to y_{\mathcal{A}}$. With this expression for the posterior (6.91), we can express the update $\overline{d\mathcal{O}}$ after Bayesian learning as

$$\overline{d\mathcal{O}} = \mathbb{E}\left[\frac{p\Big(y_{\mathcal{A}}\Big|z_{\mathcal{D}}^{(L-1)}\Big)}{p(y_{\mathcal{A}})}\mathcal{O}\Big(z_{\mathcal{D}}^{(L-1)}\Big)\right] - \mathbb{E}\left[\mathcal{O}\Big(z_{\mathcal{D}}^{(L-1)}\Big)\right] . \tag{6.92}$$

As always, the full expectation $\mathbb{E}\left[\,\cdot\,\right]$ is to be evaluated with respect to the *prior* or initialization distribution $p\Big(z_{\mathcal{D}}^{(L-1)}\Big)$; all learning will always be represented explicitly with the insertion of other factors as we did above.

Let's now determine how this insertion, the likelihood-to-evidence ratio

$$\frac{p\Big(y_{\mathcal{A}}\Big|z_{\mathcal{D}}^{(L-1)}\Big)}{p(y_{\mathcal{A}})} , \tag{6.93}$$

depends on the preactivations $z_{\mathcal{D}}^{(L-1)}$. As we pointed out when working through the infinite-width example, we already worked out the form of this likelihood in (4.69) as the conditional distribution between layers. In our current context and notation, the likelihood reads

$$p\Big(y_{\mathcal{A}}\Big|z_{\mathcal{D}}^{(L-1)}\Big) = \frac{1}{\sqrt{\left|2\pi\widehat{G}^{(L)}\right|^{n_L}}} \exp\left(-\frac{1}{2}\sum_{i=1}^{n_L}\sum_{\tilde{\alpha}_1,\tilde{\alpha}_2\in\mathcal{A}} \widehat{G}^{\tilde{\alpha}_1\tilde{\alpha}_2}_{(L)} y_{i;\tilde{\alpha}_1} y_{i;\tilde{\alpha}_2}\right) , \tag{6.94}$$

where as a reminder the *stochastic metric* $\widehat{G}^{(L)}_{\tilde{\alpha}_1\tilde{\alpha}_2} = \widehat{G}^{(L)}_{\tilde{\alpha}_1\tilde{\alpha}_2}\Big(z_{\mathcal{A}}^{(L-1)}\Big)$ depends explicitly on the preactivations in the penultimate layer $z_{i;\mathcal{A}}^{(L-1)}$.[33] Thus, the stochastic metric acts as a coupling here, inducing interlayer interactions between the $(L-1)$-th-layer preactivations and the observations $y_{\mathcal{A}}$. As we will see, this endows the updated distribution over $z_{\mathcal{D}}^{(L-1)}$ with a dependence on $y_{\mathcal{A}}$.

As should be fairly familiar at this point, we can decompose the stochastic metric into a mean and a fluctuation,

$$\widehat{G}^{(L)}_{\tilde{\alpha}_1\tilde{\alpha}_2} \equiv G^{(L)}_{\tilde{\alpha}_1\tilde{\alpha}_2} + \widehat{\Delta G}^{(L)}_{\tilde{\alpha}_1\tilde{\alpha}_2} , \tag{6.95}$$

[33] Strictly speaking, we should really denote the stochastic metric here as $\widehat{\widehat{G}}^{(L)}_{\tilde{\alpha}_1\tilde{\alpha}_2}$ to indicate that we're focusing on the $N_{\mathcal{A}}$-by-$N_{\mathcal{A}}$ submatrix of the full stochastic metric on $\mathcal{D}$, $\widehat{G}^{(L)}_{\delta_1\delta_2}$. It's the matrix inverse of this submatrix $\widehat{\widehat{G}}^{(L)}_{\tilde{\alpha}_1\tilde{\alpha}_2}$ – and not the $(\tilde{\alpha}_1,\tilde{\alpha}_2)$ block of the inverse of the full matrix $\widehat{G}^{(L)}_{\delta_1\delta_2}$ – that appears in (6.94). Since this tilde-with-a-hat looks ridiculous – and since we are already heavily overburdened on the notational front – if you promise to keep this caveat in mind, we'll do everyone a favor and temporarily suppress this tilde.

in terms of which the likelihood (6.94) can be Taylor-expanded à la Schwinger-Dyson as we did before in (4.56) and (4.57). At first nontrivial order, we find for the likelihood-to-evidence ratio (6.93)

$$\begin{aligned}&\frac{p\Big(y_{\mathcal{A}}\Big|z_{\mathcal{D}}^{(L-1)}\Big)}{p(y_{\mathcal{A}})} \\ =&\frac{1}{p(y_{\mathcal{A}})\sqrt{\left|2\pi G^{(L)}\right|^{n_L}}}\left[1+\frac{1}{2}\sum_{\tilde{\alpha}_1,\ldots,\tilde{\alpha}_4\in\mathcal{A}}\widehat{\Delta G}^{(L)}_{\tilde{\alpha}_1\tilde{\alpha}_2}G^{\tilde{\alpha}_1\tilde{\alpha}_3}_{(L)}G^{\tilde{\alpha}_2\tilde{\alpha}_4}_{(L)}\sum_{i=1}^{n_L}\Big(y_{i;\tilde{\alpha}_3}y_{i;\tilde{\alpha}_4}-G^{(L)}_{\tilde{\alpha}_3\tilde{\alpha}_4}\Big)+O\Big(\Delta^2\Big)\right].\end{aligned} \tag{6.96}$$

Here, the prefactor before the square brackets is constant with respect to the variables $z_{\mathcal{D}}^{(L-1)}$, and so all of the relevant dependence needed to evaluate update $\overline{d\mathcal{O}}$ (6.92) is contained implicitly in the metric fluctuation $\widehat{\Delta G}^{(L)}_{\tilde{\alpha}_1\tilde{\alpha}_2}$. We can thus compute a posterior expectation – i.e. the first expectation in (6.92) – of any observable by integrating against the quantity in the square bracket, so long as we also divide by an integral of "1" against the same quantity in order to properly normalize. With this by-now familiar trick in mind, we can rewrite the posterior expectation as

$$\begin{aligned}&\frac{\mathbb{E}\left\{\mathcal{O}\Big(z_{\mathcal{D}}^{(L-1)}\Big)\left[1+\frac{1}{2}\sum_{\tilde{\alpha}_1,\ldots,\tilde{\alpha}_4}\widehat{\Delta G}^{(L)}_{\tilde{\alpha}_1\tilde{\alpha}_2}G^{\tilde{\alpha}_1\tilde{\alpha}_3}_{(L)}G^{\tilde{\alpha}_2\tilde{\alpha}_4}_{(L)}\sum_i\Big(y_{i;\tilde{\alpha}_3}y_{i;\tilde{\alpha}_4}-G^{(L)}_{\tilde{\alpha}_3\tilde{\alpha}_4}\Big)+O(\Delta^2)\right]\right\}}{\mathbb{E}\left[1+\frac{1}{2}\sum_{\tilde{\alpha}_1,\ldots,\tilde{\alpha}_4}\widehat{\Delta G}^{(L)}_{\tilde{\alpha}_1\tilde{\alpha}_2}G^{\tilde{\alpha}_1\tilde{\alpha}_3}_{(L)}G^{\tilde{\alpha}_2\tilde{\alpha}_4}_{(L)}\sum_i\Big(y_{i;\tilde{\alpha}_3}y_{i;\tilde{\alpha}_4}-G^{(L)}_{\tilde{\alpha}_3\tilde{\alpha}_4}\Big)+O(\Delta^2)\right]} \\ =&\mathbb{E}\left[\mathcal{O}\Big(z_{\mathcal{D}}^{(L-1)}\Big)\right] \\ &+\frac{1}{2}\sum_{\tilde{\alpha}_1,\ldots,\tilde{\alpha}_4\in\mathcal{A}}\mathbb{E}\left[\widehat{\Delta G}^{(L)}_{\tilde{\alpha}_1\tilde{\alpha}_2}\mathcal{O}\Big(z_{\mathcal{D}}^{(L-1)}\Big)\right]G^{\tilde{\alpha}_1\tilde{\alpha}_3}_{(L)}G^{\tilde{\alpha}_2\tilde{\alpha}_4}_{(L)}\sum_i\Big(y_{i;\tilde{\alpha}_3}y_{i;\tilde{\alpha}_4}-G^{(L)}_{\tilde{\alpha}_3\tilde{\alpha}_4}\Big)+O\Big(\frac{1}{n^2}\Big),\end{aligned} \tag{6.97}$$

where the details of what we actually did are hidden in this here footnote.[34] We see that the first term is just the prior expectation, while the second term expresses the update $\overline{d\mathcal{O}}$ (6.90). Finally, taking only the leading finite-width corrections at order $1/n$ and restoring the tildes to correctly represent the submatrices on $\mathcal{A}$ alone, we can write down a very general expression for the update to any penultimate-layer observable at

[34]The reason that we treated the additional $O\big(\Delta^2\big)$ pieces as $O\big(1/n^2\big)$ is hidden under the rug in the main body. To peak under that rug, first let us schematically express the likelihood-to-evidence ratio (6.96) as constant $\times\left[1+\sharp_1\Delta G+\sharp_2(\Delta G)^2+O\big(\Delta^3\big)\right]$. Then, the posterior expectation becomes

$$\begin{aligned}&\frac{\mathbb{E}\left\{\mathcal{O}\left[1+\sharp_1\Delta G+\sharp_2(\Delta G)^2+O\big(\Delta^3\big)\right]\right\}}{\mathbb{E}\left[1+\sharp_1\Delta G+\sharp_2(\Delta G)^2+O(\Delta^3)\right]}=\frac{\mathbb{E}\left[\mathcal{O}\right]+\sharp_1\mathbb{E}\left[\mathcal{O}\Delta G\right]+\sharp_2\mathbb{E}\left[(\Delta G)^2\,\mathcal{O}\right]+O\big(1/n^2\big)}{1+\sharp_2\mathbb{E}\left[(\Delta G)^2\right]+O(1/n^2)} \\ =&\mathbb{E}\left[\mathcal{O}\right]+\sharp_1\mathbb{E}\left[\mathcal{O}\Delta G\right]+\sharp_2\left\{\mathbb{E}\left[(\Delta G)^2\,\mathcal{O}\right]-\mathbb{E}\left[(\Delta G)^2\right]\mathbb{E}\left[\mathcal{O}\right]\right\}+O\big(1/n^2\big)\,.\end{aligned} \tag{6.98}$$

Decomposing the observable into a mean and a fluctuation as $\mathcal{O}=\mathbb{E}\left[\mathcal{O}\right]+\Delta\mathcal{O}$, we see that the term proportional to the coefficient $\sharp_2$ is $\mathbb{E}\left[O\big(\Delta^3\big)\right]=O\big(1/n^2\big)$ and thus can be neglected, while the leading finite-width correction cannot be neglected: $\sharp_1\mathbb{E}\left[\mathcal{O}\Delta G\right]=\sharp_1\mathbb{E}\left[\Delta\mathcal{O}\Delta G\right]=O(1/n)$.

leading nontrivial order in $1/n$:

$$\overline{d\mathcal{O}} = \frac{1}{2} \sum_{\tilde{\alpha}_1,\ldots,\tilde{\alpha}_4 \in \mathcal{A}} \mathbb{E}\left[\widehat{\Delta G}^{(L)}_{\tilde{\alpha}_1\tilde{\alpha}_2} \mathcal{O}\left(z_{\mathcal{D}}^{(L-1)}\right)\right] \widetilde{K}^{\tilde{\alpha}_1\tilde{\alpha}_3}_{(L)} \widetilde{K}^{\tilde{\alpha}_2\tilde{\alpha}_4}_{(L)} \sum_i^{n_L} \left(y_{i;\tilde{\alpha}_3} y_{i;\tilde{\alpha}_4} - \widetilde{K}^{(L)}_{\tilde{\alpha}_3\tilde{\alpha}_4}\right) . \tag{6.99}$$

Again, please be careful and remember that the $\mathbb{E}\left[\,\cdot\,\right]$ in (6.99) is to be evaluated with respect to the prior distribution $p\left(z_{\mathcal{D}}^{(L-1)}\right)$. Note also that the lone expectation in the update (6.99) is just the covariance of the stochastic metric with the observable:

$$\mathbb{E}\left[\widehat{\Delta G}^{(L)}_{\tilde{\alpha}_1\tilde{\alpha}_2} \mathcal{O}\left(z_{\mathcal{D}}^{(L-1)}\right)\right] = \mathbb{E}\left[\widehat{G}^{(L)}_{\tilde{\alpha}_1\tilde{\alpha}_2} \mathcal{O}\left(z_{\mathcal{D}}^{(L-1)}\right)\right] - \mathbb{E}\left[\widehat{G}^{(L)}_{\tilde{\alpha}_1\tilde{\alpha}_2}\right] \mathbb{E}\left[\mathcal{O}\left(z_{\mathcal{D}}^{(L-1)}\right)\right] . \tag{6.100}$$

As we addressed in that rugly footnote, for a general order-one observable this covariance is $1/n$-suppressed but nonzero. Thus, we see that at large-but-finite width ($1 \ll n < \infty$), such observables get updated: representations are learned.

In order to see how this works, let's consider a concrete example. The simplest observable turns out to be the average norm of the activations

$$\mathcal{O}\left(z_{\mathcal{D}}^{(L-1)}\right) \equiv \frac{1}{n_{L-1}} \sum_{j=1}^{n_{L-1}} \sigma^{(L-1)}_{j;\delta_1} \sigma^{(L-1)}_{j;\delta_2} , \tag{6.101}$$

which we can decompose in terms of a mean and a fluctuation as

$$\mathcal{O}\left(z_{\mathcal{D}}^{(L-1)}\right) = \mathbb{E}\left[\mathcal{O}\left(z_{\mathcal{D}}^{(L-1)}\right)\right] + \frac{1}{C_W^{(L)}} \widehat{\Delta G}^{(L)}_{\delta_1\delta_2} , \tag{6.102}$$

if we also recall the explicit form of the metric fluctuation (4.74)

$$\widehat{\Delta G}^{(L)}_{\tilde{\alpha}_1\tilde{\alpha}_2} = C_W^{(L)} \frac{1}{n_{L-1}} \sum_{j=1}^{n_{L-1}} \left(\sigma^{(L-1)}_{j;\tilde{\alpha}_1} \sigma^{(L-1)}_{j;\tilde{\alpha}_2} - \mathbb{E}\left[\sigma^{(L-1)}_{j;\tilde{\alpha}_1} \sigma^{(L-1)}_{j;\tilde{\alpha}_2}\right]\right) . \tag{6.103}$$

Then, plugging into our expression for the leading-order finite-width update (6.99), we find

$$\begin{aligned}
\overline{d\mathcal{O}} =& \frac{1}{2C_W^{(L)}} \sum_{\tilde{\alpha}_1,\ldots,\tilde{\alpha}_4 \in \mathcal{A}} \mathbb{E}\left[\widehat{\Delta G}^{(L)}_{\delta_1\delta_2} \widehat{\Delta G}^{(L)}_{\tilde{\alpha}_1\tilde{\alpha}_2}\right] \widetilde{K}^{\tilde{\alpha}_1\tilde{\alpha}_3}_{(L)} \widetilde{K}^{\tilde{\alpha}_2\tilde{\alpha}_4}_{(L)} \sum_i \left(y_{i;\tilde{\alpha}_3} y_{i;\tilde{\alpha}_4} - \widetilde{K}^{(L)}_{\tilde{\alpha}_3\tilde{\alpha}_4}\right) \\
=& \frac{1}{2n_{L-1}C_W^{(L)}} \sum_{\tilde{\alpha}_1,\ldots,\tilde{\alpha}_4 \in \mathcal{A}} V^{(L)}_{(\delta_1\delta_2)(\tilde{\alpha}_1\tilde{\alpha}_2)} \widetilde{K}^{\tilde{\alpha}_1\tilde{\alpha}_3}_{(L)} \widetilde{K}^{\tilde{\alpha}_2\tilde{\alpha}_4}_{(L)} \sum_i \left(y_{i;\tilde{\alpha}_3} y_{i;\tilde{\alpha}_4} - \widetilde{K}^{(L)}_{\tilde{\alpha}_3\tilde{\alpha}_4}\right) ,
\end{aligned} \tag{6.104}$$

where to go to the second line we used the definition of the four-point vertex in terms of the two-point function of the metric fluctuation (4.76). As this vertex characterizes the non-Gaussianity of the output distribution, we see explicitly here how interactions are mediating updates to the penultimate-layer activations. In addition, the leading factor of $1/n_{L-1}$ makes it clear that this update is a finite-width effect. Further, the term in the last parenthesis shows that the update depends explicitly on the difference

between our observations of the outputs, $y_{i;\tilde{\alpha}_3} y_{i;\tilde{\alpha}_4}$, and our prior expectations of them, $\mathbb{E}\left[z^{(L)}_{i;\tilde{\alpha}_3} z^{(L)}_{i;\tilde{\alpha}_4}\right] \equiv \widetilde{K}^{(L)}_{\tilde{\alpha}_3\tilde{\alpha}_4} + O(1/n)$. This means that the observations are in fact propagating *backward* to induce changes in the hidden-layer representations.[35]

Although perhaps not practically useful, this Bayesian analysis of representation learning at finite width will serve as a theoretically useful blueprint for studying a similar type of representation learning that occurs with gradient-based learning at finite width in §11. Now, with all these allusions to gradient-based learning having accrued with interest, you must be really excited to flip the page to the next chapter!

[35] This kind of backward-propagation or *backpropagation*, if you will, persists further into the shallower hidden layers as well. However, in the $(L-2)$-th layer, the posterior update turns out to be of order $O\big(1/n^2\big)$. Intuitively this makes sense because the change in the representation in the penultimate layer $(L-1)$ is already down by a factor of $1/n$, and it gets further suppressed due to the $1/n$-suppression of the interlayer interaction in going back to the $(L-2)$-th layer.

Mathematically, we can consider the update to an $(L-2)$-th-layer observable $\mathcal{O}\Big(z^{(L-2)}_{\mathcal{D}}\Big)$ as

$$\overline{d\mathcal{O}} \equiv \int dz^{(L-2)}_{\mathcal{D}} p\Big(z^{(L-2)}_{\mathcal{D}}\Big|y_{\mathcal{A}}\Big)\,\mathcal{O}\Big(z^{(L-2)}_{\mathcal{D}}\Big) - \int dz^{(L-2)}_{\mathcal{D}} p\Big(z^{(L-2)}_{\mathcal{D}}\Big)\,\mathcal{O}\Big(z^{(L-2)}_{\mathcal{D}}\Big)\,. \tag{6.105}$$

Through the chain of Bayes', sum, and product rules, the posterior insertion in this formula is given in terms of the following marginalization:

$$p\Big(z^{(L-2)}_{\mathcal{D}}\Big|y_{\mathcal{A}}\Big) = \frac{p\Big(y_{\mathcal{A}}\Big|z^{(L-2)}_{\mathcal{D}}\Big)\,p\Big(z^{(L-2)}_{\mathcal{D}}\Big)}{p(y_{\mathcal{A}})} = \int dz^{(L-1)}_{\mathcal{D}} \frac{p\Big(y_{\mathcal{A}}\Big|z^{(L-1)}_{\mathcal{D}}\Big)}{p(y_{\mathcal{A}})} p\Big(z^{(L-1)}_{\mathcal{D}}, z^{(L-2)}_{\mathcal{D}}\Big)\,. \tag{6.106}$$

From here, through the same set of manipulations that led to the update equation for the penultimate layer (6.99), we get

$$\overline{d\mathcal{O}} = \frac{1}{2}\sum_{\tilde{\alpha}_1,\ldots,\tilde{\alpha}_4\in\mathcal{A}} \mathbb{E}\left[\widehat{\Delta G}^{(L)}_{\tilde{\alpha}_1\tilde{\alpha}_2}\Big(z^{(L-1)}_{\mathcal{D}}\Big)\,\mathcal{O}\Big(z^{(L-2)}_{\mathcal{D}}\Big)\right] \widetilde{K}^{\tilde{\alpha}_1\tilde{\alpha}_3}_{(L)} \widetilde{K}^{\tilde{\alpha}_2\tilde{\alpha}_4}_{(L)} \sum_{i}^{n_L}\Big(y_{i;\tilde{\alpha}_3} y_{i;\tilde{\alpha}_4} - \widetilde{K}^{(L)}_{\tilde{\alpha}_3\tilde{\alpha}_4}\Big) + O\Big(\frac{1}{n^2}\Big)\,. \tag{6.107}$$

Thus, to show that this change is of order $O\big(1/n^2\big)$, we need to show that the *interlayer correlation*,

$$\mathbb{E}\left[\widehat{\Delta G}^{(L)}_{\tilde{\alpha}_1\tilde{\alpha}_2}\Big(z^{(L-1)}_{\mathcal{D}}\Big)\,\mathcal{O}\Big(z^{(L-2)}_{\mathcal{D}}\Big)\right], \tag{6.108}$$

is of order $O\big(1/n^2\big)$. This is most swiftly carried out in the future, first by the application of the formula (8.54) with $\ell = L-2$ and then with the associated trickery (8.70). If you are up for a challenge, please flip forward and write a note next to (8.70) reminding yourself to come back to footnote 35 in §6.4.3. *Spoiler alert:* you should in fact find that (6.108) is of order $O\big(1/n^2\big)$.

Chapter 7

Gradient-Based Learning

> *Of course, that's like saying Newton's second law $F = ma$, as it appears in textbooks on mechanics, is just a definition of what you mean by "force". That's true, strictly speaking, but we live in a landscape where there is an implicit promise that when someone writes that down ... that they will give laws for the force, and not, say, for some quantity involving the 17th time derivative of the position.*
>
> Sidney Coleman, in his "Quantum Mechanics in Your Face" Dirac Lecture [54].

In the last chapter, we discussed Bayesian inference as a learning algorithm, which followed naturally from our study of networks at initialization. Starting from a description of a neural network architecture with parameters – weights and biases – we integrated out these parameters to find a distribution over preactivations $z^{(\ell)}(x)$ as a function of layer and input sample, which in particular includes the output distribution $p\left(z^{(L)}(x)\right)$. This was interpreted as a *prior distribution* over an ensemble of such models, and then we explained how the logic of Bayes' rule lets us evolve the prior into a *posterior distribution* conditioned on observed data. Despite the theoretical elegance of Bayesian inference, the naive implementation quickly became computationally intractable as the number of conditioned data samples grew large.

Stepping back, there's actually something a little bit odd about this setup. Once we worked out the output distribution, the actual network itself was discarded, with the parameters long since integrated out. Since Bayesian inference only cares about the output distribution of a model, the starting point for inference can really be any ensemble of models as it isn't specifically tailored to neural networks at all. So why go through all the trouble of starting with neural-network models? How did we even know that these models are a good abstraction to begin with?

Deep neural networks are exciting because they work surprisingly well. We know this because in practice such networks are explicitly *trained* and used to perform useful tasks. Most commonly, learning occurs by repeatedly updating the model parameters via a gradient-based optimization procedure such as *gradient descent*.

In particular, gradient-based learning algorithms can efficiently process a large amount

of training data by optimizing an auxiliary *loss* function that directly compares the network output $f(x;\theta) \equiv z^{(L)}(x)$ to some desired result or *label*. This optimization procedure involves sampling only a single set of network parameters from the initialization distribution, yielding just a single network trained for the task of interest rather than a full ensemble of networks. In this way, gradient-based learning methods offset their inability to express confidence in their predictions – due to the absence of an ensemble – with data efficiency and easy scalability.

Since gradient descent involves making explicit updates to the model parameters, the first step is to bring them back (from whatever place that variables go when they are integrated out). In supervised learning, the adjustments of model parameters are directly proportional to the function-approximation error times the gradient of the model output with respect to the parameters. This decomposition motivates the study of the *neural tangent kernel* (NTK).[1] In short, the NTK is a type of Hamiltonian that controls the training dynamics of observables whenever gradient descent is used to optimize an auxiliary loss that scores a function approximation. As we detail in §10, §11, and §∞, understanding the NTK for a given neural-network architecture will enable us to effectively describe gradient-based learning for that model.

In this chapter, we give a short introduction to supervised learning in §7.1, followed by a discussion of gradient descent in §7.2 with a very general focus on how the NTK arises in supervised learning. In the next chapter, we'll incorporate the NTK into our effective theory of deep learning by exploiting the same layer-to-layer RG flow technique we used in §4.

7.1 Supervised Learning

One of the most basic modeling tasks at which neural networks excel is known as **supervised learning**. Given a **data distribution** $p(x, y) = p(y|x)p(x)$, the goal is to predict a **label** y given an input x, for any pair that is jointly sampled from the distribution.[2] To be precise, the model tries to learn the conditional distribution $p(y|x)$, and the resulting model is sometimes called a **discriminative model**. In one canonical example from computer vision, we might want to **classify** an image x_δ of a hand-written digit "3" according to its literal value $y_\delta = \texttt{3}$. Or, for a natural language processing example, given a sentence containing the word $x_\delta = \texttt{cat}$ we might want to identify the part of the speech as $y_\delta = \texttt{noun}$. The better the *probabilistic model* learns the distribution $p(y|x)$, the more accurately it can predict a true label y for a novel input example x. Generating these datasets generally requires human annotators to label the inputs, hence the name supervised learning.

In this setup, the supervised-learning model outputs a prediction $z(x_\delta;\theta)$. This notation emphasizes that the model output is both a function of the input x_δ as well as

[1] The NTK was first identified in the seminal work of Jacot *et al.* [55] in the context of infinite-width networks.

[2] In this section, we suppress *vectorial indices* on the inputs x_δ, labels y_δ, and model outputs $z\left(x_\delta;\theta\right)$, while often retaining *sample indices* $\delta \in \mathcal{D}$.

some adjustable parameters θ. This should already be familiar in the context of neural-network function approximation, where the model parameters consist of the biases and weights.

As discussed in §2.3, the model parameters are drawn from an easy-to-sample prior distribution over the parameters, which is also known as the *initialization distribution* in the context of gradient-based learning. Importantly, this parameter distribution knows nothing about the data distribution. Thus, in order for the model to make good predictions, its parameters will need to be adjusted somehow. Really, this is just a specific application of the function approximation that we discussed in §2.1 where the function to be approximated is a conditional distribution $p(y|x)$.

Before we understand how to adjust or *fit* the model parameters, we need to understand what we mean by making good predictions. What we want is, for a typical input x_δ and a label y_δ sampled from the data distribution $p(x, y)$, that the model output $z(x_\delta; \theta)$ is as close to the label y_δ as possible on average. In order to measure this proximity, for a prediction-label pair we need to define an auxiliary *objective function* or **loss**,

$$\mathcal{L}\Big(z(x_\delta; \theta),\, y_\delta\Big)\,, \tag{7.1}$$

with the property that the closer $z(x_\delta; \theta)$ is to y_δ, the lower the value of the function is. One very intuitive choice for the loss is *MSE loss* (6.17),

$$\mathcal{L}_{\text{MSE}}\Big(z(x_\delta; \theta),\, y_\delta\Big) \equiv \frac{1}{2}\Big[z(x_\delta; \theta) - y_\delta\Big]^2\,, \tag{7.2}$$

which clearly has the required property, though this is not the most common choice in deep learning. The specific form of the loss will not matter for the rest of the chapter.

With the loss function in hand, the goal of training is to adjust model parameters so as to minimize the loss for as many input-label pairs as possible. Ideally, we would like to minimize the loss averaged over the entire data distribution,

$$\mathbb{E}\left[\mathcal{L}(\theta)\right] = \int dxdy\, p(x, y)\, \mathcal{L}\Big(z(x; \theta),\, y\Big)\,. \tag{7.3}$$

But since we almost never have access to the analytical form of the data distribution $p(x, y)$, in practice this would require the sampling of an infinite number of input-label pairs. Instead, as a proxy of the entire loss (7.3), we sample a large-but-finite number of pairs $(x_{\tilde{\alpha}}, y_{\tilde{\alpha}})_{\tilde{\alpha}\in\mathcal{A}}$ and try to minimize

$$\mathcal{L}_{\mathcal{A}}(\theta) \equiv \sum_{\tilde{\alpha}\in\mathcal{A}} \mathcal{L}\Big(z(x_{\tilde{\alpha}}; \theta),\, y_{\tilde{\alpha}}\Big)\,. \tag{7.4}$$

This set of examples $\mathcal{A}$ is referred to as the **training set**, and the estimate of the loss (7.4) is called the **training loss**; here we've also inherited from §6 our sample-index notation of alpha-with-tilde for the inputs in the training set $\tilde{\alpha} \in \mathcal{A}$, while denoting generic inputs as delta-with-no-decoration $\delta \in \mathcal{D}$, and soon we'll use beta-with-dot for

inputs in the *test set* $\dot{\beta} \in \mathcal{B}$.[3] To train our model, we try to find a configuration of the model parameters that minimizes the training loss

$$\theta^\star = \arg\min_\theta \mathcal{L}_\mathcal{A}(\theta) = \arg\min_\theta \left[\sum_{\tilde{\alpha} \in \mathcal{A}} \mathcal{L}\Big(z(x_{\tilde{\alpha}}; \theta),\, y_{\tilde{\alpha}} \Big) \right] . \tag{7.6}$$

In the next section, we will present the gradient descent algorithm as a way to accomplish this goal.

Having set the minimization of the training loss (7.4) as our optimization problem, it is important to keep in mind that the true goal of supervised learning is the minimization of the loss over the entire data distribution in the sense of (7.3). Said another way, the question is not whether the model is able to memorize all the input-label pairs in the training set, but rather whether it's able to generalize its predictions to additional input-label pairs not seen during training. One might then worry about whether a training set is *biased* in its sampling of the data distribution or whether there is high *variance* in a particular set of samples.

To explicitly assess this **generalization** property of a model, a separate set of input-label samples $(x_{\dot{\beta}}, y_{\dot{\beta}})_{\dot{\beta} \in \mathcal{B}}$ – known as the **test set** – is typically set aside and only used to evaluate a model after training is complete. To the extent that the training set $\mathcal{A}$ is representative of the full data distribution $p(x, y)$, decreasing the training loss will often decrease the entire loss (7.3), as estimated by the test loss $\mathcal{L}_\mathcal{B}$. We will address this question directly in §10.

7.2 Gradient Descent and Function Approximation

Considering the training loss minimization (7.6), we see that learning is a complicated optimization problem. Being entirely naive about it, in order to find extrema of a function, calculus instructs us to differentiate the training loss and find the value of the

[3] Note that our definition of the training loss (7.4) is a bit at odds with our definition of the expected loss (7.3). In particular, the expected loss is *intensive*, while the training loss is *extensive*, scaling linearly with the size of the training set $N_\mathcal{A} \equiv |\mathcal{A}|$. This latter choice is consistent with our first definition of this loss, (6.17), in the context of MLE as an approximate method for Bayesian model fitting in §6.2.1. There, the extensivity of the loss was natural according to the Bayesian framework: as the number of observed input-output pairs $N_\mathcal{A}$ increases, we want the likelihood to dominate the prior. As such, we will find it natural to follow that convention. (You also might more accurately call the extensive loss (6.17) as the ~~*mean*~~ *squared error*.) However, from a non-Bayesian perspective, it is often customary to define a training loss as

$$\mathcal{L}_\mathcal{A}(\theta) \equiv \frac{1}{|\mathcal{A}|} \sum_{\tilde{\alpha} \in \mathcal{A}} \mathcal{L}\Big(z(x_{\tilde{\alpha}}; \theta),\, y_{\tilde{\alpha}} \Big) , \tag{7.5}$$

which better corresponds to the expected loss (7.3). Since in the context of gradient-based learning the overall normalization can always be absorbed in a redefinition of the global learning rate η, to be introduced next section, the only advantage we see of this latter definition (7.5) is the better correspondence of the loss with its name.

argument for which the resulting expression vanishes:

$$0 = \left.\frac{d\mathcal{L}_{\mathcal{A}}}{d\theta_\mu}\right|_{\theta=\theta^\star} . \tag{7.7}$$

Unfortunately this equation is exactly solvable only in special cases, for instance when the loss is quadratic in the model parameters. Rather than trying to find minima analytically, practitioners typically employ an iterative procedure to bring the loss closer and closer to a minimum.

Gradient descent is one such method that can be used to minimize nontrivial functions like the training loss (7.4), and so it's a natural candidate for model fitting. The algorithm involves the computation of the gradient of the loss and iteratively updates the model parameters in the (negative) direction of the gradient

$$\theta_\mu(t+1) = \theta_\mu(t) - \eta \left.\frac{d\mathcal{L}_{\mathcal{A}}}{d\theta_\mu}\right|_{\theta_\mu=\theta_\mu(t)} , \tag{7.8}$$

where t keeps track of the number of steps in the iterative training process, with $t = 0$ conventionally being the point of initialization. Here, η is a positive **training hyperparameter** called the **learning rate**, which controls how large of a step is taken in *parameter space*. Note that the computational cost of gradient descent scales linearly with the size of the dataset $\mathcal{A}$, as one just needs to compute the gradient for each sample and then add them up.

For sufficiently small learning rates, the updates (7.8) are guaranteed to decrease the training loss $\mathcal{L}_{\mathcal{A}}$. In order to see this, let us Taylor-expand the training loss around the current value of the parameters $\theta(t)$ and compute the change in the loss after making an update

$$\Delta\mathcal{L}_{\mathcal{A}} \equiv \mathcal{L}_{\mathcal{A}}\Big(\theta(t+1)\Big) - \mathcal{L}_{\mathcal{A}}\Big(\theta(t)\Big) = -\eta \sum_\mu \left(\frac{d\mathcal{L}_{\mathcal{A}}}{d\theta_\mu}\right)^2 \Bigg|_{\theta=\theta(t)} + O(\eta^2) . \tag{7.9}$$

As minus a sum of squares, this is strictly negative. Pretty typically, iterating these updates will eventually lead to (at least) a local minimum of the training loss. In practice, small variants of the gradient descent algorithm are responsible for almost all training and optimization in deep learning.[4]

[4]In particular, the most popular learning algorithm is **stochastic gradient descent** (SGD). SGD uses updates of the form

$$\theta_\mu(t+1) = \theta_\mu(t) - \eta \left.\frac{d\mathcal{L}_{\mathcal{S}_t}}{d\theta_\mu}\right|_{\theta_\mu=\theta_\mu(t)} , \tag{7.10}$$

where $\mathcal{S}_t$ is a subset of the training set, $\mathcal{S}_t \subset \mathcal{A}$. Each subset $\mathcal{S}_t$ is called a *mini-batch* or **batch**. Training is then organized by **epoch**, which is a complete passes through the training set. Typically, for each epoch the training set is *stochastically* partitioned into subsets of equal size, which are then sequentially used to estimate the gradient.

Tensorial Gradient Descent

In one such variant, we can define a more general family of learning algorithms by modifying the update (7.8) as

$$\theta_\mu(t+1) = \theta_\mu(t) - \eta \sum_\nu \lambda_{\mu\nu} \frac{d\mathcal{L}_\mathcal{A}}{d\theta_\nu}\bigg|_{\theta=\theta(t)} , \tag{7.11}$$

where the tensor $\lambda_{\mu\nu}$ is a **learning-rate tensor** on parameter space; the original gradient-descent update (7.8) is a special case with the Kronecker delta as the tensor $\lambda_{\mu\nu} = \delta_{\mu\nu}$. While in the original gradient descent (7.8) we have one global learning rate η, in the tensorial gradient descent (7.11) we have the freedom to separately specify how the ν-th component of the gradient $d\mathcal{L}_\mathcal{A}/d\theta_\nu$ contributes to the update of the μ-th parameter θ_μ via the tensor $\lambda_{\mu\nu}$. Repeating the same Taylor-expansion in η (7.9) with the generalized update (7.11), we find

$$\Delta\mathcal{L}_\mathcal{A} = -\eta \sum_{\mu,\nu} \lambda_{\mu\nu} \frac{d\mathcal{L}_\mathcal{A}}{d\theta_\mu} \frac{d\mathcal{L}_\mathcal{A}}{d\theta_\nu} + O(\eta^2) , \tag{7.12}$$

indicating that the training loss again is almost surely decreasing for sufficiently small learning rates, so long as the learning-rate tensor $\lambda_{\mu\nu}$ is a positive semidefinite matrix.

Neural Tangent Kernel

Everything we have said so far about gradient descent could be applied equally to the optimization of any function. However, in the context of function approximation there is additional structure: the optimization objective is a function of the model output.

To take advantage of this structure, first note that by the chain rule the gradient of the loss can be expressed as

$$\frac{d\mathcal{L}_\mathcal{A}}{d\theta_\mu} = \sum_{i=1}^{n_\text{out}} \sum_{\tilde{\alpha}\in\mathcal{A}} \frac{\partial\mathcal{L}_\mathcal{A}}{\partial z_{i;\tilde{\alpha}}} \frac{dz_{i;\tilde{\alpha}}}{d\theta_\mu} , \tag{7.13}$$

which means that the change in the loss (7.12) after an update can be nicely decomposed as

$$\Delta\mathcal{L}_\mathcal{A} = -\eta \sum_{i_1,i_2=1}^{n_\text{out}} \sum_{\tilde{\alpha}_1,\tilde{\alpha}_2\in\mathcal{A}} \left[\frac{\partial\mathcal{L}_\mathcal{A}}{\partial z_{i_1;\tilde{\alpha}_1}} \frac{\partial\mathcal{L}_\mathcal{A}}{\partial z_{i_2;\tilde{\alpha}_2}}\right] \left[\sum_{\mu,\nu} \lambda_{\mu\nu} \frac{dz_{i_1;\tilde{\alpha}_1}}{d\theta_\mu} \frac{dz_{i_2;\tilde{\alpha}_2}}{d\theta_\nu}\right] + O(\eta^2) . \tag{7.14}$$

The quantity in the first square bracket is a measure of the function approximation error. For instance, for the MSE loss (7.2) we see that the gradient of the loss with respect to the model output is exactly the prediction error,

$$\frac{\partial\mathcal{L}_\mathcal{A}}{\partial z_{i;\tilde{\alpha}}} = z_i(x_{\tilde{\alpha}};\theta) - y_{i;\tilde{\alpha}} . \tag{7.15}$$

The advantage of this algorithm is twofold: *(i)* the computational cost of training now scales with the fixed size of the sets $\mathcal{S}_t$ rather than with the size of the whole training set $\mathcal{A}$; and *(ii)* SGD is thought to have better generalization properties than gradient descent. Nevertheless, essentially everything we will say about gradient descent will apply to stochastic gradient descent as well.

More generally for other losses, the gradient of the loss or **error factor**

$$\epsilon_{i;\tilde{\alpha}} \equiv \frac{\partial \mathcal{L}_{\mathcal{A}}}{\partial z_{i;\tilde{\alpha}}}\,, \tag{7.16}$$

is small when the model output is close to the label. Sensibly, the greater the error factor, the larger the update (7.13), and the greater the change in the loss (7.14). The quantity in the second square bracket is called the **neural tangent kernel** (NTK)

$$H_{i_1 i_2;\tilde{\alpha}_1\tilde{\alpha}_2} \equiv \sum_{\mu,\nu} \lambda_{\mu\nu} \frac{dz_{i_1;\tilde{\alpha}_1}}{d\theta_\mu} \frac{dz_{i_2;\tilde{\alpha}_2}}{d\theta_\nu}\,. \tag{7.17}$$

As is clear from (7.17), the NTK is independent of the auxiliary loss function.

Importantly, the NTK is the main driver of the function-approximation dynamics. To the point, it governs the evolution of a much more general set of observables than the training loss. Consider any observable that depends on the model's outputs

$$\mathcal{O}(\theta) \equiv \mathcal{O}\Big(z(x_{\delta_1};\theta),\, \ldots\,, z(x_{\delta_M};\theta)\Big)\,, \tag{7.18}$$

where $x_{\delta_1}, \ldots, x_{\delta_M} \in \mathcal{D}$ for some dataset $\mathcal{D}$. For example, if $\mathcal{D}$ is the test set $\mathcal{B}$ and $\mathcal{O}$ is the loss function, then this observable would be the test loss $\mathcal{L}_{\mathcal{B}}$. In addition to the test loss, one might want to observe the change in a particular component of the output $\mathcal{O} = z_i(x)$ or perhaps track correlations among different vectorial components of the output $\mathcal{O} = z_i(x)\, z_j(x)$ for a given input x. For any such observable (7.18), its change after an update is given by the expression

$$\mathcal{O}\Big(\theta(t+1)\Big) - \mathcal{O}\Big(\theta(t)\Big) = -\eta \sum_{i_1,i_2=1}^{n_{\text{out}}} \sum_{\tilde{\alpha}\in\mathcal{A}} \sum_{\delta\in\mathcal{D}} \left[\frac{\partial \mathcal{L}_{\mathcal{A}}}{\partial z_{i_1;\tilde{\alpha}}} \frac{\partial \mathcal{O}}{\partial z_{i_2;\delta}}\right] H_{i_1 i_2;\tilde{\alpha}\delta} + O(\eta^2)\,. \tag{7.19}$$

As we see, the square bracket contains the function-approximation error as well as the particulars about how the observable depends on the model output. In contrast, the NTK contains all the dynamical information pertaining to the particular model, depending only on the model architecture and parameters.[5]

We can further understand the function-approximation dynamics under gradient descent by considering a particular vectorial component of the output for a particular sample as an observable, i.e. $\mathcal{O} = z_i(x_\delta)$. In this case, the derivative of $\mathcal{O}$ in (7.19) is a Kronecker delta on both the vectorial indices and sample indices, and the evolution reduces to

$$z_i\Big(x_\delta;\theta(t+1)\Big) - z_i\Big(x_\delta;\theta(t)\Big) = -\eta \sum_{j=1}^{n_{\text{out}}} \sum_{\tilde{\alpha}\in\mathcal{A}} H_{ij;\delta\tilde{\alpha}} \epsilon_{j;\tilde{\alpha}} + O(\eta^2)\,. \tag{7.20}$$

[5]As our discussion makes clear, the NTK can generally be defined for any function approximator. This means that its name masks its true generality. In addition to objecting to the "neural" part of the name, one could object to the "kernel" part. In particular, the NTK is more akin to a Hamiltonian than a kernel as it generates the evolution of observables; we'll fully justify this claim in §∞.2.2.

This equation shows how the model output changes after a training update. Importantly, we see how the error factor $\epsilon_{j;\tilde{\alpha}}$ (7.16) from example $x_{\tilde{\alpha}}$ on the model output component j affects the updated behavior of the model output component i on a different example x_δ: it's mediated by the NTK component $H_{ij;\delta\tilde{\alpha}}$. This is what makes function approximation possible; the ability to learn something about one example, x_δ, by observing another, $x_{\tilde{\alpha}}$. We see that the off-diagonal components of the NTK in the sample indices determine the generalization behavior of the model, while the off-diagonal components in vectorial indices allow for one feature to affect the training of another feature. We will have more to say about the former property in §10 and the latter property in §∞.

Finally, let us note in passing that unlike the case of the training loss (7.14), for general observables (7.19) the term in the square bracket is not necessarily positive. While the training loss $\mathcal{L}_{\mathcal{A}}$ will always decrease for small enough learning rates, a given observable may not. In particular, nothing guarantees that the test loss will decrease and – for models that overfit their training set – the test loss may even increase.

Chapter 8

RG Flow of the Neural Tangent Kernel

People get things backwards and they shouldn't—it has been said, and wisely said, that every successful physical theory swallows its predecessors alive.

Sidney Coleman, more forward and a little bit deeper in that same "Quantum Mechanics in Your Face" Dirac Lecture [54].

In the last chapter, we introduced gradient-based learning as an alternative to Bayesian learning and specifically focused on the gradient descent algorithm. In short, the gradient descent algorithm involved instantiating a network from the prior distribution and then repeatedly updating the model parameters by running training data through the network. This algorithm is straightforward to implement and very efficient to run for any particular network. In practice, it makes things very easy.

In theory, it makes things a little more difficult. For the Bayesian prior, we were able to integrate out the model parameters layer by layer in deriving the output distribution because the initialization distributions of the biases and weights were extremely simple; in addition, the large-width expansion made it possible to derive analytic expressions for the Bayesian posterior for finite-width networks. By contrast, the model parameters and the outputs of any particular network trained by gradient descent are a complicated correlated mess.

To make progress, we first need to shift the perspective back to a statistical one. Rather than focusing on how any particular network learns from the data, we instead ask how a *typical* network behaves when being trained. If we understand the typical behavior (i.e. the mean) under gradient descent and have control of the fluctuations from network instantiation to instantiation (i.e. the variance), then we can describe gradient-based learning as used in practice.

With that statistical perspective in mind, recall from the last chapter that the gradient-descent updates decompose into an error factor times a function-approximation factor. The latter factor was dubbed the *neural tangent kernel* (NTK) and conveniently

summarizes the effect of the model parameters' changes on the behavior of the network. This means that the statistics of changes in network observables in the initial stage of training are governed by the statistics of the NTKs at initialization. To proceed forward, the core of the current chapter and the next will involve explicitly computing such NTK statistics for deep MLPs; we will postpone the actual analysis of neural network training – enabled by these computations of the NTK statistics – until §10 and §∞.

In §8.0, we will lay the groundwork for the recursive computation of the NTK statistics. Namely, starting from the MLP iteration equation, or the *forward* equation for the preactivations, we'll derive a corresponding forward equation for the NTK. This equation is a layer-to-layer iteration equation that holds for each distinct instantiation of the model parameters. (Here we'll also remark on how the learning-rate tensor should be scaled with network width, an important point that is often neglected in practice.)

By averaging over different instantiations, we can then use the forward equation to recursively compute the joint statistics of the NTK and the preactivations. The approach taken here completely mirrors the *RG-flow* approach taken in §4 for the preactivations. In §8.1, §8.2, and §8.3, we will progressively determine the sequence of joint NTK-preactivation distributions in the first, second, and deeper layers, respectively.

8.0 Forward Equation for the NTK

As we saw in the previous chapter, the evolution of observables $\mathcal{O}(z)$ under gradient descent is governed by the NTK,

$$H_{i_1 i_2;\alpha_1\alpha_2} \equiv \sum_{\mu,\nu} \lambda_{\mu\nu} \frac{dz_{i_1;\alpha_1}}{d\theta_\mu} \frac{dz_{i_2;\alpha_2}}{d\theta_\nu} \,, \tag{8.1}$$

where $\lambda_{\mu\nu}$ is the learning-rate tensor.

Specializing to MLPs, observables can depend not only on the network's output $z_{i;\alpha} = z^{(L)}_{i;\alpha}$, but also on the preactivations $z_{i;\alpha} = z^{(\ell)}_{i;\alpha}$ in any layer. Such ℓ-th-layer observables for $\ell < L$ tell us about the *hidden-layer representations* of the network. For instance, the neural component $\mathcal{O} = z^{(\ell)}_i(x)$ tells us about an ℓ-th-layer feature evaluated on an input x, while $\mathcal{O} = z^{(\ell)}_i(x)\, z^{(\ell)}_j(x)$ with neural indices $i \neq j$ tracks correlations among different features given x.

With similar manipulations as before, we find that an observable $\mathcal{O}$ that depends only on the ℓ-th-layer preactivations

$$\mathcal{O}(\theta) \equiv \mathcal{O}\Big(z^{(\ell)}(x_{\delta_1};\theta), \ldots, z^{(\ell)}(x_{\delta_M};\theta)\Big) \,, \tag{8.2}$$

evolves after a gradient descent update as

$$\mathcal{O}\Big(\theta(t+1)\Big) - \mathcal{O}\Big(\theta(t)\Big) = -\eta \sum_{i_1,i_2=1}^{n_\ell} \sum_{\alpha\in\mathcal{A}} \sum_{\delta\in\mathcal{D}} \left[\frac{d\mathcal{L}_\mathcal{A}}{dz^{(\ell)}_{i_1;\alpha}} \frac{\partial\mathcal{O}}{\partial z^{(\ell)}_{i_2;\delta}}\right] H^{(\ell)}_{i_1 i_2;\alpha\delta} + O(\eta^2) \,, \tag{8.3}$$

where $x_{\delta_1}, \ldots, x_{\delta_M} \in \mathcal{D}$ for some dataset $\mathcal{D}$.[1] Here, we have defined the **ℓ-th-layer NTK** as

$$H^{(\ell)}_{i_1 i_2; \alpha_1 \alpha_2} \equiv \sum_{\mu,\nu} \lambda_{\mu\nu} \frac{dz^{(\ell)}_{i_1;\alpha_1}}{d\theta_\mu} \frac{dz^{(\ell)}_{i_2;\alpha_2}}{d\theta_\nu} \,, \tag{8.5}$$

which governs the evolution of the ℓ-th-layer observables; in terms of this notation, the output NTK is simply $H^{(\ell=L)}_{i_1 i_2; \alpha_1 \alpha_2}$. Note that whenever we write the ℓ-th-layer NTK as above, we will always assume that the learning-rate tensor $\lambda_{\mu\nu}$ does not mix network parameters from different layers, though in general it can still mix the biases and weights within a layer. We will place further restrictions on this in another paragraph.

At initialization, the model parameters are sampled from their initialization distributions, and the ℓ-th-layer NTK is a stochastic object. In order to emphasize this stochasticity, in what follows we'll decorate the NTK *at initialization* with a hat: $\widehat{H}^{(\ell)}_{i_1 i_2; \alpha_1 \alpha_2}$. Our goal is to evaluate its statistics.

Before we go any further, it is convenient to make a specialized choice for the learning-rate tensor $\lambda_{\mu\nu}$. In practice, typically $\lambda_{\mu\nu} = \delta_{\mu\nu}$, and there is only the *global* learning rate η for the entire model. Even in a more general setup, a learning rate is often shared among each group of parameters that are sampled from the same distribution. Recalling that the same distribution was shared among the biases in a given layer with the same variance $C_b^{(\ell)}$ (2.19) and similarly for the weights with the *rescaled* weight variance $C_W^{(\ell)}$ (2.20), this suggests an ansatz for our **training hyperparameters**: we should decompose the learning-rate tensor $\lambda_{\mu\nu}$ into a diagonal matrix

$$\lambda_{b^{(\ell)}_{i_1} b^{(\ell)}_{i_2}} = \delta_{i_1 i_2} \lambda_b^{(\ell)} \,, \quad \lambda_{W^{(\ell)}_{i_1 j_1} W^{(\ell)}_{i_2 j_2}} = \delta_{i_1 i_2} \delta_{j_1 j_2} \frac{\lambda_W^{(\ell)}}{n_{\ell-1}} \,, \tag{8.6}$$

giving each group of biases in a layer the same learning rate and each group of weights in a layer the same learning rate, and allowing such learning rates to vary from layer to layer.

Importantly, we have normalized the learning rate for a given weight $W^{(\ell)}_{i_1 j_1}$ by the width of the previous layer $n_{\ell-1}$, just as we did for the variance of the weight's initialization distribution. This normalization is there for much the same reason: the freedom to tune the weight learning rates separately from the bias learning rates will prove necessary for having a sensible large-width expansion. Going forward, our training hyperparame-

[1] However, note that this is not quite as simple as the expression for the evolution of the network output that we gave in the last chapter (7.19). In particular, the derivative of the loss with respect to the ℓ-th-layer preactivations needs to be computed by the chain rule as

$$\frac{d\mathcal{L}_\mathcal{A}}{dz^{(\ell)}_{i_1;\alpha}} = \sum_{j=1}^{n_L} \frac{\partial \mathcal{L}_\mathcal{A}}{\partial z^{(L)}_{j;\alpha}} \frac{dz^{(L)}_{j;\alpha}}{dz^{(\ell)}_{i_1;\alpha}} \,, \tag{8.4}$$

with the error factor $\partial \mathcal{L}_\mathcal{A} / \partial z^{(L)}_{j;\alpha}$ now multiplied by the chain-rule factor $dz^{(L)}_{j;\alpha} / dz^{(\ell)}_{i_1;\alpha}$. For observables that depend on preactivations from multiple layers, the generalization of (8.3) further involves additional chain-rule factors as well as a sum over NTKs from different layers.

ters will consist of the global learning rate η and the individual ℓ-th-layer learning rates for the biases and weights, $\lambda_b^{(\ell)}$ and $\lambda_W^{(\ell)}$.

Substituting our choice for $\lambda_{\mu\nu}$ back into the definition of the ℓ-th-layer NTK (8.5), this expression decomposes as

$$\widehat{H}^{(\ell)}_{i_1 i_2;\alpha_1\alpha_2} = \sum_{\ell'=1}^{\ell} \left[\sum_{j=1}^{n_{\ell'}} \left(\lambda_b^{(\ell')} \frac{dz^{(\ell)}_{i_1;\alpha_1}}{db_j^{(\ell')}} \frac{dz^{(\ell)}_{i_2;\alpha_2}}{db_j^{(\ell')}} + \frac{\lambda_W^{(\ell')}}{n_{\ell'-1}} \sum_{k=1}^{n_{\ell'-1}} \frac{dz^{(\ell)}_{i_1;\alpha_1}}{dW_{jk}^{(\ell')}} \frac{dz^{(\ell)}_{i_2;\alpha_2}}{dW_{jk}^{(\ell')}} \right) \right] . \tag{8.7}$$

Here, the part in the square brackets is the per-layer contribution of the model parameters to the ℓ-th-layer NTK, treating the biases and weights separately. We also see that our intuition above in (8.6) was correct: the ℓ'-th-layer weight learning rate $\lambda_W^{(\ell')}$ needs to be accompanied by a factor of $1/n_{\ell'-1}$ in order to compensate for the additional summation over the $(\ell'-1)$-th layer neural indices in the second term as compared to the first. Even so, the layer sum in (8.7) makes this expression somewhat unwieldy and suggests that we should search for an alternate representation.

Following our analysis of the preactivations, let's try to find a recursive expression. To that end, consider the $(\ell+1)$-th-layer NTK, $\widehat{H}^{(\ell+1)}_{i_1 i_2;\alpha_1\alpha_2}$, and decompose the sum over layers in its definition by separating the $(\ell+1)$-th-layer term from all of the rest, giving

$$\begin{aligned} \widehat{H}^{(\ell+1)}_{i_1 i_2;\alpha_1\alpha_2} = &\sum_{j=1}^{n_{\ell+1}} \left(\lambda_b^{(\ell+1)} \frac{dz^{(\ell+1)}_{i_1;\alpha_1}}{db_j^{(\ell+1)}} \frac{dz^{(\ell+1)}_{i_2;\alpha_2}}{db_j^{(\ell+1)}} + \frac{\lambda_W^{(\ell+1)}}{n_\ell} \sum_{k=1}^{n_\ell} \frac{dz^{(\ell+1)}_{i_1;\alpha_1}}{dW_{jk}^{(\ell+1)}} \frac{dz^{(\ell+1)}_{i_2;\alpha_2}}{dW_{jk}^{(\ell+1)}} \right) \\ &+ \sum_{j_1,j_2=1}^{n_\ell} \frac{dz^{(\ell+1)}_{i_1;\alpha_1}}{dz^{(\ell)}_{j_1;\alpha_1}} \frac{dz^{(\ell+1)}_{i_2;\alpha_2}}{dz^{(\ell)}_{j_2;\alpha_2}} \widehat{H}^{(\ell)}_{j_1 j_2;\alpha_1\alpha_2} . \end{aligned} \tag{8.8}$$

Here, the first line is the $(\ell+1)$-th-layer term that we left alone, while the second line gives the terms from all the other layers after applying the chain rule and then recalling the definition (8.7). In this way, the ℓ-th-layer NTK appears naturally. This means that we can find a simple iterative expression for the NTK, similar in spirit to the forward equation for the preactivations that defines the MLP.

To finish our derivation, we need to evaluate the derivatives in (8.8). To do so, recall the preactivation forward iteration equation

$$z^{(\ell+1)}_{i;\alpha} = b_i^{(\ell+1)} + \sum_{j=1}^{n_\ell} W_{ij}^{(\ell+1)} \sigma^{(\ell)}_{j;\alpha} , \tag{8.9}$$

and remember that the activations are explicit functions of the preactivations $\sigma^{(\ell)}_{i;\alpha} \equiv \sigma\left(z^{(\ell)}_{i;\alpha}\right)$. The factors in the second line of (8.8) coming from the chain rule evaluate to

$$\frac{dz^{(\ell+1)}_{i;\alpha}}{dz^{(\ell)}_{j;\alpha}} = W_{ij}^{(\ell+1)} \sigma'^{(\ell)}_{j;\alpha} , \tag{8.10}$$

while the derivatives with respect to the $(\ell+1)$-th-layer parameters evaluate to

$$\frac{dz^{(\ell+1)}_{i;\alpha}}{db^{(\ell+1)}_{j}} = \delta_{ij}\,, \qquad \frac{dz^{(\ell+1)}_{i;\alpha}}{dW^{(\ell+1)}_{jk}} = \delta_{ij}\,\sigma^{(\ell)}_{k;\alpha}\,. \tag{8.11}$$

All together, we can rewrite (8.8) as

$$\begin{aligned}\widehat{H}^{(\ell+1)}_{i_1 i_2;\alpha_1\alpha_2} =& \delta_{i_1 i_2}\left[\lambda_b^{(\ell+1)} + \lambda_W^{(\ell+1)}\left(\frac{1}{n_\ell}\sum_{j=1}^{n_\ell}\sigma^{(\ell)}_{j;\alpha_1}\sigma^{(\ell)}_{j;\alpha_2}\right)\right] \\ &+ \sum_{j_1,j_2=1}^{n_\ell} W^{(\ell+1)}_{i_1 j_1} W^{(\ell+1)}_{i_2 j_2}\sigma'^{(\ell)}_{j_1;\alpha_1}\sigma'^{(\ell)}_{j_2;\alpha_2}\widehat{H}^{(\ell)}_{j_1 j_2;\alpha_1\alpha_2}\,.\end{aligned} \tag{8.12}$$

This is the **forward equation for the NTK**, which is an iteration equation that computes the NTK layer by layer for any realization of the biases and weights. This is analogous to the way in which (8.9) computes the network output – as well as all the hidden-layer preactivations – via a layer-to-layer iteration for a given realization of model parameters.

Scaling in the effective theory

The forward equation (8.12) further clarifies our decomposition (8.6) in which we made a distinction between the learning rates for the biases and those for the weights, giving each a different scaling with respect to the layer widths n_ℓ of the network.[2]

To see why, first recall from §7 that the change in the training loss after a step of gradient descent is proportional to the product of the global learning rate η and the final-layer NTK $\widehat{H}^{(L)}_{i_1 i_2;\alpha_1\alpha_2}$:

$$\Delta\mathcal{L}_{\mathcal{A}} = -\eta\sum_{i_1,i_2=1}^{n_L}\sum_{\alpha_1,\alpha_2\in\mathcal{A}}\epsilon_{i_1;\alpha_1}\epsilon_{i_2;\alpha_2}\,\widehat{H}^{(L)}_{i_1 i_2;\alpha_1\alpha_2} + O(\eta^2)\,, \tag{8.13}$$

where here we also recall the definition of the error factor

$$\epsilon_{i;\alpha} \equiv \frac{\partial\mathcal{L}_{\mathcal{A}}}{\partial z^{(L)}_{i;\alpha}}\,. \tag{8.14}$$

Note that this error factor generally stays of order one in the large-width limit, cf. the explicit expression when using the MSE loss (7.15). Thus, it's essential that the product of the global learning rate and the NTK, $\eta\widehat{H}^{(L)}$, also stays of order one for large-width networks: if it diverged as the width increases, then the higher-order terms in (8.13) would dominate and the loss would no longer be guaranteed to decrease; if instead it

[2] You'll have to wait until §9 to understand why it is advantageous to give a layer dependence to $\lambda_b^{(\ell)}$ and $\lambda_W^{(\ell)}$ and to learn how they should be scaled with depth.

vanished in this limit, then no training would take place. Either way, training would fail.

With that in mind, we chose the width scaling of our learning-rate tensor so that the NTK naturally stays of order one in the large-width limit and hence a (sufficiently small but not parametrically small) order-one global learning η ensures the success of training. In particular, the $(\ell+1)$-th-layer contribution in the first line of the forward equation (8.12) stays of order one if we take $\lambda_b^{(\ell+1)}, \lambda_W^{(\ell+1)} = O(1)$, with the $1/n_\ell$ normalization of the $(\ell+1)$-th-layer weight learning rate playing an essential role in compensating for the summation over the n_ℓ terms.[3]

If instead we had considered the original version of gradient descent with $\lambda_{\mu\nu} = \delta_{\mu\nu}$ rather than *tensorial gradient descent*, we would have been in trouble. In the language of our effective theory, the original gradient descent corresponds to setting $\lambda_b^{(\ell)} = 1$ and $\lambda_W^{(\ell)} = n_{\ell-1}$, which means that the NTK itself would be $O(n)$. We'd then have to scale the global learning rate as $\eta = O(1/n)$ to compensate for this $O(n)$ scaling of the NTK. However, since in this case $\eta\lambda_b^{(\ell)} = O(1/n)$, the order-one contribution from the weights to the NTK would completely overwhelm the $1/n$-suppressed contribution from the biases. This would lead to both a lack of appropriate contribution of the biases to the updates of the weights as well as an extreme under-training of the biases themselves.

Finally, let's make a general point: in any effective theory, it's really essential to make all large or small scales explicit – and rescale hyperparameters accordingly – as we did earlier for the variance of the weight initialization distribution, did here for the weight learning rate, and will do later for the depth scaling of both the bias and weight learning rates. For the effective theorist this ensures that the asymptotic $1/n$ and $1/\ell$ expansions are sound and nontrivial, and for the practical practitioner this enables comparisons of hyperparameter values across architectures with different widths and depths. In particular, we expect very generally that this should help mitigate expensive hyperparameter tuning, remove the need for heuristic fixes, and increase the robustness of optimal hyperparameter settings when scaling a model up.

[3] With this choice, the recursive term in the second line of the forward equation (8.12) also stays of order one. To see this, let's evaluate its expectation:

$$\begin{aligned}\mathbb{E}\left[\sum_{j_1,j_2=1}^{n_\ell} W_{i_1 j_1}^{(\ell+1)} W_{i_2 j_2}^{(\ell+1)} \sigma'^{(\ell)}_{j_1;\alpha_1} \sigma'^{(\ell)}_{j_2;\alpha_2} \widehat{H}^{(\ell)}_{j_1 j_2;\alpha_1\alpha_2}\right] &= \sum_{j_1,j_2=1}^{n_\ell} \mathbb{E}\left[W_{i_1 j_1}^{(\ell+1)} W_{i_2 j_2}^{(\ell+1)}\right] \mathbb{E}\left[\sigma'^{(\ell)}_{j_1;\alpha_1} \sigma'^{(\ell)}_{j_2;\alpha_2} \widehat{H}^{(\ell)}_{j_1 j_2;\alpha_1\alpha_2}\right] \\ &= \delta_{i_1 i_2} C_W^{(\ell+1)} \left(\frac{1}{n_\ell} \sum_{j=1}^{n_\ell} \mathbb{E}\left[\sigma'^{(\ell)}_{j;\alpha_1} \sigma'^{(\ell)}_{j;\alpha_2} \widehat{H}^{(\ell)}_{j j;\alpha_1\alpha_2}\right]\right) . \end{aligned} \tag{8.15}$$

In particular, we see that the $1/n_\ell$ scaling of the initialization weight variance $C_W^{(\ell+1)}$ is important for ensuring principled behavior of not only the network output, but also the NTK.

Getting things backwards

N.B. the chain-rule factors (8.10) also appear when evaluating the derivative of the network outputs with respect to model parameters

$$\frac{dz^{(L)}_{i;\alpha}}{db^{(\ell)}_j} = \frac{dz^{(L)}_{i;\alpha}}{dz^{(\ell)}_{j;\alpha}}\,, \qquad \frac{dz^{(L)}_{i;\alpha}}{dW^{(\ell)}_{jk}} = \sum_m \frac{dz^{(L)}_{i;\alpha}}{dz^{(\ell)}_{m;\alpha}} \frac{dz^{(\ell)}_{m;\alpha}}{dW^{(\ell)}_{jk}} = \frac{dz^{(L)}_{i;\alpha}}{dz^{(\ell)}_{j;\alpha}} \sigma^{(\ell-1)}_{k;\alpha}\,. \tag{8.16}$$

Evaluating these derivatives gives another neural-network iteration equation,

$$\frac{dz^{(L)}_{i;\alpha}}{dz^{(\ell)}_{j;\alpha}} = \sum_{k=1}^{n_{\ell+1}} \frac{dz^{(L)}_{i;\alpha}}{dz^{(\ell+1)}_{k;\alpha}} \frac{dz^{(\ell+1)}_{k;\alpha}}{dz^{(\ell)}_{j;\alpha}} = \sum_{k=1}^{n_{\ell+1}} \frac{dz^{(L)}_{i;\alpha}}{dz^{(\ell+1)}_{k;\alpha}} W^{(\ell+1)}_{kj} \sigma'^{(\ell)}_{j;\alpha} \quad \text{for} \quad \ell < L\,, \tag{8.17}$$

but in this case for the derivative of the output. In particular, (8.17) is a *backward* equation: starting from the *final* condition

$$\frac{dz^{(L)}_{i;\alpha}}{dz^{(L)}_{j;\alpha}} = \delta_{ij}\,, \tag{8.18}$$

we iterate layer-to-layer backwards, $\ell = L-1, L-2, \ldots, 1$, by sequential multiplications of the chain-rule factors (8.10).

An algorithm based on this backward equation can be efficiently implemented to compute derivatives with respect to the model parameters and, for that reason, is used by most deep-learning packages to compute the gradient as part of any neural network gradient-based learning algorithm. Such a package typically lets practitioners specify a deep learning model by defining a **forward pass** – for MLPs a practitioner would implement the forward equation (8.9) – and then the package will automatically work out the **backward pass** – i.e. for MLPs it would implement (8.17). The computational algorithm based on (8.17) is termed **backpropagation**, which was discovered and rediscovered numerous times in the history of deep learning. Among them, a particular rediscovery [15] was essential in convincing the machine learning community that multilayer neural networks can be trained efficiently.

All that said, when evaluating the NTK in the effective theory it's essential that we use the forward equation (8.12) rather than getting things backwards. In the next three-plus-one sections, we'll indeed use the forward equation to recursively compute the *joint* initialization distribution for the ℓ-th-layer preactivations *and* the ℓ-th-layer NTK:

$$p\Big(z^{(\ell)}, \widehat{H}^{(\ell)}\Big|\mathcal{D}\Big) \equiv p\begin{pmatrix} z^{(\ell)}(x_1) & z^{(\ell)}(x_2) & \ldots & z^{(\ell)}(x_{N_\mathcal{D}}) \\ \widehat{H}^{(\ell)}(x_1,x_1) & \widehat{H}^{(\ell)}(x_1,x_2) & \ldots & \widehat{H}^{(\ell)}(x_1,x_{N_\mathcal{D}}) \\ \widehat{H}^{(\ell)}(x_2,x_1) & \widehat{H}^{(\ell)}(x_2,x_2) & \ldots & \widehat{H}^{(\ell)}(x_2,x_{N_\mathcal{D}}) \\ \vdots & \vdots & \ddots & \vdots \\ \widehat{H}^{(\ell)}(x_{N_\mathcal{D}},x_1) & \widehat{H}^{(\ell)}(x_{N_\mathcal{D}},x_2) & \ldots & \widehat{H}^{(\ell)}(x_{N_\mathcal{D}},x_{N_\mathcal{D}}) \end{pmatrix}. \tag{8.19}$$

(On the right-hand side, we've suppressed neural indices while explicitly writing out the input dependence. This emphasizes that the preactivations are each functions of a single input and that the NTK components are each functions of a pair of inputs.)

8.1 First Layer: Deterministic NTK

Recall from §4.1 that at initialization the first-layer preactivations,

$$z_{i;\alpha}^{(1)} \equiv b_i^{(1)} + \sum_{j=1}^{n_0} W_{ij}^{(1)} x_{j;\alpha} \,, \tag{8.20}$$

are distributed according to a zero-mean Gaussian distribution,

$$p\left(z^{(1)}\middle|\mathcal{D}\right) = \frac{1}{\left|2\pi G^{(1)}\right|^{\frac{n_1}{2}}} \exp\left(-\frac{1}{2}\sum_{i=1}^{n_1}\sum_{\alpha_1,\alpha_2\in\mathcal{D}} G_{(1)}^{\alpha_1\alpha_2} z_{i;\alpha_1}^{(1)} z_{i;\alpha_2}^{(1)}\right) , \tag{8.21}$$

with the first-layer deterministic metric – a function of the inputs – given by

$$G_{\alpha_1\alpha_2}^{(1)} \equiv C_b^{(1)} + C_W^{(1)} \frac{1}{n_0}\sum_{j=1}^{n_0} x_{j;\alpha_1} x_{j;\alpha_2} \,. \tag{8.22}$$

In particular, the quadratic action in the exponent of (8.21) indicates the absence of interactions between neurons. This enables us to factor expectation values of first-layer observables into separate Gaussian integrals for each neuron.

The first-layer NTK at initialization is even more trivial and can be read off from the original definition of the NTK (8.7) by plugging in the derivatives (8.11) and remembering the identification $\sigma_{i;\alpha}^{(0)} = x_{i;\alpha}$:

$$\widehat{H}_{i_1 i_2;\alpha_1\alpha_2}^{(1)} = \delta_{i_1 i_2}\left[\lambda_b^{(1)} + \lambda_W^{(1)}\left(\frac{1}{n_0}\sum_{j=1}^{n_0} x_{j;\alpha_1} x_{j;\alpha_2}\right)\right] \equiv \delta_{i_1 i_2} H_{\alpha_1\alpha_2}^{(1)} \,. \tag{8.23}$$

Like the first-layer metric, the first-layer NTK is completely deterministic – hence no hat on the right-hand side of the equation – and is diagonal in its neural indices. Remembering our exposition on the off-diagonal components of the NTK in §7.2, this in particular means that, for single-layer networks, a feature captured by a particular neuron cannot affect the gradient-descent update for another feature on any other neuron.

Finally, recalling our discussion of deterministic distributions in §2.3, the joint distribution of the first-layer preactivations and the first-layer NTK can be written as

$$p\left(z^{(1)}, \widehat{H}^{(1)}\middle|\mathcal{D}\right) = p\left(z^{(1)}\middle|\mathcal{D}\right) \prod_{(i_1 i_2),(\alpha_1\alpha_2)} \delta\left(\widehat{H}_{i_1 i_2;\alpha_1\alpha_2}^{(1)} - \delta_{i_1 i_2} H_{\alpha_1\alpha_2}^{(1)}\right) , \tag{8.24}$$

where the product of the Dirac delta functions runs over all pairs of neural indices and sample indices. Just as the first-layer preactivation distribution was representative of

deeper layers in the infinite-width limit, this first-layer joint distribution is also representative of deeper-layer joint distributions in the infinite-width limit: the preactivation distribution is exactly Gaussian, the NTK distribution is completely deterministic, and there is no correlation between the two, i.e., they are statistically independent from each other once the dataset is fixed.

8.2 Second Layer: Fluctuating NTK

Now, let us see how finite-width corrections can modify this picture in the second layer.

Recall from §4.2 that the second-layer preactivations are given by

$$z^{(2)}_{i;\alpha} = b^{(2)}_i + \sum_{j=1}^{n_1} W^{(2)}_{ij} \sigma^{(1)}_{j;\alpha} \,. \tag{8.25}$$

After marginalizing over the first-layer preactivations $z^{(1)}$, the correlated fluctuations of the preactivations in the first layer resulted in nontrivial interaction between different neurons in the second layer. At the leading nontrivial order in $1/n_1$, this led to a nearly-Gaussian distribution with a quartic action (4.60) for the second-layer preactivations, with the leading non-Gaussianity captured by a nonzero connected four-point correlator.

As for the NTK, looking at its forward equation (8.12) and recalling that the first-layer NTK is deterministic (8.23), we see that the second-layer NTK is given by

$$\widehat{H}^{(2)}_{i_1 i_2;\alpha_1\alpha_2} = \delta_{i_1 i_2} \left[\lambda_b^{(2)} + \lambda_W^{(2)} \left(\frac{1}{n_1} \sum_{j=1}^{n_1} \sigma^{(1)}_{j;\alpha_1} \sigma^{(1)}_{j;\alpha_2} \right) \right] + \sum_{j=1}^{n_1} W^{(2)}_{i_1 j} W^{(2)}_{i_2 j} \, \sigma'^{(1)}_{j;\alpha_1} \sigma'^{(1)}_{j;\alpha_2} \, H^{(1)}_{\alpha_1\alpha_2} \,. \tag{8.26}$$

This second-layer NTK depends on two sets of stochastic variables, the weights $W^{(2)}_{ij}$ and the first-layer preactivations $z^{(1)}_{i;\alpha}$, and hence it fluctuates.

To compute its mean we take an expectation of (8.26), finding

$$\begin{aligned} &\mathbb{E}\left[\widehat{H}^{(2)}_{i_1 i_2;\alpha_1\alpha_2}\right] \\ =&\delta_{i_1 i_2} \left[\lambda_b^{(2)} + \lambda_W^{(2)} \left(\frac{1}{n_1} \sum_{j=1}^{n_1} \mathbb{E}\left[\sigma^{(1)}_{j;\alpha_1} \sigma^{(1)}_{j;\alpha_2} \right] \right) \right] + \sum_{j=1}^{n_1} \mathbb{E}\left[W^{(2)}_{i_1 j} W^{(2)}_{i_2 j} \right] \mathbb{E}\left[\sigma'^{(1)}_{j;\alpha_1} \sigma'^{(1)}_{j;\alpha_2} \right] H^{(1)}_{\alpha_1\alpha_2} \\ =&\delta_{i_1 i_2} \left[\lambda_b^{(2)} + \lambda_W^{(2)} \left\langle \sigma_{\alpha_1} \sigma_{\alpha_2} \right\rangle_{G^{(1)}} + C_W^{(2)} \left\langle \sigma'_{\alpha_1} \sigma'_{\alpha_2} \right\rangle_{G^{(1)}} H^{(1)}_{\alpha_1\alpha_2} \right] \\ \equiv&\delta_{i_1 i_2} \, H^{(2)}_{\alpha_1\alpha_2} \,. \end{aligned} \tag{8.27}$$

Here, in the second line, the expectation of the recursive term factorized because the second-layer weights $W^{(2)}_{ij}$ are statistically independent from the first-layer preactivations. Additionally, in the third line we recalled (4.27), in which we showed that the two-point correlators can be expressed as a separate Gaussian expectations for each

neuron, with the variance given by the first-layer metric $G^{(1)}$.[4] Further, inspecting our answer (8.27), we see that the mean of the second-layer NTK is diagonal in its neural indices. Furthermore, we separated the part that encodes the sample dependence and symbolized it by taking off its hat because it is a mean, not a stochastic variable.

Now, let's compute the variance. First, define the second-layer NTK fluctuation through our usual decomposition,

$$\widehat{H}^{(2)}_{i_1 i_2;\alpha_1\alpha_2} \equiv \delta_{i_1 i_2} H^{(2)}_{\alpha_1\alpha_2} + \widehat{\Delta H}^{(2)}_{i_1 i_2;\alpha_1\alpha_2} \,, \tag{8.28}$$

so that the expectation of the magnitude of this fluctuation determines the covariance:

$$\mathbb{E}\left[\widehat{\Delta H}^{(2)}_{i_1 i_2;\alpha_1\alpha_2}\widehat{\Delta H}^{(2)}_{i_3 i_4;\alpha_3\alpha_4}\right] = \mathbb{E}\left[\widehat{H}^{(2)}_{i_1 i_2;\alpha_1\alpha_2}\widehat{H}^{(2)}_{i_3 i_4;\alpha_3\alpha_4}\right] - \mathbb{E}\left[\widehat{H}^{(2)}_{i_1 i_2;\alpha_1\alpha_2}\right]\mathbb{E}\left[\widehat{H}^{(2)}_{i_3 i_4;\alpha_3\alpha_4}\right] . \tag{8.29}$$

Substituting in our expression (8.26) for the second-layer stochastic NTK and using the independence of the second-layer weights from the first-layer preactivations, we find a complicated-looking result

$$\begin{aligned}
&\mathbb{E}\left[\widehat{\Delta H}^{(2)}_{i_1 i_2;\alpha_1\alpha_2}\widehat{\Delta H}^{(2)}_{i_3 i_4;\alpha_3\alpha_4}\right] \qquad (8.30)\\
=&\frac{1}{n_1}\delta_{i_1 i_2}\delta_{i_3 i_4}\Big\{ \left(\lambda^{(2)}_W\right)^2 \left[\langle \sigma_{\alpha_1}\sigma_{\alpha_2}\sigma_{\alpha_3}\sigma_{\alpha_4}\rangle_{G^{(1)}} - \langle \sigma_{\alpha_1}\sigma_{\alpha_2}\rangle_{G^{(1)}} \langle \sigma_{\alpha_3}\sigma_{\alpha_4}\rangle_{G^{(1)}} \right] \\
&\qquad + C^{(2)}_W H^{(1)}_{\alpha_1\alpha_2}\lambda^{(2)}_W \left[\langle \sigma'_{\alpha_1}\sigma'_{\alpha_2}\sigma_{\alpha_3}\sigma_{\alpha_4}\rangle_{G^{(1)}} - \langle \sigma'_{\alpha_1}\sigma'_{\alpha_2}\rangle_{G^{(1)}} \langle \sigma_{\alpha_3}\sigma_{\alpha_4}\rangle_{G^{(1)}} \right] \\
&\qquad + \lambda^{(2)}_W C^{(2)}_W H^{(1)}_{\alpha_3\alpha_4} \left[\langle \sigma_{\alpha_1}\sigma_{\alpha_2}\sigma'_{\alpha_3}\sigma'_{\alpha_4}\rangle_{G^{(1)}} - \langle \sigma_{\alpha_1}\sigma_{\alpha_2}\rangle_{G^{(1)}} \langle \sigma'_{\alpha_3}\sigma'_{\alpha_4}\rangle_{G^{(1)}} \right] \\
&\qquad + \left(C^{(2)}_W\right)^2 H^{(1)}_{\alpha_1\alpha_2}H^{(1)}_{\alpha_3\alpha_4} \left[\langle \sigma'_{\alpha_1}\sigma'_{\alpha_2}\sigma'_{\alpha_3}\sigma'_{\alpha_4}\rangle_{G^{(1)}} - \langle \sigma'_{\alpha_1}\sigma'_{\alpha_2}\rangle_{G^{(1)}} \langle \sigma'_{\alpha_3}\sigma'_{\alpha_4}\rangle_{G^{(1)}} \right] \Big\} \\
&+ \frac{1}{n_1}\left(\delta_{i_1 i_3}\delta_{i_2 i_4} + \delta_{i_1 i_4}\delta_{i_2 i_3}\right)\left(C^{(2)}_W\right)^2 H^{(1)}_{\alpha_1\alpha_2}H^{(1)}_{\alpha_3\alpha_4} \langle \sigma'_{\alpha_1}\sigma'_{\alpha_2}\sigma'_{\alpha_3}\sigma'_{\alpha_4}\rangle_{G^{(1)}} \,.
\end{aligned}$$

To get this expression, we recalled not only (4.27) for the two-point correlators, but also both (4.28) and (4.29) for the different pairings of the four-point correlators, with the pairings depending on whether all activations are on the same neuron or are on two different neurons, respectively. As with our computation of the mean above, the computations of these four-point correlators proceed similarly regardless of whether an activation has a derivative or not.

To help make sense of this rather ugly expression (8.30), let's first decompose the

[4] Note that the logic around (4.27) is the same whether or not the Gaussian expectation is of activations or derivatives of the activation. In other words, for the first-layer preactivations we also have $\mathbb{E}\left[\sigma'^{(1)}_{j;\alpha_1}\sigma'^{(1)}_{j;\alpha_2}\right] = \langle \sigma'_{\alpha_1}\sigma'_{\alpha_2}\rangle_{G^{(1)}}$.

second-layer NTK variance into a sum of two different types of tensors

$$\begin{aligned}&\mathbb{E}\left[\widehat{\Delta H}^{(2)}_{i_1 i_2;\alpha_1\alpha_2}\widehat{\Delta H}^{(2)}_{i_3 i_4;\alpha_3\alpha_4}\right] \\ \equiv&\frac{1}{n_1}\left[\delta_{i_1 i_2}\delta_{i_3 i_4}A^{(2)}_{(\alpha_1\alpha_2)(\alpha_3\alpha_4)}+\delta_{i_1 i_3}\delta_{i_2 i_4}B^{(2)}_{\alpha_1\alpha_3\alpha_2\alpha_4}+\delta_{i_1 i_4}\delta_{i_2 i_3}B^{(2)}_{\alpha_1\alpha_4\alpha_2\alpha_3}\right].\end{aligned} \tag{8.31}$$

This decomposition was motivated by the pattern of Kronecker deltas that appear in (8.30). Next, by comparing this to our original expression (8.30), we see that these tensors are given by

$$A^{(2)}_{(\alpha_1\alpha_2)(\alpha_3\alpha_4)} = \left\langle \widehat{\Omega}^{(2)}_{\alpha_1\alpha_2}\widehat{\Omega}^{(2)}_{\alpha_3\alpha_4}\right\rangle_{G^{(1)}} - \left\langle \widehat{\Omega}^{(2)}_{\alpha_1\alpha_2}\right\rangle_{G^{(1)}}\left\langle \widehat{\Omega}^{(2)}_{\alpha_3\alpha_4}\right\rangle_{G^{(1)}}, \tag{8.32}$$

$$B^{(2)}_{\alpha_1\alpha_3\alpha_2\alpha_4} = \left(C^{(2)}_W\right)^2 H^{(1)}_{\alpha_1\alpha_2}H^{(1)}_{\alpha_3\alpha_4}\left\langle \sigma'_{\alpha_1}\sigma'_{\alpha_2}\sigma'_{\alpha_3}\sigma'_{\alpha_4}\right\rangle_{G^{(1)}}, \tag{8.33}$$

where on the first line we've introduced an auxiliary stochastic variable,

$$\widehat{\Omega}^{(2)}_{\alpha_1\alpha_2} \equiv \lambda^{(2)}_W\,\sigma^{(1)}_{\alpha_1}\sigma^{(1)}_{\alpha_2} + C^{(2)}_W H^{(1)}_{\alpha_1\alpha_2}\,\sigma'^{(1)}_{\alpha_1}\sigma'^{(1)}_{\alpha_2}, \tag{8.34}$$

in order to remedy the ugliness of what would have otherwise been a very long expression.[5] From (8.32) and (8.33) we see clearly that these tensors are of order one. Given (8.31), this in turn means that the second-layer NTK variance is suppressed by $1/n_1$ in the large-width limit. In other words, the second-layer NTK is deterministic in the strict infinite-width limit, but in backing off that limit it fluctuates according to (8.31), (8.32), and (8.33).

Moreover, at finite width the second-layer NTK not only fluctuates, but also has nontrivial cross correlation with the second-layer preactivations. This can ultimately be traced to the fact that the second-layer preactivations (8.25) and the second-layer NTK (8.26) are both functions of the same stochastic variables: the second-layer weights $W^{(2)}_{ij}$ and the first-layer preactivations $z^{(1)}_{i;\alpha}$.

This cross correlation can be computed analogously to the way we computed the NTK mean and variance. Substituting in the definition of the second-layer preactivations (8.25) and the second-layer NTK (8.26), and again using the statistical indepen-

[5]Note that $A^{(2)}_{(\alpha_1\alpha_2)(\alpha_3\alpha_4)}$ (8.32) has the same symmetries as the four-point vertex $V^{(\ell)}_{(\alpha_1\alpha_2)(\alpha_3\alpha_4)}$. In particular, it's symmetric under exchanges of sample indices $\alpha_1 \leftrightarrow \alpha_2$, $\alpha_3 \leftrightarrow \alpha_4$, and $(\alpha_1\alpha_2) \leftrightarrow (\alpha_3\alpha_4)$, and this symmetry persists to deeper layers, cf. (8.97).

Now, you might raise your hand and say that $B^{(2)}_{\alpha_1\alpha_3\alpha_2\alpha_4}$ (8.33) also respects the same symmetry. That's correct here, but this symmetry will be broken in deeper layers. In general – cf. (8.89) – $B^{(\ell)}_{\alpha_1\alpha_3\alpha_2\alpha_4}$ will be symmetric under $(\alpha_1\alpha_2) \leftrightarrow (\alpha_3\alpha_4)$ and $(\alpha_1\alpha_3) \leftrightarrow (\alpha_2\alpha_4)$ but *not* under $\alpha_1 \leftrightarrow \alpha_2$ or $\alpha_3 \leftrightarrow \alpha_4$ individually.

dence of the second-layer weights $W_{ij}^{(2)}$ from the first-layer preactivations $z_{i;\alpha}^{(1)}$, we find

$$\mathbb{E}\left[z_{i_1;\alpha_1}^{(2)}\widehat{\Delta H}_{i_2 i_3;\alpha_2\alpha_3}^{(2)}\right] = 0\,, \tag{8.35}$$

$$\begin{aligned}\mathbb{E}\left[z_{i_1;\alpha_1}^{(2)}z_{i_2;\alpha_2}^{(2)}\widehat{\Delta H}_{i_3 i_4;\alpha_3\alpha_4}^{(2)}\right] &= \mathbb{E}\left[z_{i_1;\alpha_1}^{(2)}z_{i_2;\alpha_2}^{(2)}\widehat{H}_{i_3 i_4;\alpha_3\alpha_4}^{(2)}\right] - \mathbb{E}\left[z_{i_1;\alpha_1}^{(2)}z_{i_2;\alpha_2}^{(2)}\right]\mathbb{E}\left[\widehat{H}_{i_3 i_4;\alpha_3\alpha_4}^{(2)}\right] \\ =&\frac{1}{n_1}\delta_{i_1 i_2}\delta_{i_3 i_4}\Bigg\{\lambda_W^{(2)}C_W^{(2)}\Big[\left\langle\sigma_{\alpha_1}\sigma_{\alpha_2}\sigma_{\alpha_3}\sigma_{\alpha_4}\right\rangle_{G^{(1)}} - \left\langle\sigma_{\alpha_1}\sigma_{\alpha_2}\right\rangle_{G^{(1)}}\left\langle\sigma_{\alpha_3}\sigma_{\alpha_4}\right\rangle_{G^{(1)}}\Big] \\ &\qquad + \left(C_W^{(2)}\right)^2 H_{\alpha_3\alpha_4}^{(1)}\Big[\left\langle\sigma_{\alpha_1}\sigma_{\alpha_2}\sigma'_{\alpha_3}\sigma'_{\alpha_4}\right\rangle_{G^{(1)}} - \left\langle\sigma_{\alpha_1}\sigma_{\alpha_2}\right\rangle_{G^{(1)}}\left\langle\sigma'_{\alpha_3}\sigma'_{\alpha_4}\right\rangle_{G^{(1)}}\Big]\Bigg\} \\ &+ \frac{1}{n_1}\left(\delta_{i_1 i_3}\delta_{i_2 i_4} + \delta_{i_1 i_4}\delta_{i_2 i_3}\right)\left(C_W^{(2)}\right)^2 H_{\alpha_3\alpha_4}^{(1)}\left\langle\sigma_{\alpha_1}\sigma_{\alpha_2}\sigma'_{\alpha_3}\sigma'_{\alpha_4}\right\rangle_{G^{(1)}}\,.\end{aligned} \tag{8.36}$$

Here, as for the variance (8.30), we recalled the suitably generalized versions of (4.27), (4.28), and (4.29) for the two- and four-point correlators. Thus, we see that the first measure of cross correlation between the second-layer preactivations and the second-layer NTK (8.35) vanishes, but the second one (8.36) is nonzero at finite width.

To aid us in our deep-layer analysis, it will be convenient to decompose this cross correlation (8.36) into two tensors with sample indices only, just as we did for the variance in (8.31):

$$\begin{aligned}&\mathbb{E}\left[z_{i_1;\alpha_1}^{(2)}z_{i_2;\alpha_2}^{(2)}\widehat{\Delta H}_{i_3 i_4;\alpha_3\alpha_4}^{(2)}\right] \\ =&\frac{1}{n_1}\left[\delta_{i_1 i_2}\delta_{i_3 i_4}D_{\alpha_1\alpha_2\alpha_3\alpha_4}^{(2)} + \delta_{i_1 i_3}\delta_{i_2 i_4}F_{\alpha_1\alpha_3\alpha_2\alpha_4}^{(2)} + \delta_{i_1 i_4}\delta_{i_2 i_3}F_{\alpha_1\alpha_4\alpha_2\alpha_3}^{(2)}\right]\,.\end{aligned} \tag{8.37}$$

Comparing this decomposition with our explicit formula for the correlator (8.36), we can identify expressions for these tensors

$$D_{\alpha_1\alpha_2\alpha_3\alpha_4}^{(2)} = C_W^{(2)}\left[\left\langle\sigma_{\alpha_1}\sigma_{\alpha_2}\widehat{\Omega}_{\alpha_3\alpha_4}^{(2)}\right\rangle_{G^{(1)}} - \left\langle\sigma_{\alpha_1}\sigma_{\alpha_2}\right\rangle_{G^{(1)}}\left\langle\widehat{\Omega}_{\alpha_3\alpha_4}^{(2)}\right\rangle_{G^{(1)}}\right]\,, \tag{8.38}$$

$$F_{\alpha_1\alpha_3\alpha_2\alpha_4}^{(2)} = \left(C_W^{(2)}\right)^2 H_{\alpha_3\alpha_4}^{(1)}\left\langle\sigma_{\alpha_1}\sigma_{\alpha_2}\sigma'_{\alpha_3}\sigma'_{\alpha_4}\right\rangle_{G^{(1)}}\,, \tag{8.39}$$

where we've also recalled the stochastic tensor $\widehat{\Omega}_{\alpha_1\alpha_2}^{(2)}$ defined in (8.34).[6] Just as for $A^{(2)}$ and $B^{(2)}$ above, both $D^{(2)}$ and $F^{(2)}$ are manifestly of order one. Similar to the second-layer NTK variance, this means that the cross correlator (8.37) is suppressed by $1/n_1$ in the large-width limit, vanishing in the strict infinite-width limit.

In summary, the joint distribution of the second-layer preactivations and second-layer NTK,

$$p\left(z^{(2)}, \widehat{H}^{(2)}\middle|\mathcal{D}\right)\,, \tag{8.40}$$

[6]The cross-correlation tensor $D_{\alpha_1\alpha_2\alpha_3\alpha_4}^{(2)}$ (8.38) – and more generally $D_{\alpha_1\alpha_2\alpha_3\alpha_4}^{(\ell)}$ in deeper layers, cf. (8.77) – is symmetric under exchanges of sample indices $\alpha_1 \leftrightarrow \alpha_2$ and $\alpha_3 \leftrightarrow \alpha_4$, but of course *not* under $(\alpha_1\alpha_2) \leftrightarrow (\alpha_3\alpha_4)$. The other tensor $F_{\alpha_1\alpha_3\alpha_2\alpha_4}^{(2)}$ (8.39) respects this same symmetry in the second layer, but has no symmetry at all in deeper layers, cf. (8.79).

at leading nontrivial order in the $1/n$ expansion is *nearly-Gaussian distribution* with *(i)* a quartic interaction among preactivations on different neurons, *(ii)* a fluctuating NTK, and *(iii)* cross correlation between the preactivations and NTK. All of these finite-width effects become more complicated for deeper layers.

8.3 Deeper Layers: Accumulation of NTK Fluctuations

As before with §4.1 ‖ §8.1 and §4.2 ‖ §8.2, this section parallels §4.3. In §4.3, we investigated the nearly-Gaussian distribution of preactivations $p\Big(z^{(\ell+1)}\Big|\mathcal{D}\Big)$ at finite width by considering an interlayer joint distribution $p\Big(z^{(\ell+1)}, z^{(\ell)}\Big|\mathcal{D}\Big)$ and then integrating out the ℓ-th-layer preactivations. In particular, due to correlated dependence on the preactivations in previous layers, the non-Gaussianity in the preactivation distribution accumulated as depth increased, manifesting itself in the running four-point vertex $V^{(\ell)}$.

This same mechanism makes the NTK fluctuations accumulate, amplifying the NTK variance as well as the cross correlation between the NTK and preactivations. In this section, we will derive recursions for the NTK mean, the NTK-preactivation cross correlation, and the NTK variance that together determine the ℓ-th-layer joint distribution at leading nontrivial order in $1/n$. What follows is a *goode olde calculation*, so please sharpen your quills, unfurl your parchment, and inform your majordomo that you require a cleared schedule for the rest of the day.

8.3.0 *Inter*lude: *Inter*layer Correlations

If you have an eidetic memory, then perhaps you recall that the main complication with our derivation of the general $(\ell+1)$-th-layer preactivation statistics – as compared to the second layer statistics – was that the ℓ-th-layer preactivation distribution $p\Big(z^{(\ell)}\Big|\mathcal{D}\Big)$ was also non-Gaussian, unlike the Gaussian preactivation distribution in the first layer. For such a nearly-Gaussian distribution $p\Big(z^{(\ell)}\Big|\mathcal{D}\Big)$, *interactions* imply a nontrivial **intralayer correlation** between observables of the preactivations across different neurons $i_1 \neq i_2$. Specifically, the covariance of two arbitrary single-neuron functions $\mathcal{F}\Big(z^{(\ell)}_{i_1;\mathcal{A}_1}\Big)$ and $\mathcal{G}\Big(z^{(\ell)}_{i_2;\mathcal{A}_2}\Big)$ depending on data subsamples $\mathcal{A}_1, \mathcal{A}_2 \subset \mathcal{D}$, respectively, is given by (4.64) and reprinted here:

$$\begin{aligned}
&\text{Cov}\Big[\mathcal{F}\Big(z^{(\ell)}_{i_1;\mathcal{A}_1}\Big),\, \mathcal{G}\Big(z^{(\ell)}_{i_2;\mathcal{A}_2}\Big)\Big] \equiv \mathbb{E}\left[\mathcal{F}\Big(z^{(\ell)}_{i_1;\mathcal{A}_1}\Big)\,\mathcal{G}\Big(z^{(\ell)}_{i_2;\mathcal{A}_2}\Big)\right] - \mathbb{E}\left[\mathcal{F}\Big(z^{(\ell)}_{i_1;\mathcal{A}_1}\Big)\right]\mathbb{E}\left[\mathcal{G}\Big(z^{(\ell)}_{i_2;\mathcal{A}_2}\Big)\right] \\
&= \sum_{\beta_1,\ldots,\beta_4 \in \mathcal{D}} \frac{1}{4n_{\ell-1}} V^{(\beta_1\beta_2)(\beta_3\beta_4)}_{(\ell)} \Big\langle \Big(z_{\beta_1} z_{\beta_2} - G^{(\ell)}_{\beta_1\beta_2}\Big)\,\mathcal{F}(z_{\mathcal{A}_1})\Big\rangle_{G^{(\ell)}} \Big\langle \Big(z_{\beta_3} z_{\beta_4} - G^{(\ell)}_{\beta_3\beta_4}\Big)\,\mathcal{G}(z_{\mathcal{A}_2})\Big\rangle_{G^{(\ell)}} \\
&\quad + O\Big(\frac{1}{n^2}\Big)\,. \qquad (8.41)
\end{aligned}$$

In this reprinting, we implicitly substituted in our leading large-width expressions (4.81) and (4.82) for the quadratic coupling $g_{(\ell)}$ and quartic coupling $v_{(\ell)}$, respectively. We

have also recalled our long-forgotten shorthand notation for the covariance of random variables (1.53), which we will use judiciously throughout this section. This *intralayer* formula will soon prove itself useful.

Enlarging our view to the preactivation-NTK joint distribution (8.19), we'll encounter another complication due to ***inter*layer correlation** of the form

$$\mathbb{E}\left[\mathcal{O}\left(z^{(\ell+1)}\right)\mathcal{P}\left(W^{(\ell+1)}\right)\mathcal{Q}\left(z^{(\ell)},\widehat{H}^{(\ell)}\right)\right], \tag{8.42}$$

where $\mathcal{O}$ is some function of $(\ell+1)$-th-layer preactivations, $\mathcal{P}$ is a polynomial of $(\ell+1)$-th-layer weights, and $\mathcal{Q}$ is a function of ℓ-th-layer preactivations and the ℓ-th-layer NTK. For instance, taking the NTK-preactivation cross correlation

$$\mathbb{E}\left[z^{(\ell+1)}_{i_1;\alpha_1}z^{(\ell+1)}_{i_2;\alpha_2}\widehat{H}^{(\ell+1)}_{i_3i_4;\alpha_3\alpha_4}\right] \tag{8.43}$$

and unraveling the NTK through its forward equation (8.12), we get an interlayer correlation of the form (8.42) with $\mathcal{P}=1$ from the additive term in the square brackets and an interlayer correlation of the same form with $\mathcal{P}=W^{(\ell+1)}_{i_3j_3}W^{(\ell+1)}_{i_4j_4}$ from the recursive term. While it was simple enough to evaluate such an expectation for the second layer, it's somewhat subtle for a general layer.

That said, there's actually a pretty neat trick that lets us reduce such interlayer correlations (8.42) to expectations of solely ℓ-th-layer variables. Such expectations can subsequently be evaluated with the *intra*layer formula (8.41) above. Let us now teach you this magic trick before diving deep into learning the deeper-layer analysis.[7]

First, using the definition of the expectation and the conditional structure of the distribution, the interlayer correlation (8.42) can be expressed as (suppressing all indices)

$$\begin{aligned}
&\mathbb{E}\left[\mathcal{O}\left(z^{(\ell+1)}\right)\mathcal{P}\left(W^{(\ell+1)}\right)\mathcal{Q}\left(z^{(\ell)},\widehat{H}^{(\ell)}\right)\right] \\
&=\int dz^{(\ell)}d\widehat{H}^{(\ell)}p\left(z^{(\ell)},\widehat{H}^{(\ell)}\middle|\mathcal{D}\right)\mathcal{Q}\left(z^{(\ell)},\widehat{H}^{(\ell)}\right) \\
&\quad\times\left[\int db^{(\ell+1)}dW^{(\ell+1)}p\left(b^{(\ell+1)}\right)p\left(W^{(\ell+1)}\right)\mathcal{P}\left(W^{(\ell+1)}\right)\right. \\
&\qquad\left.\times\int dz^{(\ell+1)}p\left(z^{(\ell+1)}\middle|b^{(\ell+1)},W^{(\ell+1)},z^{(\ell)}\right)\mathcal{O}\left(z^{(\ell+1)}\right)\right].
\end{aligned} \tag{8.44}$$

Our strategy will be to *integrate out* or marginalize over the $(\ell+1)$-th-layer parameters in order to express the object inside the square bracket as a function of the ℓ-th-layer variables *only*. In this way, the entire object will become an ℓ-th-layer expectation that we already know how to handle.

Second, rather than working with an abstract polynomial $\mathcal{P}$, let's construct a *generating function* for these interlayer correlations through the use of a *source term*:

$$\mathcal{P}\left(W^{(\ell+1)}\right)=e^{\sum_{i,j}\mathcal{J}_{ij}W^{(\ell+1)}_{ij}}. \tag{8.45}$$

[7] As similar interlayer correlations appear in §11, we'll keep our exposition completely general rather than specializing to the NTK-preactivation cross correlation (8.43).

Recall from your pretraining days (§1.1) that a generating function such as (8.45) could be used to evaluate expectations such as (8.42) with any polynomial insertions of weights $W_{ij}^{(\ell+1)}$. To do this, we differentiate the evaluated generating function some number of times with respect to the source $\mathcal{J}_{ij}$ and then set the source to zero.

Now with our choice (8.45) in mind for $\mathcal{P}$, we can explicitly evaluate the expression in the square brackets in (8.44) as follows: *(i)* recall the initialization distributions for the biases (2.21) and weights (2.22), *(ii)* recall from (2.34) that the conditional distribution $p\Big(z^{(\ell+1)}\Big|b^{(\ell+1)},W^{(\ell+1)},z^{(\ell)}\Big)$ encodes the MLP forward equation (8.9) as a Dirac delta function, and finally *(iii)* recall the integral representation of the Dirac delta function (2.32). All together, this gives the following set of integrals

$$\int\left[\prod_i \frac{db_i}{\sqrt{2\pi C_b^{(\ell+1)}}}\right]\left[\prod_{i,j}\frac{dW_{ij}}{\sqrt{2\pi C_W^{(\ell+1)}/n_\ell}}\right]\left[\prod_{i,\alpha}\frac{d\Lambda_i{}^\alpha\, dz_{i;\alpha}^{(\ell+1)}}{2\pi}\right]\mathcal{O}\Big(z^{(\ell+1)}\Big) \tag{8.46}$$
$$\times\exp\left[-\sum_i\frac{b_i^2}{2C_b^{(\ell+1)}}-\sum_{i,j}\frac{n_\ell W_{ij}^2}{2C_W^{(\ell+1)}}+i\sum_{i,\alpha}\Lambda_i{}^\alpha\left(z_{i;\alpha}^{(\ell+1)}-b_i-\sum_j W_{ij}\,x_{j;\alpha}\right)+\sum_{i,j}\mathcal{J}_{ij}W_{ij}\right],$$

which we recognize as the good-old *Hubbard-Stratonovich transformation* that we first encountered in §4.1.

Next, as we did in §4.1 and in high school, we can *complete the squares* with respect to the biases and weights and integrate them out. The only substantial deviation here from the presentation in §4.1 is that the source term shifts the linear coupling of the weights as

$$-iW_{ij}\sum_\alpha\Lambda_i{}^\alpha\sigma_{j;\alpha}^{(\ell)}\to -iW_{ij}\left(\sum_\alpha\Lambda_i{}^\alpha\sigma_{j;\alpha}^{(\ell)}+i\mathcal{J}_{ij}\right). \tag{8.47}$$

Performing these Gaussian integrals, we find

$$\int\left[\prod_{i,\alpha}\frac{d\Lambda_i{}^\alpha\, dz_{i;\alpha}^{(\ell+1)}}{2\pi}\right]\mathcal{O}\left(z^{(\ell+1)}\right)\exp\left[-\sum_{i,\alpha_1,\alpha_2}\Lambda_i{}^{\alpha_1}\Lambda_i{}^{\alpha_2}\left(\frac{C_b^{(\ell+1)}}{2}+\frac{C_W^{(\ell+1)}}{2n_\ell}\sum_j\sigma_{\alpha_1;j}^{(\ell)}\sigma_{\alpha_2;j}^{(\ell)}\right)\right.$$
$$\left.+\,i\sum_{i,\alpha}\Lambda_i{}^\alpha\left(z_{i;\alpha}^{(\ell+1)}-\frac{C_W^{(\ell+1)}}{n_\ell}\sum_j\mathcal{J}_{ij}\sigma_{j;\alpha}^{(\ell)}\right)+\frac{C_W^{(\ell+1)}}{2n_\ell}\sum_{i,j}\mathcal{J}_{ij}^2\right]. \tag{8.48}$$

Just as in our previous Hubbard-Stratonoviching (4.20), the *stochastic metric* (4.70),

$$\widehat{G}_{\alpha_1\alpha_2}^{(\ell+1)}\equiv C_b^{(\ell+1)}+C_W^{(\ell+1)}\frac{1}{n_\ell}\sum_{j=1}^{n_\ell}\sigma_{j;\alpha_1}^{(\ell)}\sigma_{j;\alpha_2}^{(\ell)}\,, \tag{8.49}$$

appears in the quadratic term of the Hubbard-Stratonovich variables $\Lambda_i{}^\alpha$, while the linear term is slightly modified by a shifting of the preactivations with the subtraction of the quantity

$$\widehat{\mathcal{M}}_{i;\alpha}\equiv C_W^{(\ell+1)}\left(\frac{1}{n_\ell}\sum_{j=1}^{n_\ell}\mathcal{J}_{ij}\sigma_{j;\alpha}^{(\ell)}\right). \tag{8.50}$$

Completing the squares with the Hubbard-Stratonovich variables and integrating them out, we get

$$\frac{\exp\left(\frac{C_W^{(\ell+1)}}{2n_\ell}\sum_{i,j}\mathcal{J}_{ij}^2\right)}{\sqrt{\left|2\pi\widehat{G}^{(\ell+1)}\right|^{n_{\ell+1}}}}\int\left[\prod_{i,\alpha}dz_{i;\alpha}^{(\ell+1)}\right]\mathcal{O}\Big(z^{(\ell+1)}\Big) \tag{8.51}$$
$$\times\exp\left[-\frac{1}{2}\sum_i\sum_{\alpha_1,\alpha_2}\widehat{G}^{\alpha_1\alpha_2}_{(\ell+1)}\left(z_{i;\alpha_1}^{(\ell+1)}-\widehat{\mathcal{M}}_{i;\alpha_1}\right)\left(z_{i;\alpha_2}^{(\ell+1)}-\widehat{\mathcal{M}}_{i;\alpha_2}\right)\right].$$

Ignoring the quadratic source factor $\mathcal{J}_{ij}^2$ outside the integral, this is just a *Gaussian expectation* of $\mathcal{O}$ against the $(\ell+1)$-th-layer preactivation distribution with a mean $\widehat{\mathcal{M}}_{i;\alpha}$ and a variance $\widehat{G}^{(\ell+1)}_{\alpha_1\alpha_2}$. (Make sure you remember our general relativity convention: $\widehat{G}^{\alpha_1\alpha_2}_{(\ell+1)}$ is the inverse of $\widehat{G}^{(\ell+1)}_{\alpha_1\alpha_2}$.)

Now, let's compensate for this mean by shifting the dummy integration variable as $z_{i;\alpha}^{(\ell+1)}\to z_{i;\alpha}^{(\ell+1)}+\widehat{\mathcal{M}}_{i;\alpha_1}$, which yields a compact expression in terms of our *zero-mean* Gaussian expectation notation (4.45):

$$\exp\left(\frac{C_W^{(\ell+1)}}{2n_\ell}\sum_{i,j}\mathcal{J}_{ij}^2\right)\Big\langle\!\Big\langle\mathcal{O}\Big(z^{(\ell+1)}+\widehat{\mathcal{M}}\Big)\Big\rangle\!\Big\rangle_{\widehat{G}^{(\ell+1)}}\,. \tag{8.52}$$

Plugging this result (8.52) back into our interlayer correlation (8.42) and substituting back in for the mean shift (8.50), we arrive at a simple formula for our generating function:

$$\mathbb{E}\left[\mathcal{O}\Big(z^{(\ell+1)}\Big)\,e^{\sum_{i,j}\mathcal{J}_{ij}W_{ij}^{(\ell+1)}}\,\mathcal{Q}\Big(z^{(\ell)},\widehat{H}^{(\ell)}\Big)\right] \tag{8.53}$$
$$=\exp\left(\frac{C_W^{(\ell+1)}}{2n_\ell}\sum_{i,j}\mathcal{J}_{ij}^2\right)\mathbb{E}\left[\Big\langle\!\Big\langle\mathcal{O}\Big(z_{i;\alpha}^{(\ell+1)}+C_W^{(\ell+1)}\frac{1}{n_\ell}\sum_{j=1}^{n_\ell}\mathcal{J}_{ij}\sigma_{j;\alpha}^{(\ell)}\Big)\Big\rangle\!\Big\rangle_{\widehat{G}^{(\ell+1)}}\,\mathcal{Q}\Big(z^{(\ell)},\widehat{H}^{(\ell)}\Big)\right].$$

After performing the Gaussian expectation over the $(\ell+1)$-th-layer preactivations $z_{i;\alpha}^{(\ell+1)}$ – which is typically trivial in all the concrete applications that we'll encounter – the expectation in (8.53) is only with respect to ℓ-th-layer variables.[8] This was our desired result.

To see how to use the generating function (8.53), let's work out some explicit examples. First, consider the case with no weight insertions. Setting the source to zero, $\mathcal{J}_{ij}=0$, we find

$$\mathbb{E}\left[\mathcal{O}\Big(z^{(\ell+1)}\Big)\,\mathcal{Q}\Big(z^{(\ell)},\widehat{H}^{(\ell)}\Big)\right]=\mathbb{E}\left[\Big\langle\!\Big\langle\mathcal{O}\Big(z^{(\ell+1)}\Big)\Big\rangle\!\Big\rangle_{\widehat{G}^{(\ell+1)}}\,\mathcal{Q}\Big(z^{(\ell)},\widehat{H}^{(\ell)}\Big)\right]. \tag{8.54}$$

This formula is not trivial and is not something we knew before: here we see that the correlation between preactivations in neighboring layers is given by first computing a

[8] Recall that the stochastic metric $\widehat{G}^{(\ell+1)}$ (8.49) depends only on the ℓ-th-layer preactivations.

Gaussian expectation of the $(\ell+1)$-th-layer function against the stochastic metric and then taking the full expectation of the resulting ℓ-th-layer quantity.

Next, let's consider two weight insertions. Twice-differentiating the generating function (8.53) by the source as $\frac{d}{d\mathcal{J}_{i_3 j_3}}\frac{d}{d\mathcal{J}_{i_4 j_4}}$ and then setting the source to zero, we get

$$\begin{aligned}
&\mathbb{E}\left[\mathcal{O}\left(z^{(\ell+1)}\right) W_{i_3 j_3}^{(\ell+1)} W_{i_4 j_4}^{(\ell+1)} \mathcal{Q}\left(z^{(\ell)}, \widehat{H}^{(\ell)}\right)\right] \\
=&\delta_{i_3 i_4}\delta_{j_3 j_4}\frac{C_W^{(\ell+1)}}{n_\ell}\mathbb{E}\left[\langle\!\langle \mathcal{O} \rangle\!\rangle_{\widehat{G}^{(\ell+1)}} \mathcal{Q}\left(z^{(\ell)}, \widehat{H}^{(\ell)}\right)\right] \\
&+\left(\frac{C_W^{(\ell+1)}}{n_\ell}\right)^2 \sum_{\beta_3,\beta_4,\gamma_3,\gamma_4} \mathbb{E}\Bigg[\left\langle\!\!\left\langle \left(z_{i_3;\beta_3}^{(\ell+1)} z_{i_4;\beta_4}^{(\ell+1)} - \delta_{i_3 i_4}\widehat{G}_{\beta_3\beta_4}^{(\ell+1)}\right)\mathcal{O}\right\rangle\!\!\right\rangle_{\widehat{G}^{(\ell+1)}} \\
&\qquad\qquad \times \widehat{G}_{(\ell+1)}^{\beta_3\gamma_3}\widehat{G}_{(\ell+1)}^{\beta_4\gamma_4}\sigma_{j_3;\gamma_3}^{(\ell)}\sigma_{j_4;\gamma_4}^{(\ell)}\mathcal{Q}\left(z^{(\ell)}, \widehat{H}^{(\ell)}\right)\Bigg] .
\end{aligned} \tag{8.55}$$

Here, we used integration by parts to exchange the derivatives for a projection as

$$\left\langle\!\!\left\langle \frac{\partial^2 \mathcal{O}}{\partial z_{i_3;\gamma_3}^{(\ell+1)}\partial z_{i_4;\gamma_4}^{(\ell+1)}}\right\rangle\!\!\right\rangle_{\widehat{G}^{(\ell+1)}} = \sum_{\beta_3,\beta_4}\widehat{G}_{(\ell+1)}^{\beta_3\gamma_3}\widehat{G}_{(\ell+1)}^{\beta_4\gamma_4}\left\langle\!\!\left\langle \left(z_{i_3;\beta_3}^{(\ell+1)} z_{i_4;\beta_4}^{(\ell+1)} - \delta_{i_3 i_4}\widehat{G}_{\beta_3\beta_4}^{(\ell+1)}\right)\mathcal{O}\right\rangle\!\!\right\rangle_{\widehat{G}^{(\ell+1)}} . \tag{8.56}$$

Intuitively, the first term in (8.55) comes from forming a Wick contraction with the two weight insertions, while the second term comes from two pairs of Wick contractions, each between an inserted weight and a weight hidden inside the $z^{(\ell+1)}$'s in $\mathcal{O}$.

Thusly, with the *intra*layer formula (8.41) recalled and the *inter*layer formulae (8.54) and (8.55) derived, we are as ready as we'll ever be to recursively analyze the joint statistics of the NTK and preactivations in deeper layers. This concludes our *inter*lude.

8.3.1 NTK Mean

Taking the expectation of the stochastic NTK forward equation (8.12), we get

$$\begin{aligned}
\mathbb{E}\left[\widehat{H}_{i_1 i_2;\alpha_1\alpha_2}^{(\ell+1)}\right] =&\delta_{i_1 i_2}\left[\lambda_b^{(\ell+1)} + \lambda_W^{(\ell+1)}\left(\frac{1}{n_\ell}\sum_{j=1}^{n_\ell}\mathbb{E}\left[\sigma_{j;\alpha_1}^{(\ell)}\sigma_{j;\alpha_2}^{(\ell)}\right]\right)\right] \\
&+\delta_{i_1 i_2}C_W^{(\ell+1)}\frac{1}{n_\ell}\sum_{j=1}^{n_\ell}\mathbb{E}\left[\sigma_{j;\alpha_1}^{\prime(\ell)}\sigma_{j;\alpha_2}^{\prime(\ell)}\widehat{H}_{jj;\alpha_1\alpha_2}^{(\ell)}\right] ,
\end{aligned} \tag{8.57}$$

where, as is now familiar, on the second line we used the independence of the $(\ell+1)$-th-layer weights from the ℓ-th-layer preactivations, and then immediately evaluated the weight expectation. Immediately, we see that NTK mean is diagonal in neural indices at any network depth.

Given that, let's decompose the ℓ-th-layer NTK into a mean and fluctuation as

$$\widehat{H}_{i_1 i_2;\alpha_1\alpha_2}^{(\ell)} \equiv \delta_{i_1 i_2}H_{\alpha_1\alpha_2}^{(\ell)} + \widehat{\Delta H}_{i_1 i_2;\alpha_1\alpha_2}^{(\ell)} , \tag{8.58}$$

where we have denoted the ℓ-th-layer NTK mean as $\delta_{i_1 i_2} H^{(\ell)}_{\alpha_1\alpha_2}$. As before, we separated the part of the mean that encodes the sample dependence and symbolized it without a hat. Substituting this decomposition into our expression (8.57) for the NTK mean, we see that the $(\ell+1)$-th-layer mean obeys a recursion

$$\begin{aligned} H^{(\ell+1)}_{\alpha_1\alpha_2} =& \lambda_b^{(\ell+1)} + \lambda_W^{(\ell+1)} \left(\frac{1}{n_\ell} \sum_{j=1}^{n_\ell} \mathbb{E}\left[\sigma^{(\ell)}_{j;\alpha_1} \sigma^{(\ell)}_{j;\alpha_2} \right] \right) + C_W^{(\ell+1)} H^{(\ell)}_{\alpha_1\alpha_2} \left(\frac{1}{n_\ell} \sum_{j=1}^{n_\ell} \mathbb{E}\left[\sigma'^{(\ell)}_{j;\alpha_1} \sigma'^{(\ell)}_{j;\alpha_2} \right] \right) \\ & + C_W^{(\ell+1)} \left(\frac{1}{n_\ell} \sum_{j=1}^{n_\ell} \mathbb{E}\left[\sigma'^{(\ell)}_{j;\alpha_1} \sigma'^{(\ell)}_{j;\alpha_2} \widehat{\Delta H}^{(\ell)}_{jj;\alpha_1\alpha_2} \right] \right) , \end{aligned} \tag{8.59}$$

depending on both the mean and fluctuation in the previous layer ℓ.

To the leading order in $1/n$, the first two expectation values on the right-hand side of (8.59) are given by Gaussian expectations

$$\mathbb{E}\left[\sigma^{(\ell)}_{j;\alpha_1} \sigma^{(\ell)}_{j;\alpha_2} \right] = \langle \sigma_{\alpha_1} \sigma_{\alpha_2} \rangle_{G^{(\ell)}} + O\left(\frac{1}{n}\right) , \tag{8.60}$$

$$\mathbb{E}\left[\sigma'^{(\ell)}_{j;\alpha_1} \sigma'^{(\ell)}_{j;\alpha_2} \right] = \langle \sigma'_{\alpha_1} \sigma'_{\alpha_2} \rangle_{G^{(\ell)}} + O\left(\frac{1}{n}\right) . \tag{8.61}$$

To see this, note that the first expectation is just the leading Gaussian contribution (4.61) with the non-Gaussian coupling v suppressed as $\sim 1/n$ as per (4.82), and that the evaluation of the second expectation proceeds identically to the first regardless of whether the activation has a derivative or not. Meanwhile, the final expectation on the second line of (8.59) involves an NTK-preactivation cross correlation, which is also suppressed in the large-width limit:

$$\mathbb{E}\left[\sigma'^{(\ell)}_{j;\alpha_1} \sigma'^{(\ell)}_{j;\alpha_2} \widehat{\Delta H}^{(\ell)}_{jj;\alpha_1\alpha_2} \right] = O\left(\frac{1}{n}\right) . \tag{8.62}$$

We will prove this rather shortly in the next subsection in (8.71).

Assembling these leading contributions, the NTK mean recursion simplifies to

$$H^{(\ell+1)}_{\alpha_1\alpha_2} = \lambda_b^{(\ell+1)} + \lambda_W^{(\ell+1)} \langle \sigma_{\alpha_1} \sigma_{\alpha_2} \rangle_{G^{(\ell)}} + C_W^{(\ell+1)} \langle \sigma'_{\alpha_1} \sigma'_{\alpha_2} \rangle_{G^{(\ell)}} H^{(\ell)}_{\alpha_1\alpha_2} + O\left(\frac{1}{n}\right) . \tag{8.63}$$

If you're in the habit of marking up your book, feel free to draw a box around this formula.

8.3.2 NTK-Preactivation Cross Correlations

Next, let's evaluate a cross-correlation expectation of a very general form

$$\mathbb{E}\left[\mathcal{O}\left(z^{(\ell+1)} \right) \widehat{\Delta H}^{(\ell+1)}_{i_3 i_4;\alpha_3\alpha_4} \right] . \tag{8.64}$$

For instance, setting $\mathcal{O} = z^{(\ell+1)}_{i_1;\alpha_1} z^{(\ell+1)}_{i_2;\alpha_2}$ gives the elementary cross correlation (8.43), while setting $\mathcal{O} = \sigma'^{(\ell)}_{i_1;\alpha_1} \sigma'^{(\ell)}_{i_2;\alpha_2}$ gives the subleading cross correlation (8.62) that just appeared in (and then immediately disappeared from) our recursion for the NTK mean.

To begin, simply substitute the NTK forward equation (8.12) into the cross correlator (8.64), which yields

$$\begin{aligned}
&\mathbb{E}\left[\mathcal{O}\left(z^{(\ell+1)}\right) \widehat{\Delta H}^{(\ell+1)}_{i_3 i_4;\alpha_3\alpha_4}\right] = \mathrm{Cov}\left[\mathcal{O}\left(z^{(\ell+1)}\right),\, \widehat{H}^{(\ell+1)}_{i_3 i_4;\alpha_3\alpha_4}\right] \qquad (8.65)\\
=&\delta_{i_3 i_4}\, \lambda_W^{(\ell+1)} \frac{1}{n_\ell} \sum_{j=1}^{n_\ell} \left\{ \mathbb{E}\left[\mathcal{O}\left(z^{(\ell+1)}\right) \sigma^{(\ell)}_{j;\alpha_3} \sigma^{(\ell)}_{j;\alpha_4}\right] - \mathbb{E}\left[\mathcal{O}\left(z^{(\ell+1)}\right)\right] \mathbb{E}\left[\sigma^{(\ell)}_{j;\alpha_3} \sigma^{(\ell)}_{j;\alpha_4}\right] \right\}\\
&+ \sum_{j_3,j_4=1}^{n_\ell} \Big\{ \mathbb{E}\left[\mathcal{O}\left(z^{(\ell+1)}\right) W^{(\ell+1)}_{i_3 j_3} W^{(\ell+1)}_{i_4 j_4} \sigma'^{(\ell)}_{j_3;\alpha_3} \sigma'^{(\ell)}_{j_4;\alpha_4} \widehat{H}^{(\ell)}_{j_3 j_4;\alpha_3\alpha_4}\right]\\
&\qquad - \mathbb{E}\left[\mathcal{O}\left(z^{(\ell+1)}\right)\right] \mathbb{E}\left[W^{(\ell+1)}_{i_3 j_3} W^{(\ell+1)}_{i_4 j_4}\right] \mathbb{E}\left[\sigma'^{(\ell)}_{j_3;\alpha_3} \sigma'^{(\ell)}_{j_4;\alpha_4} \widehat{H}^{(\ell)}_{j_3 j_4;\alpha_3\alpha_4}\right] \Big\}.
\end{aligned}$$

Now putting the freshly-derived interlayer formulae (8.54) and (8.55) to use, this cross correlator becomes

$$\begin{aligned}
&\mathbb{E}\left[\mathcal{O}\left(z^{(\ell+1)}\right) \widehat{\Delta H}^{(\ell+1)}_{i_3 i_4;\alpha_3\alpha_4}\right] \qquad (8.66)\\
=&\delta_{i_3 i_4} \frac{\lambda_W^{(\ell+1)}}{C_W^{(\ell+1)}} \mathbb{E}\left[\left\langle\!\left\langle \mathcal{O}\left(z^{(\ell+1)}\right)\right\rangle\!\right\rangle_{\widehat{G}^{(\ell+1)}} \widehat{\Delta G}^{(\ell+1)}_{\alpha_3\alpha_4}\right]\\
&+ \delta_{i_3 i_4} \frac{C_W^{(\ell+1)}}{n_\ell} \sum_{j=1}^{n_\ell} \left\{ \mathbb{E}\left[\left\langle\!\left\langle \mathcal{O}\left(z^{(\ell+1)}\right)\right\rangle\!\right\rangle_{\widehat{G}^{(\ell+1)}} \sigma'^{(\ell)}_{j;\alpha_3} \sigma'^{(\ell)}_{j;\alpha_4} \widehat{H}^{(\ell)}_{jj;\alpha_3\alpha_4}\right] \right.\\
&\qquad \left. - \mathbb{E}\left[\left\langle\!\left\langle \mathcal{O}\left(z^{(\ell+1)}\right)\right\rangle\!\right\rangle_{\widehat{G}^{(\ell+1)}}\right] \mathbb{E}\left[\sigma'^{(\ell)}_{j;\alpha_3} \sigma'^{(\ell)}_{j;\alpha_4} \widehat{H}^{(\ell)}_{jj;\alpha_3\alpha_4}\right] \right\}\\
&+ \left(\frac{C_W^{(\ell+1)}}{n_\ell}\right)^2 \sum_{j_3,j_4=1}^{n_\ell} \sum_{\beta_3,\beta_4,\gamma_3,\gamma_4} \mathbb{E}\Big[\Big\langle\!\Big\langle \left(z^{(\ell+1)}_{i_3;\beta_3} z^{(\ell+1)}_{i_4;\beta_4} - \delta_{i_3 i_4} \widehat{G}^{(\ell+1)}_{\beta_3\beta_4}\right) \mathcal{O}\left(z^{(\ell+1)}\right)\Big\rangle\!\Big\rangle_{\widehat{G}^{(\ell+1)}}\\
&\qquad\qquad \times \widehat{G}^{\beta_3\gamma_3}_{(\ell+1)} \widehat{G}^{\beta_4\gamma_4}_{(\ell+1)} \sigma^{(\ell)}_{j_3;\gamma_3} \sigma^{(\ell)}_{j_4;\gamma_4} \sigma'^{(\ell)}_{j_3;\alpha_3} \sigma'^{(\ell)}_{j_4;\alpha_4} \widehat{H}^{(\ell)}_{j_3 j_4;\alpha_3\alpha_4} \Big].
\end{aligned}$$

Here, for the first term, we also recalled the definition of the metric fluctuation (4.74). From this general expression, we can already learn two important lessons.

First, setting $\mathcal{O} = z^{(\ell+1)}_{i_1;\alpha_1} z^{(\ell+1)}_{i_2;\alpha_2}$, we get an expression for the elementary $(\ell+1)$-th-layer

cross correlation in terms of ℓ-th-layer variables

$$
\begin{aligned}
&\mathbb{E}\left[z^{(\ell+1)}_{i_1;\alpha_1} z^{(\ell+1)}_{i_2;\alpha_2} \widehat{\Delta H}^{(\ell+1)}_{i_3 i_4;\alpha_3\alpha_4}\right] \\
=&\delta_{i_1 i_2}\delta_{i_3 i_4}\left\{\frac{\lambda_W^{(\ell+1)}}{C_W^{(\ell+1)}}\mathbb{E}\left[\widehat{\Delta G}^{(\ell+1)}_{\alpha_1\alpha_2}\widehat{\Delta G}^{(\ell+1)}_{\alpha_3\alpha_4}\right] + C_W^{(\ell+1)}\frac{1}{n_\ell}\sum_{j=1}^{n_\ell}\mathbb{E}\left[\widehat{\Delta G}^{(\ell+1)}_{\alpha_1\alpha_2}\sigma'^{(\ell)}_{j;\alpha_3}\sigma'^{(\ell)}_{j;\alpha_4}\widehat{H}^{(\ell)}_{jj;\alpha_3\alpha_4}\right]\right\} \\
&+\delta_{i_1 i_3}\delta_{i_2 i_4}\left(\frac{C_W^{(\ell+1)}}{n_\ell}\right)^2\sum_{j,k=1}^{n_\ell}\mathbb{E}\left[\sigma^{(\ell)}_{j;\alpha_1}\sigma^{(\ell)}_{k;\alpha_2}\sigma'^{(\ell)}_{j;\alpha_3}\sigma'^{(\ell)}_{k;\alpha_4}\widehat{H}^{(\ell)}_{jk;\alpha_3\alpha_4}\right] \\
&+\delta_{i_1 i_4}\delta_{i_2 i_3}\left(\frac{C_W^{(\ell+1)}}{n_\ell}\right)^2\sum_{j,k=1}^{n_\ell}\mathbb{E}\left[\sigma^{(\ell)}_{j;\alpha_2}\sigma^{(\ell)}_{k;\alpha_1}\sigma'^{(\ell)}_{j;\alpha_3}\sigma'^{(\ell)}_{k;\alpha_4}\widehat{H}^{(\ell)}_{jk;\alpha_3\alpha_4}\right] \\
\equiv&\frac{1}{n_\ell}\left[\delta_{i_1 i_2}\delta_{i_3 i_4}D^{(\ell+1)}_{\alpha_1\alpha_2\alpha_3\alpha_4} + \delta_{i_1 i_3}\delta_{i_2 i_4}F^{(\ell+1)}_{\alpha_1\alpha_3\alpha_2\alpha_4} + \delta_{i_1 i_4}\delta_{i_2 i_3}F^{(\ell+1)}_{\alpha_1\alpha_4\alpha_2\alpha_3}\right], \qquad (8.67)
\end{aligned}
$$

where on the final line we decomposed the cross correlation into two tensors with sample indices only, just as we did for the second layer in (8.37). Equating the first expression with the second, we see that these tensors are defined by the following ℓ-th-layer expectations:

$$
\begin{aligned}
\frac{1}{n_\ell}D^{(\ell+1)}_{\alpha_1\alpha_2\alpha_3\alpha_4} \equiv&\frac{\lambda_W^{(\ell+1)}}{C_W^{(\ell+1)}}\mathbb{E}\left[\widehat{\Delta G}^{(\ell+1)}_{\alpha_1\alpha_2}\widehat{\Delta G}^{(\ell+1)}_{\alpha_3\alpha_4}\right] \qquad (8.68)\\
&+C_W^{(\ell+1)}\left(\frac{1}{n_\ell}\sum_{j=1}^{n_\ell}\mathbb{E}\left[\widehat{\Delta G}^{(\ell+1)}_{\alpha_1\alpha_2}\sigma'^{(\ell)}_{j;\alpha_3}\sigma'^{(\ell)}_{j;\alpha_4}\widehat{H}^{(\ell)}_{jj;\alpha_3\alpha_4}\right]\right), \\
\frac{1}{n_\ell}F^{(\ell+1)}_{\alpha_1\alpha_3\alpha_2\alpha_4} \equiv&\left(C_W^{(\ell+1)}\right)^2\left(\frac{1}{n_\ell^2}\sum_{j,k=1}^{n_\ell}\mathbb{E}\left[\sigma^{(\ell)}_{j;\alpha_1}\sigma^{(\ell)}_{k;\alpha_2}\sigma'^{(\ell)}_{j;\alpha_3}\sigma'^{(\ell)}_{k;\alpha_4}\widehat{H}^{(\ell)}_{jk;\alpha_3\alpha_4}\right]\right). \qquad (8.69)
\end{aligned}
$$

We'll come back to evaluate these last two expressions – and thereby derive recursions for both cross correlation tensors – after we reveal the second lesson.

Second, we can start again from the general cross correlator (8.66) and push our calculation a little bit further to leading order in $1/n$. The key step is perturbatively expanding the Gaussian expectation – just as we expanded the stochastic Gaussian distribution (4.56) before using the Schwinger-Dyson equation (4.55) – to get

$$
\begin{aligned}
&\left\langle\!\left\langle \mathcal{O}\left(z^{(\ell+1)}\right)\right\rangle\!\right\rangle_{\widehat{G}^{(\ell+1)}} \qquad (8.70)\\
=&\left\langle\!\left\langle \mathcal{O}\left(z^{(\ell+1)}\right)\right\rangle\!\right\rangle_{G^{(\ell+1)}} \\
&+\frac{1}{2}\sum_{\beta_1,\beta_2,\gamma_1,\gamma_2}\left\langle\!\left\langle \sum_m\left(z^{(\ell+1)}_{m;\beta_1}z^{(\ell+1)}_{m;\beta_2} - G^{(\ell+1)}_{\beta_1\beta_2}\right)\mathcal{O}\left(z^{(\ell+1)}\right)\right\rangle\!\right\rangle_{G^{(\ell+1)}} G^{\beta_1\gamma_1}_{(\ell+1)}G^{\beta_2\gamma_2}_{(\ell+1)}\widehat{\Delta G}^{(\ell+1)}_{\gamma_1\gamma_2} + O\left(\Delta^2\right).
\end{aligned}
$$

Plugging this back into (8.66), picking up the (leading-order) pieces, and using the

definitions (8.68) and (8.69), we get

$$\begin{aligned}
&\mathbb{E}\left[\mathcal{O}\left(z^{(\ell)}\right)\widehat{\Delta H}^{(\ell)}_{i_3 i_4;\alpha_3\alpha_4}\right] \qquad (8.71)\\
=&\delta_{i_3 i_4}\frac{1}{n_{\ell-1}}\left[\frac{1}{2}\sum_{\beta_1,\beta_2,\gamma_1,\gamma_2}\left\langle\!\!\left\langle\sum_{m=1}^{n_\ell}\left(z^{(\ell)}_{m;\beta_1}z^{(\ell)}_{m;\beta_2}-G^{(\ell)}_{\beta_1\beta_2}\right)\mathcal{O}\left(z^{(\ell)}\right)\right\rangle\!\!\right\rangle_{G^{(\ell)}} G^{\beta_1\gamma_1}_{(\ell)}G^{\beta_2\gamma_2}_{(\ell)}\right]D^{(\ell)}_{\gamma_1\gamma_2\alpha_3\alpha_4}\\
&+\frac{1}{n_{\ell-1}}\sum_{\beta_1,\beta_2,\gamma_1,\gamma_2}\left\langle\!\!\left\langle\left(z^{(\ell)}_{i_3;\beta_1}z^{(\ell)}_{i_4;\beta_2}-\delta_{i_3 i_4}G^{(\ell)}_{\beta_1\beta_2}\right)\mathcal{O}\left(z^{(\ell)}\right)\right\rangle\!\!\right\rangle_{G^{(\ell)}} G^{\beta_1\gamma_1}_{(\ell)}G^{\beta_2\gamma_2}_{(\ell)}F^{(\ell)}_{\gamma_1\alpha_3\gamma_2\alpha_4}\\
&+O\left(\frac{1}{n^2}\right)\,,
\end{aligned}$$

where we have also relabeled *layer indices* as $(\ell+1)\to\ell$ everywhere for the ease of later substitutions.[9] This result illustrates that these more general cross correlations are governed by the same tensors, $D^{(\ell)}$ and $F^{(\ell)}$, as the elementary cross correlation (8.67). We can indeed compute all the cross correlators if we find and solve recursions for $D^{(\ell)}$ and $F^{(\ell)}$. It is this task that we turn to next.

D-recursion

Starting from our expression for $D^{(\ell+1)}$ (8.68) and substituting in the definition of the stochastic metric (8.49), we get

$$\begin{aligned}
D^{(\ell+1)}_{\alpha_1\alpha_2\alpha_3\alpha_4}=&C_W^{(\ell+1)}\frac{1}{n_\ell}\sum_{j,k=1}^{n_\ell}\mathrm{Cov}\left[\sigma^{(\ell)}_{j;\alpha_1}\sigma^{(\ell)}_{j;\alpha_2},\,\lambda_W^{(\ell+1)}\sigma^{(\ell)}_{k;\alpha_3}\sigma^{(\ell)}_{k;\alpha_4}+C_W^{(\ell+1)}H^{(\ell)}_{\alpha_3\alpha_4}\sigma'^{(\ell)}_{k;\alpha_3}\sigma'^{(\ell)}_{k;\alpha_4}\right]\\
&+\left(C_W^{(\ell+1)}\right)^2\frac{1}{n_\ell}\sum_{j,k=1}^{n_\ell}\mathrm{Cov}\left[\sigma^{(\ell)}_{j;\alpha_1}\sigma^{(\ell)}_{j;\alpha_2},\,\sigma'^{(\ell)}_{k;\alpha_3}\sigma'^{(\ell)}_{k;\alpha_4}\widehat{\Delta H}^{(\ell)}_{kk;\alpha_3\alpha_4}\right]+O\left(\frac{1}{n}\right)\,,
\end{aligned} \qquad (8.72)$$

where we have again decomposed the ℓ-th-layer NTK into a mean and fluctuation piece. We see that there are two types of terms here: covariances on a single neuron $j=k$ and covariances between pairs of neurons $j\neq k$.

For the single-neuron contribution with $j=k$, at the leading order, we find the same contribution that we found for the second layer (8.38)[10]

$$C_W^{(\ell+1)}\left[\left\langle\sigma_{\alpha_1}\sigma_{\alpha_2}\widehat{\Omega}^{(\ell+1)}_{\alpha_3\alpha_4}\right\rangle_{G^{(\ell)}}-\left\langle\sigma_{\alpha_1}\sigma_{\alpha_2}\right\rangle_{G^{(\ell)}}\left\langle\widehat{\Omega}^{(\ell+1)}_{\alpha_3\alpha_4}\right\rangle_{G^{(\ell)}}\right]+O\left(\frac{1}{n}\right)\,. \qquad (8.73)$$

Here, the auxiliary stochastic matrix $\widehat{\Omega}^{(\ell+1)}_{\alpha_1\alpha_2}$ is defined as

$$\widehat{\Omega}^{(\ell+1)}_{\alpha_1\alpha_2}\equiv\lambda_W^{(\ell+1)}\sigma^{(\ell)}_{\alpha_1}\sigma^{(\ell)}_{\alpha_2}+C_W^{(\ell+1)}H^{(\ell)}_{\alpha_1\alpha_2}\sigma'^{(\ell)}_{\alpha_1}\sigma'^{(\ell)}_{\alpha_2}\,, \qquad (8.74)$$

[9] You might worry about the summation $\sum_{m=1}^{n_\ell}$ inside the first Gaussian expectation in (8.71). However, due to Gaussian factorization, this expectation stays of order one so long as the observable $\mathcal{O}$ depends on only a finite number of neurons.

[10] The subleading $O(1/n)$ piece includes both a contribution from the non-Gaussian part of the distribution as well as a cross correlation contribution from the previous layer.

which simply generalizes the second-layer definition (8.34). As a reminder the unhatted matrix $H^{(\ell)}_{\alpha_1\alpha_2}$ is the NTK mean, which is not a random variable and can safely be taken outside the Gaussian expectation $\langle\cdot\rangle_{G^{(\ell)}}$. This means that, as a stochastic variable, $\widehat{\Omega}^{(\ell+1)}_{\alpha_1\alpha_2}$ depends only on the ℓ-th-layer preactivations.

Next, for the pairs-of-neurons contribution to (8.72) with $j \neq k$, the first term can be evaluated by the intralayer formula (8.41) and yields

$$\frac{n_\ell}{4n_{\ell-1}} C_W^{(\ell+1)} \sum_{\gamma_1,\gamma_2,\gamma_3,\gamma_4} V^{(\gamma_1\gamma_2)(\gamma_3\gamma_4)}_{(\ell)} \left\langle \left(z_{\gamma_1} z_{\gamma_2} - G^{(\ell)}_{\gamma_1\gamma_2}\right) \sigma_{\alpha_1}\sigma_{\alpha_2} \right\rangle_{G^{(\ell)}} \left\langle \left(z_{\gamma_3} z_{\gamma_4} - G^{(\ell)}_{\gamma_3\gamma_4}\right) \widehat{\Omega}^{(\ell+1)}_{\alpha_3\alpha_4} \right\rangle_{G^{(\ell)}} . \tag{8.75}$$

Meanwhile, the covariance in the second term can be unrolled as

$$\begin{aligned} &\mathrm{Cov}\left[\sigma^{(\ell)}_{j;\alpha_1}\sigma^{(\ell)}_{j;\alpha_2},\, \sigma'^{(\ell)}_{k;\alpha_3}\sigma'^{(\ell)}_{k;\alpha_4}\widehat{\Delta H}^{(\ell)}_{kk;\alpha_3\alpha_4}\right] \\ =&\mathbb{E}\left[\sigma^{(\ell)}_{j;\alpha_1}\sigma^{(\ell)}_{j;\alpha_2}\sigma'^{(\ell)}_{k;\alpha_3}\sigma'^{(\ell)}_{k;\alpha_4}\widehat{\Delta H}^{(\ell)}_{kk;\alpha_3\alpha_4}\right] - \mathbb{E}\left[\sigma^{(\ell)}_{j;\alpha_1}\sigma^{(\ell)}_{j;\alpha_2}\right]\mathbb{E}\left[\sigma'^{(\ell)}_{k;\alpha_3}\sigma'^{(\ell)}_{k;\alpha_4}\widehat{\Delta H}^{(\ell)}_{kk;\alpha_3\alpha_4}\right] \\ =&\frac{1}{2n_{\ell-1}} \sum_{\beta_1,\beta_2,\gamma_1,\gamma_2} \left\langle \left(z_{\beta_1} z_{\beta_2} - G^{(\ell)}_{\beta_1\beta_2}\right) \sigma_{\alpha_1}\sigma_{\alpha_2} \right\rangle_{G^{(\ell)}} \langle \sigma'_{\alpha_3}\sigma'_{\alpha_4}\rangle_{G^{(\ell)}} G^{\beta_1\gamma_1}_{(\ell)} G^{\beta_2\gamma_2}_{(\ell)} D^{(\ell)}_{\gamma_1\gamma_2\alpha_3\alpha_4} + O\left(\frac{1}{n^2}\right), \end{aligned} \tag{8.76}$$

where in the last line we used the cross-correlation formula (8.71) with the observables $\mathcal{O} = \sigma^{(\ell)}_{j;\alpha_1}\sigma^{(\ell)}_{j;\alpha_2}\sigma'^{(\ell)}_{k;\alpha_3}\sigma'^{(\ell)}_{k;\alpha_4}$ and $\mathcal{O} = \sigma'^{(\ell)}_{k;\alpha_3}\sigma'^{(\ell)}_{k;\alpha_4}$, respectively. Combining these contributions with the first piece (8.73), we get our desired recursion:

$$\begin{aligned} &D^{(\ell+1)}_{\alpha_1\alpha_2\alpha_3\alpha_4} \\ =&C_W^{(\ell+1)} \left(\left\langle \sigma_{\alpha_1}\sigma_{\alpha_2}\widehat{\Omega}^{(\ell+1)}_{\alpha_3\alpha_4} \right\rangle_{G^{(\ell)}} - \langle \sigma_{\alpha_1}\sigma_{\alpha_2}\rangle_{G^{(\ell)}} \left\langle \widehat{\Omega}^{(\ell+1)}_{\alpha_3\alpha_4} \right\rangle_{G^{(\ell)}} \right) \\ &+ \frac{n_\ell}{4n_{\ell-1}} C_W^{(\ell+1)} \sum_{\gamma_1,\gamma_2,\gamma_3,\gamma_4} V^{(\gamma_1\gamma_2)(\gamma_3\gamma_4)}_{(\ell)} \left\langle \left(z_{\gamma_1} z_{\gamma_2} - G^{(\ell)}_{\gamma_1\gamma_2}\right) \sigma_{\alpha_1}\sigma_{\alpha_2} \right\rangle_{G^{(\ell)}} \left\langle \left(z_{\gamma_3} z_{\gamma_4} - G^{(\ell)}_{\gamma_3\gamma_4}\right) \widehat{\Omega}^{(\ell+1)}_{\alpha_3\alpha_4} \right\rangle_{G^{(\ell)}} \\ &+ \frac{n_\ell}{2n_{\ell-1}} \left(C_W^{(\ell+1)}\right)^2 \sum_{\beta_1,\beta_2,\gamma_1,\gamma_2} D^{(\ell)}_{\gamma_1\gamma_2\alpha_3\alpha_4} \left\langle \left(z_{\beta_1} z_{\beta_2} - G^{(\ell)}_{\beta_1\beta_2}\right) \sigma_{\alpha_1}\sigma_{\alpha_2} \right\rangle_{G^{(\ell)}} G^{\beta_1\gamma_1}_{(\ell)} G^{\beta_2\gamma_2}_{(\ell)} \langle \sigma'_{\alpha_3}\sigma'_{\alpha_4}\rangle_{G^{(\ell)}} \\ &+ O\left(\frac{1}{n}\right) . \end{aligned} \tag{8.77}$$

Interestingly, we see that at leading order the D-type cross correlation in layer $(\ell+1)$ mixes D-type correlations from layer ℓ with the four-point vertex $V^{(\ell)}$, but does not mix with the F-type cross correlations or any part of the NTK variance.

F-recursion

Starting from our expression for $F^{(\ell+1)}$ (8.69) and decomposing the NTK into a mean and fluctuation, we get

$$F^{(\ell+1)}_{\alpha_1\alpha_3\alpha_2\alpha_4} = \left(C_W^{(\ell+1)}\right)^2 \frac{1}{n_\ell}\sum_{j=1}^{n_\ell} \mathbb{E}\left[\sigma^{(\ell)}_{j;\alpha_1}\sigma'^{(\ell)}_{j;\alpha_3}\sigma^{(\ell)}_{j;\alpha_2}\sigma'^{(\ell)}_{j;\alpha_4}\right] H^{(\ell)}_{\alpha_3\alpha_4} \tag{8.78}$$
$$+ \left(C_W^{(\ell+1)}\right)^2 \frac{1}{n_\ell}\sum_{j,k=1}^{n_\ell} \mathbb{E}\left[\sigma^{(\ell)}_{j;\alpha_1}\sigma'^{(\ell)}_{j;\alpha_3}\sigma^{(\ell)}_{k;\alpha_2}\sigma'^{(\ell)}_{k;\alpha_4}\widehat{\Delta H}^{(\ell)}_{jk;\alpha_3\alpha_4}\right] .$$

At leading order, the first term simply becomes a single-neuron Gaussian expectation; the second term can be evaluated with the cross-correlation formula (8.71), where the diagonal sum with $j = k$ is of order $O(1/n)$ and can be neglected while the off-diagonal sum with $j \neq k$ yields the term involving $F^{(\ell)}$. All together, this gives

$$F^{(\ell+1)}_{\alpha_1\alpha_3\alpha_2\alpha_4} = \left(C_W^{(\ell+1)}\right)^2 \left\langle \sigma_{\alpha_1}\sigma_{\alpha_2}\sigma'_{\alpha_3}\sigma'_{\alpha_4}\right\rangle_{G^{(\ell)}} H^{(\ell)}_{\alpha_3\alpha_4} \tag{8.79}$$
$$+ \frac{n_\ell}{n_{\ell-1}}\left(C_W^{(\ell+1)}\right)^2 \sum_{\beta_1,\beta_2,\gamma_1,\gamma_2} \left\langle \sigma_{\alpha_1}\sigma'_{\alpha_3}z_{\beta_1}\right\rangle_{G^{(\ell)}} \left\langle \sigma_{\alpha_2}\sigma'_{\alpha_4}z_{\beta_2}\right\rangle_{G^{(\ell)}} G^{\beta_1\gamma_1}_{(\ell)} G^{\beta_2\gamma_2}_{(\ell)} F^{(\ell)}_{\gamma_1\alpha_3\gamma_2\alpha_4}$$
$$+ O\left(\frac{1}{n}\right) .$$

As with the D-recursion before, the first term was present in the second-layer $F^{(2)}$ (8.39), while the second term is a direct consequence of having a fluctuating NTK in the previous layer ℓ. Additionally, we see that at leading order the F-type cross correlation doesn't mix at all with any of our other finite-width tensors.

Finally, before moving on to discuss the NTK variance, let us note that the recursions for both $D^{(\ell)}$ and $F^{(\ell)}$ – combined with the initial condition $D^{(1)} = F^{(1)} = 0$ from the first layer where the NTK is deterministic – ensure that they each stay of order one. Given the factor of $1/n_\ell$ in the decomposition of the cross correlation into these tensors (8.67) and our "second lesson" encapsulated by the cross-correlation formula (8.71), this means that any and all cross correlations are suppressed in the $1/n$ expansion and vanish identically in the strict infinite-width limit.

8.3.3 NTK Variance

Now let's finally *slay the beast* that is the NTK variance. Similar to the NTK-preactivation cross correlation, the NTK-variance calculation in deeper layers differs from the second-layer calculation due to nontrivial intralayer correlations (8.41) in the previous layer and due to the fluctuating NTK (8.58).

The NTK variance is given by the expected magnitude of the NTK fluctuation

$$\mathbb{E}\left[\widehat{\Delta H}^{(\ell+1)}_{i_1i_2;\alpha_1\alpha_2}\widehat{\Delta H}^{(\ell+1)}_{i_3i_4;\alpha_3\alpha_4}\right] = \mathrm{Cov}\left[\widehat{H}^{(\ell+1)}_{i_1i_2;\alpha_1\alpha_2}, \widehat{H}^{(\ell+1)}_{i_3i_4;\alpha_3\alpha_4}\right] . \tag{8.80}$$

To begin our calculation, let us plug the NTK forward equation (8.12) into this defining expression and then integrate out the weights $W^{(\ell+1)}$, which is easy since they are independent random variables. Although there are many terms, the algebra is mostly straightforward:

$$\begin{aligned}
&\mathbb{E}\left[\widehat{\Delta H}^{(\ell+1)}_{i_1 i_2;\alpha_1\alpha_2}\widehat{\Delta H}^{(\ell+1)}_{i_3 i_4;\alpha_3\alpha_4}\right] \qquad (8.81)\\
=&\delta_{i_1 i_2}\delta_{i_3 i_4}\frac{1}{n_\ell^2}\sum_{j,k=1}^{n_\ell}\left\{\left(\lambda_W^{(\ell+1)}\right)^2\mathrm{Cov}\left[\sigma^{(\ell)}_{j;\alpha_1}\sigma^{(\ell)}_{j;\alpha_2},\,\sigma^{(\ell)}_{k;\alpha_3}\sigma^{(\ell)}_{k;\alpha_4}\right]\right.\\
&\qquad+\left(\lambda_W^{(\ell+1)}\right)C_W^{(\ell+1)}\mathrm{Cov}\left[\sigma^{(\ell)}_{j;\alpha}\sigma^{(\ell)}_{j;\alpha_2},\,\sigma'^{(\ell)}_{k;\alpha_3}\sigma'^{(\ell)}_{k;\alpha_4}\widehat{H}^{(\ell)}_{kk;\alpha_3\alpha_4}\right]\\
&\qquad+\left(\lambda_W^{(\ell+1)}\right)C_W^{(\ell+1)}\mathrm{Cov}\left[\sigma'^{(\ell)}_{j;\alpha}\sigma'^{(\ell)}_{j;\alpha_2}\widehat{H}^{(\ell)}_{jj;\alpha\alpha_2},\,\sigma^{(\ell)}_{k;\alpha_3}\sigma^{(\ell)}_{k;\alpha_4}\right]\\
&\qquad\left.+\left(C_W^{(\ell+1)}\right)^2\mathrm{Cov}\left[\sigma'^{(\ell)}_{j;\alpha_1}\sigma'^{(\ell)}_{j;\alpha_2}\widehat{H}^{(\ell)}_{jj;\alpha_1\alpha_2},\,\sigma'^{(\ell)}_{k;\alpha_3}\sigma'^{(\ell)}_{k;\alpha_4}\widehat{H}^{(\ell)}_{kk;\alpha_3\alpha_4}\right]\right\}\\
&+\delta_{i_1 i_3}\delta_{i_2 i_4}\left(C_W^{(\ell+1)}\right)^2\frac{1}{n_\ell^2}\sum_{j,k=1}^{n_\ell}\mathbb{E}\left[\sigma'^{(\ell)}_{j;\alpha_1}\sigma'^{(\ell)}_{k;\alpha_2}\widehat{H}^{(\ell)}_{jk;\alpha_1\alpha_2}\sigma'^{(\ell)}_{j;\alpha_3}\sigma'^{(\ell)}_{k;\alpha_4}\widehat{H}^{(\ell)}_{jk;\alpha_3\alpha_4}\right]\\
&+\delta_{i_1 i_4}\delta_{i_2 i_3}\left(C_W^{(\ell+1)}\right)^2\frac{1}{n_\ell^2}\sum_{j,k=1}^{n_\ell}\mathbb{E}\left[\sigma'^{(\ell)}_{j;\alpha_1}\sigma'^{(\ell)}_{k;\alpha_2}\widehat{H}^{(\ell)}_{jk;\alpha_1\alpha_2}\sigma'^{(\ell)}_{k;\alpha_3}\sigma'^{(\ell)}_{j;\alpha_4}\widehat{H}^{(\ell)}_{kj;\alpha_3\alpha_4}\right].
\end{aligned}$$

In obtaining these last three terms, you should have made three distinct pairings for the two pairs of Wick contractions of the four $W^{(\ell+1)}$'s, one paring within the same NTK and two pairings across the NTKs.

An inspection of the pattern of neural indices in the Kronecker deltas from (8.81) suggests that we should again decompose the NTK variance into two tensors as

$$\begin{aligned}
&\mathbb{E}\left[\widehat{\Delta H}^{(\ell)}_{i_1 i_2;\alpha_1\alpha_2}\widehat{\Delta H}^{(\ell)}_{i_3 i_4;\alpha_3\alpha_4}\right] \qquad (8.82)\\
\equiv&\frac{1}{n_{\ell-1}}\left[\delta_{i_1 i_2}\delta_{i_3 i_4}A^{(\ell)}_{(\alpha_1\alpha_2)(\alpha_3\alpha_4)}+\delta_{i_1 i_3}\delta_{i_2 i_4}B^{(\ell)}_{\alpha_1\alpha_3\alpha_2\alpha_4}+\delta_{i_1 i_4}\delta_{i_2 i_3}B^{(\ell)}_{\alpha_1\alpha_4\alpha_2\alpha_3}\right],
\end{aligned}$$

just as we did for the second layer before in (8.31). Here, a factor of $1/n_{\ell-1}$ was pulled out in anticipation that the overall variance will be $O(1/n)$ just as it was for the second layer. For now you can think of this parameterization as an ansatz; we will soon recursively show that $A^{(\ell)}_{(\alpha_1\alpha_2)(\alpha_3\alpha_4)}$ and $B^{(\ell)}_{\alpha_1\alpha_3\alpha_2\alpha_4}$ stay of order one as the network width increases.

Now, let's work out the layer recursions for $A^{(\ell)}$ and $B^{(\ell)}$.

B-recursion

We'll start with B-recursion because it's simpler. Considering (8.81) with the decomposition (8.82) in mind, we see that $B^{(\ell+1)}$ is given by the following ℓ-th-layer expectation:

$$B^{(\ell+1)}_{\alpha_1\alpha_3\alpha_2\alpha_4}=\left(C_W^{(\ell+1)}\right)^2\frac{1}{n_\ell}\sum_{j,k=1}^{n_\ell}\mathbb{E}\left[\sigma'^{(\ell)}_{j;\alpha_1}\sigma'^{(\ell)}_{k;\alpha_2}\widehat{H}^{(\ell)}_{jk;\alpha_1\alpha_2}\sigma'^{(\ell)}_{j;\alpha_3}\sigma'^{(\ell)}_{k;\alpha_4}\widehat{H}^{(\ell)}_{jk;\alpha_3\alpha_4}\right]. \qquad (8.83)$$

As should now be familiar, the double summation in (8.83) splits into two types of terms, diagonal ones with $j = k$ and off-diagonal ones with $j \neq k$.

For the diagonal part, the leading contribution is from the NTK mean

$$\begin{aligned}&\left(C_W^{(\ell+1)}\right)^2 \frac{1}{n_\ell} \sum_{j=1}^{n_\ell} \mathbb{E}\left[\sigma'^{(\ell)}_{j;\alpha_1}\sigma'^{(\ell)}_{j;\alpha_2}\sigma'^{(\ell)}_{j;\alpha_3}\sigma'^{(\ell)}_{j;\alpha_4}\right] H^{(\ell)}_{\alpha_1\alpha_2} H^{(\ell)}_{\alpha_3\alpha_4} + O\left(\frac{1}{n}\right) \\ &= \left(C_W^{(\ell+1)}\right)^2 \left\langle \sigma'_{\alpha_1}\sigma'_{\alpha_2}\sigma'_{\alpha_3}\sigma'_{\alpha_4}\right\rangle_{G^{(\ell)}} H^{(\ell)}_{\alpha_1\alpha_2} H^{(\ell)}_{\alpha_3\alpha_4} + O\left(\frac{1}{n}\right) ,\end{aligned} \tag{8.84}$$

which is analogous to what we found in the second layer (8.33).[11]

For the off-diagonal part of (8.83), the NTK mean vanishes and the leading contribution is from the NTK fluctuation

$$\left(C_W^{(\ell+1)}\right)^2 \frac{1}{n_\ell} \sum_{\substack{j,k=1 \\ j\neq k}}^{n_\ell} \mathbb{E}\left[\sigma'^{(\ell)}_{j;\alpha_1}\sigma'^{(\ell)}_{k;\alpha_2}\sigma'^{(\ell)}_{j;\alpha_3}\sigma'^{(\ell)}_{k;\alpha_4}\widehat{\Delta H}^{(\ell)}_{jk;\alpha_1\alpha_2}\widehat{\Delta H}^{(\ell)}_{jk;\alpha_3\alpha_4}\right] . \tag{8.85}$$

The expectation already is $O(\Delta^2)$ from the two NTK fluctuations inside it and thus, neglecting higher-order correlations of order $O(\Delta^3)$, we have

$$\begin{aligned}&\mathbb{E}\left[\sigma'^{(\ell)}_{j;\alpha_1}\sigma'^{(\ell)}_{k;\alpha_2}\sigma'^{(\ell)}_{j;\alpha_3}\sigma'^{(\ell)}_{k;\alpha_4}\widehat{\Delta H}^{(\ell)}_{jk;\alpha_1\alpha_2}\widehat{\Delta H}^{(\ell)}_{jk;\alpha_3\alpha_4}\right] \\ =&\mathbb{E}\left[\sigma'^{(\ell)}_{j;\alpha_1}\sigma'^{(\ell)}_{k;\alpha_2}\sigma'^{(\ell)}_{j;\alpha_3}\sigma'^{(\ell)}_{k;\alpha_4}\right]\mathbb{E}\left[\widehat{\Delta H}^{(\ell)}_{jk;\alpha_1\alpha_2}\widehat{\Delta H}^{(\ell)}_{jk;\alpha_3\alpha_4}\right] + O\left(\frac{1}{n^2}\right) ,\end{aligned} \tag{8.86}$$

where the detailed explanation for such a factorization is given in this footnote.[12] Then, using the decomposition (8.82) for the NTK variance and similar logic as (4.63) to

[11] Again, the subleading $O(1/n)$ piece includes both a contribution from the non-Gaussian distribution as well as a cross correlation contribution from the previous layer.

[12] In greater detail, you can think of what we are doing here as separating $\sigma'^{(\ell)}_{j;\alpha_1}\sigma'^{(\ell)}_{k;\alpha_2}\sigma'^{(\ell)}_{j;\alpha_3}\sigma'^{(\ell)}_{k;\alpha_4}$ into a mean $\mathbb{E}\left[\sigma'^{(\ell)}_{j;\alpha_1}\sigma'^{(\ell)}_{k;\alpha_2}\sigma'^{(\ell)}_{j;\alpha_3}\sigma'^{(\ell)}_{k;\alpha_4}\right]$ and fluctuation and – since the expectation already contains two fluctuations $\widehat{\Delta H}^{(\ell)}_{jk;\alpha_1\alpha_2}\widehat{\Delta H}^{(\ell)}_{jk;\alpha_3\alpha_4}$ – the latter fluctuating piece contributes $O\left(\Delta^3\right)$ and thus can be neglected.

In alternate detail, we can view this expectation (8.86) as a correlator of three random variables $\sigma'^{(\ell)}_{j;\alpha_1}\sigma'^{(\ell)}_{k;\alpha_2}\sigma'^{(\ell)}_{j;\alpha_3}\sigma'^{(\ell)}_{k;\alpha_4}$, $\widehat{\Delta H}^{(\ell)}_{jk;\alpha_1\alpha_2}$, and $\widehat{\Delta H}^{(\ell)}_{jk;\alpha_3\alpha_4}$, and decompose it into one-point and two-point correlators as

$$\begin{aligned}&\mathbb{E}\left[\sigma'^{(\ell)}_{j;\alpha_1}\sigma'^{(\ell)}_{k;\alpha_2}\sigma'^{(\ell)}_{j;\alpha_3}\sigma'^{(\ell)}_{k;\alpha_4}\widehat{\Delta H}^{(\ell)}_{jk;\alpha_1\alpha_2}\widehat{\Delta H}^{(\ell)}_{jk;\alpha_3\alpha_4}\right] \\ =&\mathbb{E}\left[\sigma'^{(\ell)}_{j;\alpha_1}\sigma'^{(\ell)}_{k;\alpha_2}\sigma'^{(\ell)}_{j;\alpha_3}\sigma'^{(\ell)}_{k;\alpha_4}\right]\mathbb{E}\left[\widehat{\Delta H}^{(\ell)}_{jk;\alpha_1\alpha_2}\right]\mathbb{E}\left[\widehat{\Delta H}^{(\ell)}_{jk;\alpha_3\alpha_4}\right] \\ &+\mathbb{E}\left[\sigma'^{(\ell)}_{j;\alpha_1}\sigma'^{(\ell)}_{k;\alpha_2}\sigma'^{(\ell)}_{j;\alpha_3}\sigma'^{(\ell)}_{k;\alpha_4}\right]\mathbb{E}\left[\widehat{\Delta H}^{(\ell)}_{jk;\alpha_1\alpha_2}\widehat{\Delta H}^{(\ell)}_{jk;\alpha_3\alpha_4}\right] \\ &+\mathbb{E}\left[\sigma'^{(\ell)}_{j;\alpha_1}\sigma'^{(\ell)}_{k;\alpha_2}\sigma'^{(\ell)}_{j;\alpha_3}\sigma'^{(\ell)}_{k;\alpha_4}\widehat{\Delta H}^{(\ell)}_{jk;\alpha_1\alpha_2}\right]\mathbb{E}\left[\widehat{\Delta H}^{(\ell)}_{jk;\alpha_3\alpha_4}\right] \\ &+\mathbb{E}\left[\sigma'^{(\ell)}_{j;\alpha_1}\sigma'^{(\ell)}_{k;\alpha_2}\sigma'^{(\ell)}_{j;\alpha_3}\sigma'^{(\ell)}_{k;\alpha_4}\widehat{\Delta H}^{(\ell)}_{jk;\alpha_3\alpha_4}\right]\mathbb{E}\left[\widehat{\Delta H}^{(\ell)}_{jk;\alpha_1\alpha_2}\right] + O\left(\frac{1}{n^2}\right) ,\end{aligned} \tag{8.87}$$

where the $O\left(1/n^2\right)$ part contains the connected piece of the decomposition. Since the NTK fluctuation has mean zero, only the second term survives at this order.

evaluate the four-point correlator of off-diagonal activations, we get

$$\left(C_W^{(\ell+1)}\right)^2 \frac{1}{n_\ell} \sum_{\substack{j,k=1 \\ j\neq k}}^{n_\ell} \mathbb{E}\left[\sigma'^{(\ell)}_{j;\alpha_1}\sigma'^{(\ell)}_{k;\alpha_2}\sigma'^{(\ell)}_{j;\alpha_3}\sigma'^{(\ell)}_{k;\alpha_4}\right] \mathbb{E}\left[\widehat{\Delta H}^{(\ell)}_{jk;\alpha_1\alpha_2}\widehat{\Delta H}^{(\ell)}_{jk;\alpha_3\alpha_4}\right] + O\left(\frac{1}{n}\right) \tag{8.88}$$

$$= \left(C_W^{(\ell+1)}\right)^2 \left\langle \sigma'_{\alpha_1}\sigma'_{\alpha_3}\right\rangle_{G^{(\ell)}} \left\langle \sigma'_{\alpha_2}\sigma'_{\alpha_4}\right\rangle_{G^{(\ell)}} \frac{n_\ell}{n_{\ell-1}} B^{(\ell)}_{\alpha_1\alpha_3\alpha_2\alpha_4} + O\left(\frac{1}{n}\right) .$$

Substituting both the diagonal contribution (8.84) and off-diagonal contribution (8.88) back into (8.83), we get the B-recursion:

$$\begin{aligned} B^{(\ell+1)}_{\alpha_1\alpha_3\alpha_2\alpha_4} = \left(C_W^{(\ell+1)}\right)^2 \Bigg[& \left\langle \sigma'_{\alpha_1}\sigma'_{\alpha_2}\sigma'_{\alpha_3}\sigma'_{\alpha_4}\right\rangle_{G^{(\ell)}} H^{(\ell)}_{\alpha_1\alpha_2} H^{(\ell)}_{\alpha_3\alpha_4} \\ & + \left(\frac{n_\ell}{n_{\ell-1}}\right) \left\langle \sigma'_{\alpha_1}\sigma'_{\alpha_3}\right\rangle_{G^{(\ell)}} \left\langle \sigma'_{\alpha_2}\sigma'_{\alpha_4}\right\rangle_{G^{(\ell)}} B^{(\ell)}_{\alpha_1\alpha_3\alpha_2\alpha_4} \Bigg] + O\left(\frac{1}{n}\right) . \end{aligned} \tag{8.89}$$

As promised, we recursively see that $B^{(\ell)}$ is an order-one quantity. Additionally, we note that at leading order this B-type NTK variance doesn't mix with any other finite-width tensors.

A-recursion

Let us now determine the A-recursion. Again equating our expression for the NTK variance (8.81) with the A/B-decomposition (8.82), we see that $A^{(\ell+1)}$ is given by the following ℓ-th-layer covariances:

$$\begin{aligned} A^{(\ell+1)}_{(\alpha_1\alpha_2)(\alpha_3\alpha_4)} = & \frac{1}{n_\ell} \sum_{j,k=1}^{n_\ell} \Bigg\{ \left(\lambda_W^{(\ell+1)}\right)^2 \mathrm{Cov}\left[\sigma^{(\ell)}_{j;\alpha_1}\sigma^{(\ell)}_{j;\alpha_2}, \sigma^{(\ell)}_{k;\alpha_3}\sigma^{(\ell)}_{k;\alpha_4}\right] \\ & + \lambda_W^{(\ell+1)} C_W^{(\ell+1)} \mathrm{Cov}\left[\sigma^{(\ell)}_{j;\alpha}\sigma^{(\ell)}_{j;\alpha_2}, \sigma'^{(\ell)}_{k;\alpha_3}\sigma'^{(\ell)}_{k;\alpha_4}\widehat{H}^{(\ell)}_{kk;\alpha_3\alpha_4}\right] \\ & + \lambda_W^{(\ell+1)} C_W^{(\ell+1)} \mathrm{Cov}\left[\sigma'^{(\ell)}_{j;\alpha}\sigma'^{(\ell)}_{j;\alpha_2}\widehat{H}^{(\ell)}_{jj;\alpha\alpha_2}, \sigma^{(\ell)}_{k;\alpha_3}\sigma^{(\ell)}_{k;\alpha_4}\right] \\ & + \left(C_W^{(\ell+1)}\right)^2 \mathrm{Cov}\left[\sigma'^{(\ell)}_{j;\alpha_1}\sigma'^{(\ell)}_{j;\alpha_2}\widehat{H}^{(\ell)}_{jj;\alpha_1\alpha_2}, \sigma'^{(\ell)}_{k;\alpha_3}\sigma'^{(\ell)}_{k;\alpha_4}\widehat{H}^{(\ell)}_{kk;\alpha_3\alpha_4}\right] \Bigg\} . \end{aligned} \tag{8.90}$$

As we've now seen many times previously, our approach will be to divide up the double summation in (8.90) into two types of terms, diagonal terms on a single neuron with $j = k$ and off-diagonal terms on pairs of neurons with $j \neq k$.

As was the case for the B-recursion, the leading contribution from the diagonal part with $j = k$ comes from the NTK mean and matches what we found for the second layer (8.32)

$$\left\langle \widehat{\Omega}^{(\ell+1)}_{\alpha_1\alpha_2}\widehat{\Omega}^{(\ell+1)}_{\alpha_3\alpha_4}\right\rangle_{G^{(\ell)}} - \left\langle \widehat{\Omega}^{(\ell+1)}_{\alpha_1\alpha_2}\right\rangle_{G^{(\ell)}} \left\langle \widehat{\Omega}^{(\ell+1)}_{\alpha_3\alpha_4}\right\rangle_{G^{(\ell)}} + O\left(\frac{1}{n}\right) , \tag{8.91}$$

where the definition of the auxiliary stochastic tensor $\widehat{\Omega}^{(\ell+1)}_{\alpha_1\alpha_2}$ was given in (8.74).[13]

This leaves us with the off-diagonal part of (8.90) with $j \neq k$. Here, there will be leading contributions both from the NTK mean and from the NTK fluctuations. The contributions from the mean are given by replacing $\widehat{H}^{(\ell)}_{i_1 i_2;\alpha_1\alpha_2} \to \delta_{i_1 i_2} H^{(\ell)}_{\alpha_1\alpha_2}$:

$$\begin{aligned}\frac{1}{n_\ell}\sum_{\substack{j,k=1\\ j\neq k}}^{n_\ell}\Bigg\{ &\left(\lambda_W^{(\ell+1)}\right)^2 \mathrm{Cov}\Big[\sigma^{(\ell)}_{j;\alpha_1}\sigma^{(\ell)}_{j;\alpha_2},\, \sigma^{(\ell)}_{k;\alpha_3}\sigma^{(\ell)}_{k;\alpha_4}\Big] \\ &+ \lambda_W^{(\ell+1)} C_W^{(\ell+1)} H^{(\ell)}_{\alpha_3\alpha_4}\mathrm{Cov}\Big[\sigma^{(\ell)}_{j;\alpha}\sigma^{(\ell)}_{j;\alpha_2},\, \sigma'^{(\ell)}_{k;\alpha_3}\sigma'^{(\ell)}_{k;\alpha_4}\Big] \\ &+ \lambda_W^{(\ell+1)} C_W^{(\ell+1)} H^{(\ell)}_{\alpha_1\alpha_2}\mathrm{Cov}\Big[\sigma'^{(\ell)}_{j;\alpha}\sigma'^{(\ell)}_{j;\alpha_2},\, \sigma^{(\ell)}_{k;\alpha_3}\sigma^{(\ell)}_{k;\alpha_4}\Big] \\ &+ \left(C_W^{(\ell+1)}\right)^2 H^{(\ell)}_{\alpha_1\alpha_2} H^{(\ell)}_{\alpha_3\alpha_4}\mathrm{Cov}\Big[\sigma'^{(\ell)}_{j;\alpha_1}\sigma'^{(\ell)}_{j;\alpha_2},\, \sigma'^{(\ell)}_{k;\alpha_3}\sigma'^{(\ell)}_{k;\alpha_4}\Big]\Bigg\}.\end{aligned} \tag{8.92}$$

All four of these covariances can be evaluated using the intralayer formula (8.41). After a little bit of algebra, this gives

$$\frac{n_\ell}{4n_{\ell-1}}\sum_{\gamma_1,\gamma_2,\gamma_3,\gamma_4} V^{(\gamma_1\gamma_2)(\gamma_3\gamma_4)}_{(\ell)} \left\langle\left(z_{\gamma_1}z_{\gamma_2} - G^{(\ell)}_{\gamma_1\gamma_2}\right)\widehat{\Omega}^{(\ell+1)}_{\alpha_1\alpha_2}\right\rangle_{G^{(\ell)}} \left\langle\left(z_{\gamma_3}z_{\gamma_4} - G^{(\ell)}_{\gamma_3\gamma_4}\right)\widehat{\Omega}^{(\ell+1)}_{\alpha_3\alpha_4}\right\rangle_{G^{(\ell)}} \tag{8.93}$$

at leading order, where we again made use of the definition of $\widehat{\Omega}^{(\ell+1)}_{\alpha_1\alpha_2}$ (8.74).

Finally, we're left with the off-diagonal contributions from the NTK fluctuations, which – if we write them out in excruciating detail – are given by

$$\begin{aligned}\frac{1}{n_\ell}\sum_{\substack{j,k=1\\ j\neq k}}^{n_\ell}\Bigg\{ &\left(\lambda_W^{(\ell+1)}\right) C_W^{(\ell+1)}\mathrm{Cov}\Big[\sigma^{(\ell)}_{j;\alpha_1}\sigma^{(\ell)}_{j;\alpha_2},\, \sigma'^{(\ell)}_{k;\alpha_3}\sigma'^{(\ell)}_{k;\alpha_4}\widehat{\Delta H}^{(\ell)}_{kk;\alpha_3\alpha_4}\Big] \\ &+ \left(\lambda_W^{(\ell+1)}\right) C_W^{(\ell+1)}\mathrm{Cov}\Big[\sigma'^{(\ell)}_{j;\alpha_1}\sigma'^{(\ell)}_{j;\alpha_2}\widehat{\Delta H}^{(\ell)}_{jj;\alpha_1\alpha_2},\, \sigma^{(\ell)}_{k;\alpha_3}\sigma^{(\ell)}_{k;\alpha_4}\Big] \\ &+ \left(C_W^{(\ell+1)}\right)^2 H^{(\ell)}_{\alpha_1\alpha_2}\mathrm{Cov}\Big[\sigma'^{(\ell)}_{j;\alpha_1}\sigma'^{(\ell)}_{j;\alpha_2},\, \sigma'^{(\ell)}_{k;\alpha_3}\sigma'^{(\ell)}_{k;\alpha_4}\widehat{\Delta H}^{(\ell)}_{kk;\alpha_3\alpha_4}\Big] \\ &+ \left(C_W^{(\ell+1)}\right)^2 H^{(\ell)}_{\alpha_3\alpha_4}\mathrm{Cov}\Big[\sigma'^{(\ell)}_{j;\alpha_1}\sigma'^{(\ell)}_{j;\alpha_2}\widehat{\Delta H}^{(\ell)}_{jj;\alpha_1\alpha_2},\, \sigma'^{(\ell)}_{k;\alpha_3}\sigma'^{(\ell)}_{k;\alpha_4}\Big] \\ &+ \left(C_W^{(\ell+1)}\right)^2 \mathrm{Cov}\Big[\sigma'^{(\ell)}_{j;\alpha_1}\sigma'^{(\ell)}_{j;\alpha_2}\widehat{\Delta H}^{(\ell)}_{jj;\alpha_1\alpha_2},\, \sigma'^{(\ell)}_{k;\alpha_3}\sigma'^{(\ell)}_{k;\alpha_4}\widehat{\Delta H}^{(\ell)}_{kk;\alpha_3\alpha_4}\Big]\Bigg\}.\end{aligned} \tag{8.94}$$

The last term involving the two NTK fluctuations can be evaluated similarly to how we

[13] Once again, the subleading $O(1/n)$ piece includes both a contribution from the non-Gaussian distribution as well as a cross correlation contribution from the previous layer *and* now also a contribution from the previous layer's NTK variance.

evaluated such a term for the B-recursion in (8.86),[14] here giving

$$\left(C_W^{(\ell+1)}\right)^2 \left\langle \sigma'_{\alpha_1}\sigma'_{\alpha_2}\right\rangle_{G^{(\ell)}} \left\langle \sigma'_{\alpha_3}\sigma'_{\alpha_4}\right\rangle_{G^{(\ell)}} \frac{n_\ell}{n_{\ell-1}} A^{(\ell)}_{(\alpha_1\alpha_2)(\alpha_3\alpha_4)} + O\left(\frac{1}{n}\right) . \tag{8.96}$$

The remaining four covariances in (8.94) with only a single NTK fluctuation are identical in structure to (8.76), letting us leave the details of this to you and your roll of parchment.

At this point, let's review all the components of our expression for $A^{(\ell+1)}$: we have the diagonal contribution (8.91); and off-diagonal contributions from the NTK mean (8.93), from the covariance of two NTK fluctuations (8.96), and from the four covariances on your parchment. Assembling these components, we get the A-recursion:

$$\begin{aligned}
&A^{(\ell+1)}_{(\alpha_1\alpha_2)(\alpha_3\alpha_4)} \qquad\qquad (8.97)\\
&= \left\langle \widehat{\Omega}^{(\ell+1)}_{\alpha_1\alpha_2}\widehat{\Omega}^{(\ell+1)}_{\alpha_3\alpha_4}\right\rangle_{G^{(\ell)}} - \left\langle \widehat{\Omega}^{(\ell+1)}_{\alpha_1\alpha_2}\right\rangle_{G^{(\ell)}} \left\langle \widehat{\Omega}^{(\ell+1)}_{\alpha_3\alpha_4}\right\rangle_{G^{(\ell)}} \\
&\quad + \frac{n_\ell}{4n_{\ell-1}} \sum_{\gamma_1,\gamma_2,\gamma_3,\gamma_4} V^{(\gamma_1\gamma_2)(\gamma_3\gamma_4)}_{(\ell)} \left\langle \widehat{\Omega}^{(\ell+1)}_{\alpha_1\alpha_2}\left(z_{\gamma_1}z_{\gamma_2} - G^{(\ell)}_{\gamma_1\gamma_2}\right)\right\rangle_{G^{(\ell)}} \left\langle \widehat{\Omega}^{(\ell+1)}_{\alpha_3\alpha_4}\left(z_{\gamma_3}z_{\gamma_4} - G^{(\ell)}_{\gamma_3\gamma_4}\right)\right\rangle_{G^{(\ell)}} \\
&\quad + \frac{n_\ell}{n_{\ell-1}} \left(C_W^{(\ell+1)}\right)^2 \left\langle \sigma'_{\alpha_1}\sigma'_{\alpha_2}\right\rangle_{G^{(\ell)}} \left\langle \sigma'_{\alpha_3}\sigma'_{\alpha_4}\right\rangle_{G^{(\ell)}} A^{(\ell)}_{(\alpha_1\alpha_2)(\alpha_3\alpha_4)} \\
&\quad + \frac{n_\ell}{n_{\ell-1}} \frac{C_W^{(\ell+1)}}{2} \sum_{\beta_1,\beta_2,\gamma_1,\gamma_2} \Big[\left\langle \widehat{\Omega}^{(\ell+1)}_{\alpha_1\alpha_2}\left(z_{\beta_1}z_{\beta_2} - G^{(\ell)}_{\beta_1\beta_2}\right)\right\rangle_{G^{(\ell)}} G^{\beta_1\gamma_1}_{(\ell)} G^{\beta_2\gamma_2}_{(\ell)} D^{(\ell)}_{\gamma_1\gamma_2\alpha_3\alpha_4} \left\langle \sigma'_{\alpha_3}\sigma'_{\alpha_4}\right\rangle_{G^{(\ell)}} \\
&\qquad\qquad + \left\langle \widehat{\Omega}^{(\ell+1)}_{\alpha_3\alpha_4}\left(z_{\beta_1}z_{\beta_2} - G^{(\ell)}_{\beta_1\beta_2}\right)\right\rangle_{G^{(\ell)}} G^{\beta_1\gamma_1}_{(\ell)} G^{\beta_2\gamma_2}_{(\ell)} D^{(\ell)}_{\gamma_1\gamma_2\alpha_1\alpha_2} \left\langle \sigma'_{\alpha_1}\sigma'_{\alpha_2}\right\rangle_{G^{(\ell)}} \Big] \\
&\quad + O\left(\frac{1}{n}\right) .
\end{aligned}$$

As promised, we recursively see that $A^{(\ell)}$ is an order-one quantity. Additionally, we note with interest that at leading order this A-type contribution to the NTK variance at layer $(\ell+1)$ mixes with the four-point vertex $V^{(\ell)}$ and the cross correlation $D^{(\ell)}$ in layer ℓ, though not with $B^{(\ell)}$ or $F^{(\ell)}$.

This completes our analysis of all the finite-width effects for the NTK-preactivation joint distribution; both the leading NTK-preactivation cross correlations and the NTK variance scale as $\sim 1/n$ in the large-width expansion and vanish in the infinite-width limit.[15] These quantities are sufficient to fully characterize the leading finite-width effects of gradient-based learning.

---

[14] The only difference between this and the B-version before (8.86) is that here, since we're evaluating a covariance, the term

$$\mathbb{E}\left[\sigma'^{(\ell)}_{j;\alpha_1}\sigma'^{(\ell)}_{j;\alpha_2}\widehat{\Delta H}^{(\ell)}_{jj;\alpha_1\alpha_2}\right]\mathbb{E}\left[\sigma'^{(\ell)}_{k;\alpha_3}\sigma'^{(\ell)}_{k;\alpha_4}\widehat{\Delta H}^{(\ell)}_{kk;\alpha_3\alpha_4}\right] \tag{8.95}$$

is being subtracted. However, this term is of order $O\left(1/n^2\right)$ and can thus be neglected.

[15] Recalling our discussion from footnote 11 in §2.3, this means that the NTK *self-averages* in the strict infinite-width limit; in this limit, the particular value of the NTK in any instantiation of the network parameters is fixed and equal to the ensemble mean.

Chapter 9

Effective Theory of the NTK at Initialization

> *In short, we believe that we have answered Minsky and Papert's challenge and* have *found a learning result sufficiently powerful to demonstrate that their pessimism about learning in multilayer machines was misplaced.*
>
> Rumelhart, Hinton, and Williams [15], acausally rising to meet the criticism from §5.

Since the last chapter was a tempest of equations, algebra, and integration, let's take some moments to value our expectations.

Our goal in §8 was to determine the NTK-preactivation joint distribution for a given layer ℓ at initialization: $p\left(z^{(\ell)}, \widehat{H}^{(\ell)} \middle| \mathcal{D}\right)$. The data-dependent couplings and the connected correlators of this distribution *run* with depth according to the recursions that we just laboriously derived – (8.63) for the NTK mean, (8.77) and (8.79) for the NTK-preactivation cross correlations, and (8.89) and (8.97) for the NTK variance – in addition to the recursions derived in §4 for the kernel (4.118) and for the four-point vertex (4.119). This *RG-flow* analysis taught us that the NTK is a deterministic object in the first layer (§8.1), stochastically fluctuates and cross-correlates in the second layer (§8.2), and then further accumulates fluctuations and cross correlations in deeper layers (§8.3).

Now that we've thoroughly discussed the math, in this chapter we'll finally be able to consider the physics of this joint distribution. Building on our discussion of *criticality* and *universality* in §5, we'll first lay the groundwork for a similar analysis of the NTK while highlighting the relevant results from the last chapter (§9.1). In particular, our focus will be on understanding how the initialization hyperparameters and the training hyperparameters affect gradient descent at finite width. We'll once again find that the depth-to-width ratio L/n plays a starring role in controlling finite-width effects, first for the scale-invariant universality class (§9.2) and then for the $K^\star = 0$ universality class (§9.3). For both cases, the growing importance of NTK fluctuations and cross correlations with depth makes the finite-width interaction *relevant* under RG flow of the NTK.

Finally, we'll introduce the infamous *exploding and vanishing gradient problem* of deep learning and see how our notion of criticality completely mitigates this problem (§9.4). In this context, we also explain how the bias and weight learning rates should each be scaled with the network depth.

9.1 Criticality Analysis of the NTK

Let's set the stage for our criticality analysis. As we did in our discussion of preactivation criticality in §5, throughout this section we'll set the bias variance $C_b^{(\ell)}$ and the rescaled weight variance $C_W^{(\ell)}$ to be uniform across layers

$$C_b^{(\ell)} = C_b \,, \qquad C_W^{(\ell)} = C_W \,. \tag{9.1}$$

Further paralleling §5.4, we will consider MLPs with uniform hidden layer widths

$$n_1 = \ldots = n_{L-1} \equiv n \,, \tag{9.2}$$

which is a sensible choice in practice as well as notationally simplifying.

For the training hyperparameters, however, we'll preserve the layer dependence of the bias learning rate $\lambda_b^{(\ell)}$ and weight learning rate $\lambda_W^{(\ell)}$ for now, as different universality classes will require different treatments. We'll explore the general principle behind these hyperparameter choices in §9.4.

Going forward, we'll only focus on the leading contributions from the $1/n$ expansion to the single-input statistics, neglecting the subleading corrections at next-to-leading-order and reserving the multi-input analysis for your private amusement.

Leading-order NTK recursions for a single input

Let's start with the NTK mean recursion. Analogously to all other observables, the $1/n$ expansion induces a series expansion on the NTK mean of the form

$$H^{(\ell)}_{\alpha_1\alpha_2} = H^{\{0\}(\ell)}_{\alpha_1\alpha_2} + \frac{1}{n_{\ell-1}} H^{\{1\}(\ell)}_{\alpha_1\alpha_2} + \frac{1}{n_{\ell-1}^2} H^{\{2\}(\ell)}_{\alpha_1\alpha_2} + O\left(\frac{1}{n^3}\right) . \tag{9.3}$$

Just as we defined the kernel $K^{(\ell)}_{\alpha_1\alpha_2}$ as the infinite-width limit of the mean metric $G^{(\ell)}_{\alpha_1\alpha_2}$ (4.106), let us give the leading $O(1)$ piece of the NTK mean a special symbol,

$$\Theta^{(\ell)}_{\alpha_1\alpha_2} \equiv H^{\{0\}(\ell)}_{\alpha_1\alpha_2} \,, \tag{9.4}$$

and a special name: the **frozen NTK**. The frozen NTK controls the training dynamics in the infinite-width limit, which we will investigate in detail next chapter.[1]

[1]Typically in the literature, the *neural tangent kernel* or NTK refers to this deterministic infinite-width NTK mean $\Theta^{(\ell)}_{\alpha_1\alpha_2}$. Since we are principally concerned with understanding finite-width networks, we instead chose to define and refer to the stochastic object $\widehat{H}^{(\ell)}_{i_1 i_2;\alpha_1\alpha_2}$ as the NTK. As a concession

Now, taking the leading piece of the NTK mean recursion (8.63), we get a recursion solely for the frozen NTK

$$\Theta^{(\ell+1)}_{\alpha_1\alpha_2} = \lambda_b^{(\ell+1)} + \lambda_W^{(\ell+1)} \left\langle \sigma_{\alpha_1}\sigma_{\alpha_2} \right\rangle_{K^{(\ell)}} + C_W \left\langle \sigma'_{\alpha_1}\sigma'_{\alpha_2} \right\rangle_{K^{(\ell)}} \Theta^{(\ell)}_{\alpha_1\alpha_2} \,. \tag{9.5}$$

Concurrently, as we are neglecting subleading contributions, we have exchanged the Gaussian expectations over the mean metric $G^{(\ell)}$ for ones over the kernel $K^{(\ell)}$. Finally specializing to a single input, we simply drop the sample indices to get this recursion's final form,

$$\Theta^{(\ell+1)} = \lambda_b^{(\ell+1)} + \lambda_W^{(\ell+1)} g\Big(K^{(\ell)}\Big) + \chi_\perp\Big(K^{(\ell)}\Big)\, \Theta^{(\ell)} \,, \tag{9.6}$$

with the initial condition coming directly from our first-layer NTK analysis (8.23)

$$\Theta^{(1)} = \lambda_b^{(1)} + \lambda_W^{(1)} \left(\frac{1}{n_0} \sum_{j=1}^{n_0} x_j^2 \right) . \tag{9.7}$$

Note that here we have also made use of a helper function and susceptibility from §5.

For your convenience, let us also recall and reprint the full set of helper functions – (5.5) and (5.52) – and susceptibilities – (5.50) and (5.51) – that we first made popular in §5:

$$g(K) = \left\langle \sigma(z)\, \sigma(z) \right\rangle_K \,, \tag{9.8}$$

$$h(K) \equiv \frac{C_W}{4K^2} \Big\langle \sigma'(z)\, \sigma'(z) \left(z^2 - K\right) \Big\rangle_K = \frac{1}{2} \frac{d}{dK} \chi_\perp(K) \,, \tag{9.9}$$

$$\chi_\parallel(K) = C_W g'(K) = \frac{C_W}{2K^2} \Big\langle \sigma(z)\, \sigma(z) \left(z^2 - K\right) \Big\rangle_K = \frac{C_W}{K} \left\langle z\, \sigma'(z)\, \sigma(z) \right\rangle_K \,, \tag{9.10}$$

$$\chi_\perp(K) = C_W \left\langle \sigma'(z)\, \sigma'(z) \right\rangle_K \,. \tag{9.11}$$

As a reminder, to go between the middle and right-hand expression in (9.10) you should integrate by parts.

For the remaining recursions, we're going to fast-forward the process as the procedure for converting the multi-input recursions to leading-order single-input recursions surely requires your attention but is somewhat mindless: *(i)* drop the layer dependence of the initialization hyperparameters as (9.1) and uniformize the layer widths as (9.2); *(ii)* drop sample indices everywhere; *(iii)* replace the mean metric $G^{(\ell)}$ and the NTK mean $H^{(\ell)}$ with the kernel $K^{(\ell)}$ and the frozen NTK $\Theta^{(\ell)}$, respectively;[2] and *(iv)* substitute in for

to the literature, we've used the customary symbol for the NTK, Θ, to represent the frozen NTK. (As a helpful mnemonic, note that there is an H frozen inside the Θ.) Unfortunately, you'll have to wait until §11 to understand the reason why we call the infinite-width NTK *frozen*; there we'll see how finite-width effects *defrost* the training process and make the NTK move. Here, in this chapter, you can at least see how it gets *agitated* by finite-width fluctuations.

[2]Picking nits, we should really make $1/n$ expansions – similar to (4.106) for the mean metric $G^{(\ell)}$ and (9.3) for the NTK mean $H^{(\ell)}$ – for the finite-width tensors $A^{(\ell)}$, $B^{(\ell)}$, $D^{(\ell)}$, $F^{(\ell)}$, and also properly make use of the one that we made for $V^{(\ell)}$ (4.105), denoting the leading-order pieces as $A^{\{0\}(\ell)}$ and such, and dropping the subleading pieces. For the interest of notational sanity we won't impose this on

helper functions and susceptibilities (9.8)–(9.11). In particular, this last step has the benefit of letting us recycle our results from §5 on the deep asymptotic behavior of these functions.

It will also be necessary to recall the single-input leading-order expression for the auxiliary stochastic variable (8.74),

$$\widehat{\Omega}^{(\ell+1)} \equiv \lambda_W^{(\ell+1)} \sigma(z)\sigma(z) + C_W \, \Theta^{(\ell)} \, \sigma'(z)\sigma'(z) \,, \tag{9.12}$$

which appears in the recursions for $D^{(\ell)}$ (8.77) and $A^{(\ell)}$ (8.97); we'll make this substitution the penultimate step *(iii-b)*, if you will. In making these substitutions, please keep in mind that the frozen NTK $\Theta^{(\ell)}$ multiplying the second term is not a random variable and hence can be escorted out of any Gaussian expectations.

At this point, you should grab another roll of parchment, jot down expressions (9.8)–(9.12), flip back a few pages to locate recursions (8.77), (8.79), (8.89), and (8.97), for $D^{(\ell)}$, $F^{(\ell)}$, $B^{(\ell)}$, and $A^{(\ell)}$, respectively (or perhaps you kiddos can simply click the equation references in your eBook and copy over the equations to your tablet), and simplify them according to the four-(though-sometimes-secretly-five-)step process *(i)–(iv)* above. When you're finished, make sure you agree with us:

$$\begin{aligned}
D^{(\ell+1)} &= \chi_\perp^{(\ell)} \chi_\parallel^{(\ell)} D^{(\ell)} + \left(\frac{\lambda_W^{(\ell+1)}}{C_W} \right) \left[C_W^2 \left\langle \sigma(z)\sigma(z)\sigma(z)\sigma(z) \right\rangle_{K^{(\ell)}} - \left(C_W g^{(\ell)} \right)^2 + \left(\chi_\parallel^{(\ell)} \right)^2 V^{(\ell)} \right] \\
&\quad + \Theta^{(\ell)} \left[C_W^2 \left\langle \sigma(z)\sigma(z)\sigma'(z)\sigma'(z) \right\rangle_{K^{(\ell)}} - C_W g^{(\ell)} \chi_\perp^{(\ell)} + 2h^{(\ell)} \chi_\parallel^{(\ell)} \, V^{(\ell)} \right] , \qquad (9.13) \\
F^{(\ell+1)} &= \left(\chi_\parallel^{(\ell)} \right)^2 F^{(\ell)} + C_W^2 \left\langle \sigma(z)\sigma(z)\sigma'(z)\sigma'(z) \right\rangle_{K^{(\ell)}} \Theta^{(\ell)} \,, \qquad (9.14) \\
B^{(\ell+1)} &= \left(\chi_\perp^{(\ell)} \right)^2 B^{(\ell)} + C_W^2 \left\langle \sigma'(z)\sigma'(z)\sigma'(z)\sigma'(z) \right\rangle_{K^{(\ell)}} \left(\Theta^{(\ell)} \right)^2 , \qquad (9.15) \\
A^{(\ell+1)} &= \left(\chi_\perp^{(\ell)} \right)^2 A^{(\ell)} + \left(\frac{\lambda_W^{(\ell+1)}}{C_W} \right)^2 \left[C_W^2 \left\langle \sigma(z)\sigma(z)\sigma(z)\sigma(z) \right\rangle_{K^{(\ell)}} - \left(C_W g^{(\ell)} \right)^2 + \left(\chi_\parallel^{(\ell)} \right)^2 V^{(\ell)} \right] \\
&\quad + 2 \left(\frac{\lambda_W^{(\ell+1)}}{C_W} \right) \Theta^{(\ell)} \left[C_W^2 \left\langle \sigma(z)\sigma(z)\sigma'(z)\sigma'(z) \right\rangle_{K^{(\ell)}} - C_W g^{(\ell)} \chi_\perp^{(\ell)} + 2h^{(\ell)} \chi_\parallel^{(\ell)} \, V^{(\ell)} \right] \\
&\quad + 2 \left(\frac{\lambda_W^{(\ell+1)}}{C_W} \right) \chi_\perp^{(\ell)} \chi_\parallel^{(\ell)} D^{(\ell)} + 4h^{(\ell)} \chi_\perp^{(\ell)} \Theta^{(\ell)} D^{(\ell)} \\
&\quad + \left(\Theta^{(\ell)} \right)^2 \left[C_W^2 \left\langle \sigma'(z)\sigma'(z)\sigma'(z)\sigma'(z) \right\rangle_{K^{(\ell)}} - \left(\chi_\perp^{(\ell)} \right)^2 + \left(2h^{(\ell)} \right)^2 V^{(\ell)} \right] . \qquad (9.16)
\end{aligned}$$

For these recursions, the initial conditions (recalling that the first-layer NTK is fully deterministic) all vanish identically as

$$A^{(1)} = B^{(1)} = D^{(1)} = F^{(1)} = 0 \,. \tag{9.17}$$

you, though our recursions for these tensors should all be understood as referring to these leading-order pieces. (The kernel and the frozen NTK are special in that these infinite-width objects have already been well-studied by the community, and so in this case it's important to differentiate between the finite-width object and the infinite-width piece.)

Here also, for helper functions and susceptibilities, we used the following simplifying notation

$$g^{(\ell)} \equiv g\Big(K^{(\ell)}\Big) \,, \qquad h^{(\ell)} \equiv h\Big(K^{(\ell)}\Big) \,, \qquad \chi_{\parallel}^{(\ell)} \equiv \chi_{\parallel}\Big(K^{(\ell)}\Big) \,, \qquad \chi_{\perp}^{(\ell)} \equiv \chi_{\perp}\Big(K^{(\ell)}\Big) \,, \tag{9.18}$$

making the kernel dependence implicit.

We can further simplify (9.13) and (9.16) by recalling the single-input recursion for the four-point vertex (5.109)

$$V^{(\ell+1)} = \Big(\chi_{\parallel}^{(\ell)}\Big)^2 V^{(\ell)} + C_W^2 \left[\langle \sigma(z)\sigma(z)\sigma(z)\sigma(z)\rangle_{K^{(\ell)}} - \Big(g^{(\ell)}\Big)^2\right] . \tag{9.19}$$

Keep staring at these equations, and you'll see slightly more compact expressions emerge

$$\begin{aligned} D^{(\ell+1)} &= \chi_{\perp}^{(\ell)}\chi_{\parallel}^{(\ell)} D^{(\ell)} + \left(\frac{\lambda_W^{(\ell+1)}}{C_W}\right) V^{(\ell+1)} \\ &\quad + \Theta^{(\ell)}\left[C_W^2 \left\langle \sigma(z)\sigma(z)\sigma'(z)\sigma'(z)\right\rangle_{K^{(\ell)}} - C_W g^{(\ell)}\chi_{\perp}^{(\ell)} + 2h^{(\ell)}\chi_{\parallel}^{(\ell)}\, V^{(\ell)}\right] , \end{aligned} \tag{9.20}$$

$$\begin{aligned} A^{(\ell+1)} = {} & \Big(\chi_{\perp}^{(\ell)}\Big)^2 A^{(\ell)} - \left(\frac{\lambda_W^{(\ell+1)}}{C_W}\right)^2 V^{(\ell+1)} + 2\left(\frac{\lambda_W^{(\ell+1)}}{C_W}\right) D^{(\ell+1)} + 4h^{(\ell)}\chi_{\perp}^{(\ell)}\Theta^{(\ell)} D^{(\ell)} \\ & + \Big(\Theta^{(\ell)}\Big)^2 \left[C_W^2 \left\langle \sigma'(z)\sigma'(z)\sigma'(z)\sigma'(z)\right\rangle_{K^{(\ell)}} - \Big(\chi_{\perp}^{(\ell)}\Big)^2 + \Big(2h^{(\ell)}\Big)^2 V^{(\ell)}\right] , \end{aligned} \tag{9.21}$$

which you may find makes things simpler when solving these recursions. However, please use these formulae with caution as both ℓ-th-layer and $(\ell+1)$-th-layer objects appear on their right-hand sides.

The relevance of scaling laws

For the rest of this chapter, we will work through solving the five leading-order single-input NTK recursions (9.6) and (9.13)–(9.16). (Remember that we already solved single-input recursions for the kernel and four-point vertex way back in §5.) In solving these recursions, we will find that each observable obeys our *scaling ansatz* (5.93):

$$\begin{aligned} \mathcal{O}^{(\ell)} &= \left(\frac{1}{\ell}\right)^{p_{\mathcal{O}}} \left[c_{0,0} + c_{1,1}\left(\frac{\log \ell}{\ell}\right) + c_{1,0}\left(\frac{1}{\ell}\right) + c_{2,2}\left(\frac{\log^2 \ell}{\ell^2}\right) + \ldots\right] \\ &= \left(\frac{1}{\ell}\right)^{p_{\mathcal{O}}} \left[\sum_{s=0}^{\infty}\sum_{q=0}^{s} c_{s,q}\left(\frac{\log^q \ell}{\ell^s}\right)\right] . \end{aligned} \tag{9.22}$$

Recall that $p_{\mathcal{O}}$ is a *critical exponent*, which is universal for a given universality class of activation functions, while the constants $c_{s,q}$ depend on some of the details of the particular activation function under consideration.

To properly understand the physics of these observables, recall from §5.4 that we need to consider dimensionless quantities. For the two tensors controlling the NTK

variance, we should normalize by the square of the frozen NTK

$$\frac{A^{(\ell)}}{n\left(\Theta^{(\ell)}\right)^2} \sim \frac{1}{n}\left(\frac{1}{\ell}\right)^{p_A - 2p_\Theta} + \ldots\,, \qquad \frac{B^{(\ell)}}{n\left(\Theta^{(\ell)}\right)^2} \sim \frac{1}{n}\left(\frac{1}{\ell}\right)^{p_B - 2p_\Theta} + \ldots\,, \tag{9.23}$$

while for the NTK-preactivation cross correlation, we should instead normalize by one factor of the frozen NTK and one factor of the kernel

$$\frac{D^{(\ell)}}{nK^{(\ell)}\Theta^{(\ell)}} \sim \frac{1}{n}\left(\frac{1}{\ell}\right)^{p_D - p_\Theta - p_0} + \ldots\,, \qquad \frac{F^{(\ell)}}{nK^{(\ell)}\Theta^{(\ell)}} \sim \frac{1}{n}\left(\frac{1}{\ell}\right)^{p_F - p_\Theta - p_0} + \ldots\,, \tag{9.24}$$

where p_0 was the critical exponent for the single-input kernel $K^{(\ell)}$.

By looking at these dimensionless quantities, we'll find *scaling laws* that transcend even beyond universality classes. As a particular example, recall that the normalized four-point vertex (5.128),

$$\frac{V^{(\ell)}}{n\left(K^{(\ell)}\right)^2} \sim \frac{1}{n}\left(\frac{1}{\ell}\right)^{p_V - 2p_0} + \ldots\,, \tag{9.25}$$

gave rise to a scaling law (5.129)

$$p_V - 2p_0 = -1\,, \tag{9.26}$$

for both scale-invariant and $K^\star = 0$ activation functions. This scaling law let us interpret the ratio ℓ/n as an *emergent scale* controlling the leading finite-width behavior of the preactivation distribution. *Spoiler alert:* in much the same way, we'll find scaling laws

$$p_A - 2p_\Theta = -1\,, \quad p_B - 2p_\Theta = -1\,, \quad p_D - p_\Theta - p_0 = -1\,, \quad p_F - p_\Theta - p_0 = -1\,, \tag{9.27}$$

that also hold for both the scale-invariant and $K^\star = 0$ universality classes. Thus, we'll be able to conclude that all the leading finite-width effects of the NTK-preactivation joint distribution are *relevant* and controlled by the same ℓ/n perturbative cutoff. This means that we can effectively describe the training of realistic deep networks of finite width and nonzero L/n.

Formalities: perpendicular perturbations and the frozen NTK

Before explicitly analyzing universality classes, let us note that the frozen-NTK recursion (9.6) admits a formal solution

$$\Theta^{(\ell)} = \sum_{\ell'=1}^{\ell} \left\{ \left[\lambda_b^{(\ell')} + \lambda_W^{(\ell')} g^{(\ell'-1)}\right] \left[\prod_{\ell''=\ell'}^{\ell-1} \chi_\perp^{(\ell'')}\right] \right\}. \tag{9.28}$$

In words, we see that the solution involves a sum over all the previous layers $1, \ldots, \ell$, and that each term in the sum involves an additive contribution $\lambda_b^{(\ell')} + \lambda_W^{(\ell')} g^{(\ell'-1)}$. Such a contribution then gets recursively multiplied by perpendicular susceptibilities up to

the $(\ell-1)$-th layer, resulting in an overall multiplicative factor $\prod_{\ell''=\ell'}^{\ell-1}\chi_\perp^{(\ell'')}$. To avoid the exponential behavior that's generic with such a factor, we must set $\chi_\perp = 1$.

It is enlightening to tie this insight to the discussion we had in §5.1 where we performed our general criticality analysis of the kernel recursion. There, we first looked at the single-input kernel and set $\chi_\parallel = 1$ to avoid exponential behavior in the network outputs. Then, we looked at the two-input kernel and analyzed how the off-diagonal perpendicular perturbations $\delta\delta K_{[2]}^{(\ell)}$ flow. Turning off the odd perturbations $\delta K_{[1]}^{(\ell)}$, a brief inspection of the perpendicular recursion (5.48)

$$\delta\delta K_{[2]}^{(\ell+1)} = \chi_\perp^{(\ell)}\delta\delta K_{[2]}^{(\ell)}\,, \tag{9.29}$$

necessitated the criticality condition $\chi_\perp = 1$ so as to preserve the difference between nearby inputs as they propagate through the network. At the time, we presumed that such a condition would be useful for comparing nearby inputs when learning from data. Indeed, the same multiplicative factor that appeared in the formal solution for the frozen NTK (9.28),

$$\prod_{\ell''=\ell'}^{\ell-1}\chi_\perp^{(\ell'')} = \frac{\delta\delta K_{[2]}^{(\ell)}}{\delta\delta K_{[2]}^{(\ell')}}\,, \tag{9.30}$$

also appears in a formal solution for $\delta\delta K_{[2]}^{(\ell)}$. Thus, with both formal solutions (9.28) and (9.30), we have formalized the connection between preserving $\delta\delta K_{[2]}^{(\ell)}$ data and learning from data.

With the formalities out of the way, let's now analyze our two eminent universality classes, the scale-invariant universality class and the $K^\star = 0$ universality class.

9.2 Scale-Invariant Universality Class

As a reminder, the canonical members of the scale-invariant universality class are the `ReLU` and `linear` activation functions. For a general activation function in this universality class,

$$\sigma(z) = \begin{cases} a_+ z\,, & z \geq 0\,, \\ a_- z\,, & z < 0\,, \end{cases} \tag{9.31}$$

recall from §5.2 that the helper functions and susceptibilities evaluate to

$$g^{(\ell)} = A_2 K^{(\ell)}\,, \tag{9.32}$$

$$h^{(\ell)} = 0\,, \tag{9.33}$$

$$\chi_\parallel^{(\ell)} = \chi\,, \tag{9.34}$$

$$\chi_\perp^{(\ell)} = \chi\,, \tag{9.35}$$

with $\chi \equiv C_W A_2$. By substituting in (9.31) and performing the integrals, we can just as easily evaluate the three other Gaussian expectations that we'll need

$$\left\langle \sigma(z)\sigma(z)\sigma(z)\sigma(z) \right\rangle_{K^{(\ell)}} = 3A_4 \left(K^{(\ell)}\right)^2 , \tag{9.36}$$

$$\left\langle \sigma(z)\sigma(z)\sigma'(z)\sigma'(z) \right\rangle_{K^{(\ell)}} = A_4 K^{(\ell)} , \tag{9.37}$$

$$\left\langle \sigma'(z)\sigma'(z)\sigma'(z)\sigma'(z) \right\rangle_{K^{(\ell)}} = A_4 . \tag{9.38}$$

Here and right before, we've also made use of our previous definitions for the constants that naturally arise from these integrations:

$$A_2 \equiv \frac{a_+^2 + a_-^2}{2} , \qquad A_4 \equiv \frac{a_+^4 + a_-^4}{2} . \tag{9.39}$$

With these recollections, we are reminded of one of this class's principal characteristics: both susceptibilities are independent of the kernel and constant for all layers. With that in mind, we were able to easily satisfy criticality for the scale-invariant universality class by setting the initialization hyperparameters to

$$C_b = 0 , \qquad C_W = \frac{1}{A_2} . \tag{9.40}$$

With these tunings, both susceptibilities are set to unity $\chi = 1$ and the fixed-point value of the kernel is given in terms of the input by the expression (5.66)

$$K^\star \equiv \frac{1}{A_2} \left(\frac{1}{n_0} \sum_{j=1}^{n_0} x_j^2 \right) , \tag{9.41}$$

and the four-point vertex at criticality is given by (5.120)

$$V^{(\ell)} = (\ell - 1) \left(\frac{3A_4}{A_2^2} - 1 \right) (K^\star)^2 . \tag{9.42}$$

NTK mean (frozen NTK)

With the above expressions in mind, at criticality the recursion (9.6) for the single-input frozen NTK simplifies to

$$\Theta^{(\ell+1)} = \Theta^{(\ell)} + \lambda_b^{(\ell+1)} + \lambda_W^{(\ell+1)} A_2 K^\star , \tag{9.43}$$

This recursion, together with the initial condition (9.7), is easy to solve for a given set of bias and weight learning rates.

For instance, assuming layer-independent learning rates $\lambda_b^{(\ell)} = \lambda_b$ and $\lambda_W^{(\ell)} = \lambda_W$, we find

$$\Theta^{(\ell)} = (\lambda_b + \lambda_W A_2 K^\star)\, \ell . \tag{9.44}$$

With these uniform learning rates, we see that the frozen NTK for the scale-invariant universality class grows linearly with depth. Since the NTK involves a sum over all

the previous layers (9.28), linear growth implies that these contributions are uniform across the layers; this is in contrast to the non-critical cases, for which we would have had exponentially different contributions from the different layers of a deep network, as is clear from the formal solution (9.28). Finally, a comparison with the ansatz (9.22) implies that the critical exponent for the frozen NTK is given by $p_\Theta = -1$. We will interpret all these points further in §9.4.

NTK variance and NTK-preactivation cross correlation (agitated NTK)

Now, let's evaluate our finite-width recursions (9.13)–(9.16) to find the NTK variance and the NTK-preactivation cross correlations.[3] First, we can simplify them by substituting in for the helper functions $g^{(\ell)} = A_2 K^{(\ell)}$ (9.32) and $h^{(\ell)} = 0$ (9.33) as well as making use of our formulae for the three other Gaussian expectations (9.36)–(9.38) involving A_4. Then, let us tune the initialization hyperparameters to criticality (9.40) by picking $C_W = 1/A_2$, which sets both susceptibilities to unity $\chi_\parallel^{(\ell)} = \chi_\perp^{(\ell)} = 1$ and makes the kernel fixed $K^{(\ell)} = K^\star$. With these manipulations, we get

$$D^{(\ell+1)} = D^{(\ell)} + \lambda_W A_2 \left[\left(\frac{3A_4}{A_2^2} - 1 \right) (K^\star)^2 + V^{(\ell)} \right] + \left(\frac{A_4}{A_2^2} - 1 \right) K^\star \Theta^{(\ell)} \,, \tag{9.45}$$

$$F^{(\ell+1)} = F^{(\ell)} + \frac{A_4}{A_2^2} K^\star \Theta^{(\ell)} \,, \tag{9.46}$$

$$B^{(\ell+1)} = B^{(\ell)} + \frac{A_4}{A_2^2} \left(\Theta^{(\ell)} \right)^2 \,, \tag{9.47}$$

$$\begin{aligned} A^{(\ell+1)} =& A^{(\ell)} + (\lambda_W A_2)^2 \left[\left(\frac{3A_4}{A_2^2} - 1 \right) (K^\star)^2 + V^{(\ell)} \right] \\ &+ 2\lambda_W A_2 \left(\frac{A_4}{A_2^2} - 1 \right) K^\star \Theta^{(\ell)} + 2\lambda_W A_2 D^{(\ell)} + \left(\frac{A_4}{A_2^2} - 1 \right) \left(\Theta^{(\ell)} \right)^2 \,. \end{aligned} \tag{9.48}$$

Note that we have also assumed layer-independent learning rates as we did just before when working out the NTK mean.

Next, substituting in our solutions for $V^{(\ell)}$ (9.42) and $\Theta^{(\ell)}$ (9.44), we can easily solve the recursions for $D^{(\ell)}$, $F^{(\ell)}$, and $B^{(\ell)}$. Then, with our solution for $D^{(\ell)}$ in hand, we can

[3] You could also choose to evaluate (9.20) and then (9.21) for $D^{(\ell)}$ and $A^{(\ell)}$, respectively; it's about the same level of difficulty and obviously yields the same solution either way.

also solve the recursion for $A^{(\ell)}$. All together, this gives the following solutions

$$D^{(\ell)} = \frac{\ell(\ell-1)}{2}\left[\lambda_b\left(\frac{A_4}{A_2^2}-1\right)K^\star + \lambda_W A_2\left(\frac{4A_4}{A_2^2}-2\right)(K^\star)^2\right], \tag{9.49}$$

$$F^{(\ell)} = \frac{\ell(\ell-1)}{2}\left[\frac{A_4}{A_2^2}\left(\lambda_b + \lambda_W A_2 K^\star\right)K^\star\right], \tag{9.50}$$

$$B^{(\ell)} = \frac{\ell(\ell-1)(2\ell-1)}{6}\left(\frac{A_4}{A_2^2}\right)\left(\lambda_b + \lambda_W A_2 K^\star\right)^2, \tag{9.51}$$

$$A^{(\ell)} = \frac{\ell^3}{3}\left[\left(\frac{A_4}{A_2^2}-1\right)\lambda_b^2 + 3\left(\frac{A_4}{A_2^2}-1\right)\lambda_b\lambda_W A_2 K^\star + \left(5A_4 - 3A_2^2\right)\lambda_W^2\left(K^\star\right)^2\right] + \ldots, \tag{9.52}$$

where for $A^{(\ell)}$ we kept only the leading large-ℓ contribution.

From these four solutions, we can read off another four critical exponents for the scale-invariant universality class,

$$p_D = -2\,, \qquad p_F = -2\,, \qquad p_B = -3\,, \qquad p_A = -3\,, \tag{9.53}$$

which corresponds to the quadratic growth of $D^{(\ell)}$ and $F^{(\ell)}$ and the cubic growth of $B^{(\ell)}$ and $A^{(\ell)}$. Combined with $p_0 = 0$ for the kernel and $p_\Theta = -1$ for the frozen NTK, we obtain the advertised ℓ/n-scaling (9.27) of the appropriately normalized quantities (9.23) and (9.24).

9.3 $K^\star = 0$ Universality Class

As a reminder, two notable members of this class are `tanh` and `sin`. More generally, the $K^\star = 0$ universality class contains any activation function with a corresponding kernel that has a nontrivial fixed point at $K^\star = 0$.

Specifically, recall from §5.3.3 that we used the following notation for the Taylor coefficients of an activation function:

$$\sigma(z) = \sum_{p=0}^{\infty}\frac{\sigma_p}{p!}z^p\,. \tag{9.54}$$

Then, from an analysis of the single-input kernel recursion we learned that there's nontrivial fixed point at $K^\star = 0$ if and only if the activation function vanishes at the origin with nonzero slope (5.89)

$$\sigma_0 = 0\,, \qquad \sigma_1 \neq 0\,, \tag{9.55}$$

for which we can satisfy the criticality conditions by tuning the initialization hyperparameters as (5.90)

$$C_b = 0\,, \qquad C_W = \frac{1}{\sigma_1^2}\,. \tag{9.56}$$

Going forward, we will assume that the bias variance and rescaled weight variance have been tuned to criticality as (9.56).

Unlike the scale-invariant universality class, the criticality analysis for the $K^\star = 0$ universality class was perturbative around $K = 0$. For this analysis, we expanded the helper function $g(K)$ and both susceptibilities as (5.83)–(5.85), which – now with (9.55) and (9.56) in mind – evaluate to

$$g(K) = \sigma_1^2 \left[K + a_1 K^2 + O\Big(K^3\Big) \right] , \tag{9.57}$$

$$\chi_{\parallel}(K) = 1 + 2a_1 K + O\Big(K^2\Big) , \tag{9.58}$$

$$\chi_{\perp}(K) = 1 + b_1 K + O\Big(K^2\Big) , \tag{9.59}$$

where we've also recalled the following combinations of Taylor coefficients

$$a_1 \equiv \left(\frac{\sigma_3}{\sigma_1}\right) + \frac{3}{4}\left(\frac{\sigma_2}{\sigma_1}\right)^2 , \tag{9.60}$$

$$b_1 \equiv \left(\frac{\sigma_3}{\sigma_1}\right) + \left(\frac{\sigma_2}{\sigma_1}\right)^2 . \tag{9.61}$$

As a reminder, to get these expressions we first Taylor expanded their definitions (9.8), (9.10), and (9.11) in z, and then evaluated each series of Gaussian expectations to the desired order. Following the same method, we can evaluate the helper function $h(K)$ (9.9) as well as the two other Gaussian expectations needed to solve our NTK recursions:

$$h(K) = \frac{b_1}{2} + O\Big(K^1\Big) , \tag{9.62}$$

$$\big\langle \sigma(z)\sigma(z)\sigma'(z)\sigma'(z) \big\rangle_K = \sigma_1^4 \left[K + O\Big(K^2\Big) \right] , \tag{9.63}$$

$$\big\langle \sigma'(z)\sigma'(z)\sigma'(z)\sigma'(z) \big\rangle_K = \sigma_1^4 \left[1 + O\Big(K^1\Big) \right] . \tag{9.64}$$

Remembering that the parallel susceptibility characterizes the linear response of the kernel perturbations around the fixed point (5.10), we note from above that the parallel susceptibility at criticality (9.58) is close to one near the nontrivial fixed point at $K^\star = 0$. Consequently, we found a power-law large-ℓ asymptotic solution for the single-input kernel (5.94)

$$K^{(\ell)} = K^\star + \Delta K^{(\ell)} = \Delta K^{(\ell)} = \left[\frac{1}{(-a_1)}\right] \frac{1}{\ell} + O\left(\frac{\log \ell}{\ell^2}\right) , \tag{9.65}$$

which slowly but surely approaches the $K^\star = 0$ nontrivial fixed point, justifying our perturbative approach to the deep asymptotics. As for the single-input four-point vertex, we previously found (5.127)

$$V^{(\ell)} = \left[\frac{2}{3a_1^2}\right] \frac{1}{\ell} + O\left(\frac{\log \ell}{\ell^2}\right) , \tag{9.66}$$

and the appropriately normalized quantity (9.25) has an ℓ/n scaling:

$$\frac{V^{(\ell)}}{n \left(K^{(\ell)}\right)^2} = \left(\frac{2}{3}\right) \frac{\ell}{n} + O\left(\frac{\log(\ell)}{n}\right) . \tag{9.67}$$

NTK mean (frozen NTK)

Let's start with our generic formal solution (9.28) to the frozen NTK recursion (9.6). Note that for $K^\star = 0$ activation functions the multiplicative factor (9.30) takes the form

$$\prod_{\ell''=\ell'}^{\ell-1} \chi_\perp^{(\ell'')} = \frac{\delta\delta K_{[2]}^{(\ell)}}{\delta\delta K_{[2]}^{(\ell')}} = \left(\frac{\ell'}{\ell}\right)^{p_\perp} + \ldots \,, \tag{9.68}$$

when we plug in our large-ℓ asymptotic solution for $\delta\delta K_{[2]}^{(\ell)}$ (5.99). As a reminder, the critical exponent controlling the falloff was given in terms of the Taylor coefficient combinations, $p_\perp = b_1/a_1$, which evaluates to 1 for the `tanh` and `sin` activation functions. Plugging this multiplicative factor back into the formal solution (9.28) along with our expansion for $g(K)$ (9.57) evaluated on the asymptotic kernel (9.65), we find

$$\Theta^{(\ell)} = \sum_{\ell'=1}^{\ell} \left\{ \left[\lambda_b^{(\ell')} + \lambda_W^{(\ell')} \frac{\sigma_1^2}{(-a_1)} \left(\frac{1}{\ell'}\right) + \ldots \right] \left[\left(\frac{\ell'}{\ell}\right)^{p_\perp} + \ldots \right] \right\} . \tag{9.69}$$

Here, the factor in the first square bracket is an additive contribution picked up from the ℓ'-th layer, while the factor in the second square bracket is a multiplicative contribution from recursively passing from the ℓ'-th layer to the ℓ-th layer.

Effective theorists may take issue with two aspects of this solution (9.69) if we naively continue to choose layer-independent learning rates $\lambda_b^{(\ell)} = \lambda_b$ and $\lambda_W^{(\ell)} = \lambda_W$.

- Firstly, notice in the first square bracket that the ℓ'-dependence of the bias term differs from the ℓ'-dependence of the weight term by a factor of $\sim (1/\ell')$. This means that the contribution of the weights to the NTK decreases with depth relative to the contribution of the biases.

- Secondly, notice in the second square bracket that the $\sim (\ell')^{p_\perp}$ behavior means that the NTK is dominated by contributions from deeper layers for $p_\perp > 0$ in comparison to the shallower layers. Remembering that the NTK controls the dynamics of observables (8.3), this in turn means that the training dynamics will be heavily influenced by the model parameters near the output layer.

Additionally, practical practitioners may now wonder whether these unnatural depth scalings also contribute to the empirical preference for `ReLU` over `tanh` in the deep learning community. (More on this in §9.4.)

Having said all that, we can rectify this imbalance by scaling out the layer dependence as

$$\lambda_b^{(\ell)} \equiv \widetilde{\lambda}_b \left(\frac{1}{\ell}\right)^{p_\perp} , \qquad \lambda_W^{(\ell)} \equiv \widetilde{\lambda}_W \left(\frac{1}{\ell}\right)^{p_\perp - 1} , \tag{9.70}$$

where $\widetilde{\lambda}_b$ and $\widetilde{\lambda}_W$ are layer-independent constants. Substituting this ansatz into our solution (9.69), we find

$$\Theta^{(\ell)} = \left[\widetilde{\lambda}_b + \frac{\widetilde{\lambda}_W \sigma_1^2}{(-a_1)} \right] \left(\frac{1}{\ell}\right)^{p_\perp - 1} + \ldots \,, \tag{9.71}$$

which manifestly balances the weight and bias contributions. Thus, we see for the $K^\star = 0$ universality class that the critical exponent for the frozen NTK is given by

$$p_\Theta = p_\perp - 1 \,. \tag{9.72}$$

In particular, for both the `tanh` and `sin` activation functions, $p_\Theta = 0$.

NTK variance and NTK-preactivation cross correlation (agitated NTK)

Now, let's finally deal with the agitated NTK statistics for the $K^\star = 0$ universality class. To aid our computation at criticality, let us make use of the asymptotic behavior of the kernel (9.65) and record the leading large-ℓ asymptotics of the helper functions (9.57) and (9.62), the susceptibilities (9.58) and (9.59), and the two other needed Gaussian expectations (9.63) and (9.64):

$$C_W g^{(\ell)} = \left[\frac{1}{(-a_1)}\right] \frac{1}{\ell} + \ldots \,, \tag{9.73}$$

$$h^{(\ell)} = \frac{b_1}{2} + \ldots \,, \tag{9.74}$$

$$\chi_\parallel^{(\ell)} = 1 - \frac{2}{\ell} + \ldots \,, \tag{9.75}$$

$$\chi_\perp^{(\ell)} = 1 - \frac{p_\perp}{\ell} + \ldots \,, \tag{9.76}$$

$$C_W^2 \left\langle \sigma(z)\sigma(z)\sigma'(z)\sigma'(z) \right\rangle_{K^{(\ell)}} = \left[\frac{1}{(-a_1)}\right] \frac{1}{\ell} + \ldots \,, \tag{9.77}$$

$$C_W^2 \left\langle \sigma'(z)\sigma'(z)\sigma'(z)\sigma'(z) \right\rangle_{K^{(\ell)}} = 1 + \ldots \,. \tag{9.78}$$

Additionally, going forward we will assume that the bias and weight learning rates have the layer-dependence (9.70) motivated by equal per-layer NTK contribution.

With all this out of the way, it's straightforward to evaluate the large-ℓ asymptotics of $F^{(\ell)}$ and $B^{(\ell)}$. Plugging in the above expressions and the frozen NTK asymptotic solution (9.71) into their recursions (9.14) and (9.15), we get

$$F^{(\ell+1)} = \left[1 - \frac{4}{\ell} + \ldots\right] F^{(\ell)} + \left\{ \left[\frac{1}{(-a_1)}\right] \left[\widetilde{\lambda}_b + \frac{\widetilde{\lambda}_W \sigma_1^2}{(-a_1)}\right] \left(\frac{1}{\ell}\right)^{p_\perp} + \ldots \right\} , \tag{9.79}$$

$$B^{(\ell+1)} = \left[1 - \frac{2p_\perp}{\ell} + \ldots\right] B^{(\ell)} + \left\{ \left[\widetilde{\lambda}_b + \frac{\widetilde{\lambda}_W \sigma_1^2}{(-a_1)}\right]^2 \left(\frac{1}{\ell}\right)^{2p_\perp - 2} + \ldots \right\} . \tag{9.80}$$

Substituting in our scaling ansatz (9.22), they have the following asymptotic solutions at large ℓ:

$$F^{(\ell)} = \frac{1}{(5 - p_\perp)} \left[\frac{1}{(-a_1)}\right] \left[\widetilde{\lambda}_b + \frac{\widetilde{\lambda}_W \sigma_1^2}{(-a_1)}\right] \left(\frac{1}{\ell}\right)^{p_\perp - 1} + \ldots \,, \tag{9.81}$$

$$B^{(\ell)} = \frac{1}{3} \left[\widetilde{\lambda}_b + \frac{\widetilde{\lambda}_W \sigma_1^2}{(-a_1)}\right]^2 \left(\frac{1}{\ell}\right)^{2p_\perp - 3} + \ldots \,. \tag{9.82}$$

Next, for the $D^{(\ell)}$ recursion, let's start with the slightly more compact expression (9.20). Plugging in the expressions (9.73)–(9.78) along with the learning rates (9.70) and the asymptotic solutions for the four-point vertex (9.66) and the frozen NTK (9.71), we get a recursion

$$D^{(\ell+1)} = \left[1 - \frac{(p_\perp + 2)}{\ell} + \ldots\right] D^{(\ell)} + \left\{ \left[\frac{2}{3(-a_1)}\right] \left[-p_\perp \widetilde{\lambda}_b - (p_\perp - 1) \frac{\widetilde{\lambda}_W \sigma_1^2}{(-a_1)} \right] \left(\frac{1}{\ell}\right)^{p_\perp} + \ldots \right\} . \tag{9.83}$$

This recursion can also be easily solved by using our scaling ansatz (9.22), giving

$$D^{(\ell)} = \frac{-2}{9(-a_1)} \left[p_\perp \widetilde{\lambda}_b + (p_\perp - 1) \frac{\widetilde{\lambda}_W \sigma_1^2}{(-a_1)} \right] \left(\frac{1}{\ell}\right)^{p_\perp - 1} + \ldots . \tag{9.84}$$

Finally, for $A^{(\ell)}$ recursion (9.21), the by-now-familiar routine of flipping back and forth in your book and plugging in the large-ℓ asymptotic expressions (9.73)–(9.78), the learning rates (9.70), and the asymptotic solutions for the four-point vertex (9.66), for the frozen NTK (9.71), and for $D^{(\ell)}$ (9.84), gives

$$A^{(\ell+1)} = \left[1 - \frac{2p_\perp}{\ell} + \ldots\right] A^{(\ell)} + \left\{ \frac{4}{9} \left[p_\perp \widetilde{\lambda}_b + (p_\perp - 1) \frac{\widetilde{\lambda}_W \sigma_1^2}{(-a_1)} \right]^2 \left(\frac{1}{\ell}\right)^{2p_\perp - 2} + \ldots \right\} , \tag{9.85}$$

which can be solved using the same large-ℓ scaling ansatz (9.22), giving

$$A^{(\ell)} = \frac{4}{27} \left[p_\perp \widetilde{\lambda}_b + (p_\perp - 1) \frac{\widetilde{\lambda}_W \sigma_1^2}{(-a_1)} \right]^2 \left(\frac{1}{\ell}\right)^{2p_\perp - 3} + \ldots . \tag{9.86}$$

With this, we complete our evaluation of the agitated NTK statistics for the $K^\star = 0$ universality class.[4]

Having solved all the recursions we have, let's collect and recollect the critical exponents. From (9.81) and (9.84) we collect $p_F = p_D = p_\perp - 1$, while from (9.82) and (9.86) we collect $p_B = p_A = 2p_\perp - 3$. Recollecting $p_0 = 1$ for the kernel and $p_\Theta = p_\perp - 1$ for the frozen NTK, these critical exponents for the $K^\star = 0$ universality class again obey ℓ/n-scaling (9.27) for the normalized quantities defined in (9.23) and (9.24). Together with our scale-invariant results (9.53), this means the posited relations (9.27) do indeed persists across universality classes as scaling laws.

In summary, we have found that the leading finite-width behavior of the NTK-preactivation joint distribution – as measured by the NTK variance and NTK-preacitvation cross correlation – has a *relevant* ℓ/n scaling regardless of activation function, as is natural according to the principles of our effective theory.

[4]Curiously, for activation functions with $p_\perp = 1$ such as `tanh` and `sin`, the single-input tensors $D^{(\ell)}$ and $A^{(\ell)}$ are independent of the weight learning rate at leading order.

9.4 Criticality, Exploding and Vanishing Problems, and None of That

Having now analyzed the NTK statistics of deep networks, let us culminate our discussion by revisiting our original motivation for criticality: *exploding and vanishing problems*. In particular, let us finally introduce – and then immediately abolish – the exploding and vanishing gradient problem.[5]

Traditional view on the exploding and vanishing gradient problem

Traditionally, the exploding and vanishing gradient problem is manifested by considering the behavior of the gradient of the loss for a deep network. Using the chain rule twice, the derivative of the loss with respect to a model parameter $\theta_\mu^{(\ell)}$ in the ℓ-th layer – either a bias $\theta_\mu^{(\ell)} \equiv b_j^{(\ell)}$ or a weight $\theta_\mu^{(\ell)} \equiv W_{jk}^{(\ell)}$ – takes the form

$$\frac{d\mathcal{L}_\mathcal{A}}{d\theta_\mu^{(\ell)}} = \sum_{\alpha\in\mathcal{D}} \sum_{i_L=1}^{n_L} \sum_{i_\ell=1}^{n_\ell} \epsilon_{i_L;\alpha} \frac{dz_{i_L;\alpha}^{(L)}}{dz_{i_\ell;\alpha}^{(\ell)}} \frac{dz_{i_\ell;\alpha}^{(\ell)}}{d\theta_\mu^{(\ell)}} \,. \tag{9.87}$$

In this gradient, the first factor is the *error factor* (8.14)

$$\epsilon_{i;\alpha} \equiv \frac{\partial \mathcal{L}_\mathcal{A}}{\partial z_{i;\alpha}^{(L)}} \,, \tag{9.88}$$

the final factor is a *trivial factor* (8.11)

$$\frac{dz_{i;\alpha}^{(\ell)}}{db_j^{(\ell)}} = \delta_{ij} \,, \qquad \frac{dz_{i;\alpha}^{(\ell)}}{dW_{jk}^{(\ell)}} = \delta_{ij}\, \sigma_{k;\alpha}^{(\ell-1)} \,, \tag{9.89}$$

and the middle factor is the *chain-rule factor*

$$\frac{dz_{i_L;\alpha}^{(L)}}{dz_{i_\ell;\alpha}^{(\ell)}} = \sum_{i_{\ell+1},\ldots,i_{L-1}} \frac{dz_{i_L;\alpha}^{(L)}}{dz_{i_{L-1};\alpha}^{(L-1)}} \cdots \frac{dz_{i_{\ell+1};\alpha}^{(\ell+1)}}{dz_{i_\ell;\alpha}^{(\ell)}} = \sum_{i_{\ell+1},\ldots,i_{L-1}} \prod_{\ell'=\ell}^{L-1} \left[W_{i_{\ell'+1} i_{\ell'}}^{(\ell'+1)} \sigma_{i_{\ell'};\alpha}^{\prime(\ell')} \right] \,, \tag{9.90}$$

which can be derived by iterating the backward equation (8.17) or equivalently by repeatedly using the chain rule in conjunction with the MLP forward equation (8.9). If this text causes you to experience a large error factor yourself, please flip backward to §8.0 and review our discussion of the *backpropagation* algorithm.

The point is that without any fine-tuning, the product of matrices from layer ℓ to layer L in the chain-rule factor (9.90) will generically lead to exponential behavior. Even

[5] This problem was first noticed [56, 57] in the context of training (the now somewhat deprecated) *recurrent neural networks* (RNNs), during the era when neural networks were still *neural networks* and not yet *deep learning*, that is, at a time when MLPs weren't yet deep enough for this to have been an obvious issue.

for networks of moderate depth, this makes it extremely difficult for the shallower-layer parameters to receive a well-behaved gradient and consequentially be properly trained: a vanishing gradient means that such parameters receive no training signal from the data and loss, while an exploding gradient is indicative of an instability in which the loss may increase or even blow up. This is the **exploding and vanishing gradient problem**. In a sense, this is a backward iteration dual of the already familiar *exploding and vanishing kernel problem* that arises from the forward iteration equation.[6]

Of course, not only does the chain-rule factor (9.90) need to be well behaved for stable training, but the error factor (9.88) and trivial factor (9.89) must be as well. As we'll explain next, these latter factors are directly tied to the exploding and vanishing kernel problem. However, we'll also see that our well-understood notion of criticality is already sufficient to mitigate both exploding and vanishing problems together.

Critical view on the exploding and vanishing gradient problem

Critically, let us recall from §3.2 and then §5.1 our discussion of the exploding and vanishing kernel problem. In those sections, we first motivated criticality as remedying exponential behavior in the kernel $K^{(\ell)}_{\alpha_1\alpha_2}$. As the L-th-layer kernel controls the *typical* values of the network output – and as the dataset's labels are generically order-one numbers – we suggested that such an exploding or vanishing kernel would be problematic for training. Now that we know a little about gradient descent, we can actually see a more direct manifestation of this instability by considering all the factors that make up the network's gradient (9.87).

First, let's see how the error factor (9.88) is tied to the *exploding* kernel problem. For example, the error factor for the MSE loss (7.2) is given by (7.15)

$$\epsilon_{i;\alpha} = z^{(L)}_{i;\alpha} - y_{i;\alpha} \,. \tag{9.91}$$

As you can clearly see, if the kernel explodes, then the typical output – and hence typical values of the error factor – will explode as well.[7] To ensure this does not happen, we must set $\chi_\parallel(K^\star) \leq 1$.

Second, notice that the trivial factor (9.89) for the weights is proportional to the activation. For activation functions contained in either of the scale-invariant and $K^\star = 0$ universality classes, if the kernel – and consequently the typical preactivation – is exponentially small, then the activation – and consequently the trivial factor – will be exponentially suppressed. Subsequently, the weights in the deeper layers of the network would struggle to train as they only receive an exponentially small update. Thus, in

[6] If you'd like, you can see this duality concretely by considering a deep linear network, for which the statistics of such a product of weights can be worked out exactly exactly as in §3.

[7] Getting ahead of ourselves, a precocious reader might wonder whether this matters for the *cross-entropy loss*, since for that loss the error factor will stay of order one even if the network output explodes. However, in this case the model would then be (exponentially) overconfident on its predictions and such an inductive bias would be difficult to correct via training.

order to avoid this *vanishing* kernel problem, we demand $\chi_\parallel(K^\star) \geq 1$.[8]

Combining these two observations, we see that the exploding and vanishing kernel problem is directly manifested as a subproblem of the exploding and vanishing gradient problem and further see how our criticality condition imposed on the parallel susceptibility, $\chi_\parallel(K^\star) = 1$, serves to mitigate it.

Moreover, we can shed further light on the vanishing of the trivial factor by considering its embodiment in the NTK. Considering our formal solution for the frozen NTK (9.28) and recalling the original definition (8.7), we can track the contribution of the weight derivatives as leading to the additive term $\lambda_W^{(\ell')} g^{(\ell'-1)}$. For an exponentially vanishing kernel, this factor is exponentially suppressed as

$$\lambda_W^{(\ell')} g^{(\ell'-1)} \propto K^{(\ell'-1)} \lll 1 \,, \tag{9.92}$$

since $g(K) = A_2 K$ (9.32) for the scale-invariant universality class and $g(K) = \sigma_1^2 K + O(K^2)$ (9.57) for the $K^\star = 0$ universality class. This is another way of seeing that such deeper-layer weights are not contributing to the training dynamics and, in particular, also implies that such weights will have a minimal effect on the updates to *other* parameters.

Similarly, we see that the chain-rule factor (9.90) is also encoded in the NTK in the multiplicative factor $\prod_{\ell''=\ell'}^{\ell-1} \chi_\perp^{(\ell'')}$ (9.30). In fact, such a factor was secretly always lurking in the NTK forward equation (8.8) or (8.12) as the coefficient of the recursive term. To disclose that secret, note that in the infinite-width limit the expectation of the chain-rule factor factorizes, and we see from (8.8) or (8.12) that

$$\mathbb{E}\left[\sum_{j_1,j_2} \frac{dz_i^{(\ell+1)}}{dz_{j_1}^{(\ell)}} \frac{dz_i^{(\ell+1)}}{dz_{j_2}^{(\ell)}}\right] = \mathbb{E}\left[\sum_{j_1,j_2} W_{ij_1}^{(\ell+1)} W_{ij_2}^{(\ell+1)} \sigma_{j_1}'^{(\ell)} \sigma_{j_2}'^{(\ell)}\right] = C_W \big\langle \sigma'(z)\sigma'(z) \big\rangle_{K^{(\ell)}} = \chi_\perp^{(\ell)} \,, \tag{9.93}$$

thus making this connection explicit.

As we discussed in §9.1 under the heading *Formalities: perpendicular perturbations and the frozen NTK*, we need to ensure that these multiplicative chain-rule factors are under control for training to be well behaved, on average. In particular, if $\chi_\perp(K^\star) > 1$, then the deeper-layer NTKs will exponentially explode, and the training dynamics will be unstable, potentially leading to growing losses. If instead $\chi_\perp(K^\star) < 1$, then the contribution of the biases and weights in the shallower layers to the deeper-layer NTKs will be exponentially diminished. This means both that such parameters will struggle to move via gradient-descent updates and *also* that they will struggle to influence the evolution of the parameters in the deeper layers.

All in all, we see that contribution of the chain-rule factor to the exploding and vanishing gradient problem is directly connected to an exponentially growing or decaying multiplicative factor in the NTK and further see how our criticality condition imposed on the perpendicular susceptibility, $\chi_\perp(K^\star) = 1$, serves to mitigate it.[9]

[8] Incidentally, for the scale-invariant universality class, this same logic provides an additional justification for avoiding the *exploding* kernel problem. That is, since scale-invariant activation functions don't *saturate*, if $\chi_\parallel(K^\star) > 1$, then the activation – and consequentially the trivial factor – would explode.

[9] Having now explained why *criticality* is a complete solution to the exploding and vanishing gradient

An equivalence principle for learning rates

Although originally derived by considering the behavior of the kernel recursion, we just recovered both of our criticality conditions $\chi_{\parallel}(K^\star) = 1$ and $\chi_{\perp}(K^\star) = 1$ by a direct analysis of gradient-descent updates and of the NTK forward equation. Importantly, the guiding principle we found was that each layer should not make an exponentially different contribution to the NTK.

However, there's really no reason for us to stop at the exponential level. In fact, we can further demand at the polynomial level that no type of model parameter and no layer dominate over another for training. In other words, rather than requiring *more or less equal* contributions from the parameters in different layers, we demand parametrically *equal* contributions to the NTK for each parameter group from every layer. This gives an **equivalence principle** for setting the training hyperparameters, i.e., the bias and weight learning rates. In fact, en route to this section, we already found a way to satisfy this equivalence principle for each of our universality classes.

As we retroactively saw in §9.2, the equivalence principle was easily met for activation functions contained in the scale-invariant universality class by setting the bias and weight learning rates to be layer-independent:

$$\eta\lambda_b^{(\ell)} = \frac{\eta\widetilde{\lambda}_b}{L}\,, \qquad \frac{\eta\lambda_W^{(\ell)}}{n_{\ell-1}} = \frac{\eta\widetilde{\lambda}_W}{Ln_{\ell-1}}\,. \tag{9.94}$$

Here, we have also re-emphasized the rescaling of the weight learning rates by width of the previous layer as we discussed multiple times in §8.0. In particular, you should understand the *parametrically equal contributions* provision of equivalence principle as requiring appropriate depth *and* width scalings. Also here – with our discussion in §8.0 under the heading *Scaling in the effective theory* in mind – note that we rescaled the learning rates by the overall depth L of the network so as to have an order-one NTK in the output layer. This ensures that we can naturally compare these rescaled learning rates $\eta\widetilde{\lambda}_b$ and $\eta\widetilde{\lambda}_W$ across models of different depths as well as that we have properly scaled order-one changes in observables, e.g. the loss, after any gradient-descent update.

Meanwhile for the $K^\star = 0$ universality class, we retroactively saw in §9.3 that the

problem, let us discuss remedies of *traditionality*.

One of the first heuristic solutions – first discussed in the context of recurrent neural networks – is gradient clipping [58], in which the norm of the gradient is reduced whenever it exceeds a certain threshold. As should be apparent given our discussion, such an ad hoc, distortionary, and hard-to-tune heuristic is completely unnecessary – and potentially even destructive – in networks that are at criticality.

A second heuristic solution was the adoption of the `ReLU`. Recall that activation functions such as the `tanh` and the `sigmoid` *saturate* when $|z| \to \infty$. This implies that the derivative of the activation vanishes upon saturation as $\sigma'(z) = 0$. Such saturation naturally leads to vanishing gradients, as we can easily see from the right-hand side of (9.90). It is partially within this context that practitioners adopted the `ReLU` over saturating activation functions such as the `tanh` (see, e.g. [19]). However, with our deeper and more critical understanding, we now appreciate that criticality is sufficient to mitigate this vanishing gradient problem for any activation function that admits critical initialization hyperparameters, even for saturating ones like `tanh`.

equivalence principle requires

$$\eta\lambda_b^{(\ell)} = \eta\widetilde{\lambda}_b \left(\frac{1}{\ell}\right)^{p_\perp} L^{p_\perp - 1}, \qquad \frac{\eta\lambda_W^{(\ell)}}{n_{\ell-1}} = \frac{\eta\widetilde{\lambda}_W}{n_{\ell-1}} \left(\frac{L}{\ell}\right)^{p_\perp - 1}, \tag{9.95}$$

where here we recall our discussion of having separate depth scalings for the bias and weight learning rates (9.70) as a way to ensure both uniform contributions in parameter groups and in layers to the asymptotic frozen NTK solution (9.71).[10] In particular, for odd smooth activation functions such as `tanh` and `sin` the critical exponent for perpendicular perturbations is given by $p_\perp = 1$, and we have a simpler prescription:

$$\eta\lambda_b^{(\ell)} = \frac{\eta\widetilde{\lambda}_b}{\ell}, \qquad \frac{\eta\lambda_W^{(\ell)}}{n_{\ell-1}} = \frac{\eta\widetilde{\lambda}_W}{n_{\ell-1}}. \tag{9.96}$$

With this, we further wonder whether the empirical preference for `ReLU` over `tanh` is at least partially due to the fact that the `ReLU` learning rates are naturally ℓ-independent (9.94) while the `tanh` learning rates require nontrivial ℓ-rescalings (9.96).

In conclusion, while the optimal values of the order-one training hyperparameters $\eta\widetilde{\lambda}_b$ and $\eta\widetilde{\lambda}_W$ surely depend on the specifics of a task, we expect that the layer ℓ, depth L, and width $n_{\ell-1}$ scalings dictated by the equivalence principle will lead to the least variation across network architectures with differing widths n_ℓ and depths L.

[10] Please do not confuse these ℓ-rescalings with L-rescalings. The former modifies the learning rates of a given layer ℓ by a factor of ℓ, and the later modifies the learning rates of any layer in the network by the overall network depth L.

Also, with the recent critical discussion of the exploding and vanishing kernel problem in mind, you should see that this relative ℓ-scaling between the bias and weight learning rates is just a polynomial version of the vanishing kernel problem: the equivalence principle ensures that deeper-layer weights receive polynomially non-vanishing gradients as well as contribute polynomially-equally to the training dynamics of other parameters.

Chapter 10

Kernel Learning

> *I protest against the use of an infinite quantity as an actual entity; this is never allowed in mathematics. The infinite is only a manner of speaking ....*
>
> Carl Friedrich Gauss [59].

Now that we know essentially everything we can possibly know about the initialization distribution of the preactivations *and the NTK*, it's finally time to learn *with gradients*!

In this chapter, we'll analyze the training of infinite-width neural networks by gradient-descent optimization. Of course, the infinite-width network is really only a manner of speaking, and we cannot actually instantiate one in practice. However, as we saw from our previous finite-width analyses, they can still provide a useful *model* of an actual entity when the depth-to-width ratio is sufficiently small.

Thus, the analysis of such networks is important for two reasons. First, this limit can tell us a lot about the correct scaling and tuning of our various hyperparameters; we've already seen this previously as our criticality analysis always begins at infinite width. Second, since our finite-width analysis is perturbative in $1/n$, understanding the infinite-width limit is a prerequisite for us to understand learning with more realistic finite-width networks in §11 and §∞. With those remarks in mind, let's preview our analysis of gradient-based learning at infinite width.

Just as a new biological neural network begins its journey by taking a small step, so does a freshly-initialized artificial neural network. In §10.1 we'll take such a step, observing that the gradient-descent training of infinite-width networks is simply described by the frozen NTK and that the change in the network outputs can be consitently truncated to linear order in the global learning rate. This simplicity leads us first to observe that the network's output components move independently of each other (§10.1.1) and then second to find an absence of representation learning in the hidden layers (§10.1.2). At this point you might have an uncanny sense of déjà vu, as we found the exact same limitations in §6.3 for infinite-width networks that learn via exact Bayesian inference.

After a small step comes a giant leap. In §10.2 we'll make a large parameter update and find a closed-form solution for a *fully-trained* infinite-width network. For these

networks, such a solution memorizes the entire training set, and we'll show that this solution is the same regardless of whether we reach it in one Newton step (§10.2.1) or many steps of (stochastic) gradient descent (§10.2.2), and doesn't depend on the form of the loss that we use (§10.2.3).

In fact, in §10.2.4 we'll see that the prediction of a *particular* fully-trained infinite-width network on unseen test inputs is fixed entirely by the initial network output, the frozen NTK, and the contents of the training set. To analyze this, we evaluate the statistics of the associated *ensemble*, identifying the mean (neural tangent) kernel prediction as well as the covariance of that prediction across different realizations. Recalling our discussion of approximate methods for Bayesian model fitting in §6.2.1, we are now able to make more precise the connection between gradient-based learning and maximum likelihood estimation by discussing the sense in which our distribution over fully-trained infinite-width networks is a generalized posterior distribution.

In §10.3, we'll put the predictions of these fully-trained infinite-width networks to the test. Here we'll introduce the quantitative measure of training success, the *generalization error*, and decompose it into a *bias* term and a *variance* term. The former term compares the mean predictions of the ensemble on the test inputs to the true function values from the test set, while the latter term measures the instantiation-to-instantiation fluctuations of that prediction across different fully-trained networks in our ensemble.

Naturally, there is a tradeoff between these bias and variance terms, corresponding to our preference for the ensemble to contain networks that are both flexible *and* confident. By explicitly working out the generalization error when a test input is near one of the training samples in §10.3.1, we'll see how balancing such a tradeoff gives a prescription for tuning the initialization hyperparmeters, via the principle of criticality, and for tuning the training hyperparameters, according to the learning rate equivalence principle.

In §10.3.2 we'll extend our analysis to situations where a test input is near two training samples. This will let us understand how our fully-trained networks interpolate and extrapolate, letting us comment on the activation-function-induced inductive bias of the network output in general. In particular, we'll be able to see how nonlinear activation functions are able to nonlinearly interpolate or extrapolate around two training examples.

Finally, in §10.4 we'll take a small step back to give our discussion of infinite-width networks a broader context. In particular, we'll introduce the linear model, one of the simplest models in traditional machine learning, and explain its relationship to another traditional set of algorithms known as kernel methods. This will let us see that infinite-width MLPs are essentially just linear models based on random features and dually let us identify both the infinite-width Bayesian *kernel* and the frozen neural tangent *kernel* with this more traditional notion of a *kernel*.

After discussing the limitations of such kernel methods, you will thoroughly understand the need to go beyond the infinite-width limit so that our effective theory can fully incorporate some of the more exciting properties of practical deep learning models.

10.1 A Small Step

Now let's take our first step in a journey towards the minimum of the loss. We'll begin by considering how the preactivations change in the first step after initialization at $t = 0$.

Recalling that the evolution of any neural-network observable $\mathcal{O}\left(z^{(\ell)}\right)$ is governed by the NTK through the update equation (8.3), we have for an ℓ-th-layer preactivation:

$$\begin{aligned} \text{đ}z^{(\ell)}_{i;\delta} &\equiv z^{(\ell)}_{i;\delta}(t=1) - z^{(\ell)}_{i;\delta}(t=0) \\ &= -\eta \sum_{j=1}^{n_\ell} \sum_{\tilde{\alpha}\in\mathcal{A}} \frac{d\mathcal{L}_\mathcal{A}}{dz^{(\ell)}_{j;\tilde{\alpha}}} \widehat{H}^{(\ell)}_{ij;\tilde{\alpha}\delta} + O\left(\eta^2\right) . \end{aligned} \tag{10.1}$$

Here, the ℓ-th-layer NTK $\widehat{H}^{(\ell)}_{ij;\tilde{\alpha}\delta}$ and the factor $d\mathcal{L}_\mathcal{A}/dz^{(\ell)}_{j;\tilde{\alpha}}$ are both evaluated at initialization; from now on we'll drop the explicit dependence on the number of steps t when a quantity is being evaluated at initialization $t = 0$, unless we want to emphasize it for clarity. Henceforth, we'll use the prefix đ to indicate the *update* to a quantity after the first step of gradient descent.

Further, in writing (10.1) we have resurrected the sample-index notation of alpha-with-tilde for the inputs in the training set $\tilde{\alpha} \in \mathcal{A}$, and we will soon resurrect beta-with-dot for inputs in the test set $\dot{\beta} \in \mathcal{B}$; as before, we'll also use delta-with-no-decoration for generic inputs in either set: $\delta \in \mathcal{D} = \mathcal{A} \cup \mathcal{B}$. Thus, to be explicitly clear, the update (10.1) gives the change after the first gradient descent training step in an ℓ-th-layer preactivation evaluated on a sample from either the test set or the training set.

Now, let's specialize to the infinite-width limit. Recall in this limit that the NTK self-averages, such that the NTK for *any* particular realization of the network parameters will be equal to the infinite-width NTK mean, which we have been calling the *frozen NTK*: $\widehat{H}^{(\ell)}_{ij;\tilde{\alpha}\delta} = \delta_{ij}\Theta^{(\ell)}_{\tilde{\alpha}\delta} + O(1/n)$. With this in mind, the update equation at infinite width simplifies to

$$\text{đ}z^{(\ell)}_{i;\delta} = -\eta \sum_{\tilde{\alpha}\in\mathcal{A}} \Theta^{(\ell)}_{\delta\tilde{\alpha}} \frac{d\mathcal{L}_\mathcal{A}}{dz^{(\ell)}_{i;\tilde{\alpha}}} + O\left(\frac{1}{n}\right) . \tag{10.2}$$

Here, the update does not mix neural indices as the mean of the NTK is diagonal in those indices, while the presence of off-diagonal terms in the frozen NTK would indicate that information from one training sample informs the update of another. Note importantly that we have purposefully truncated the $+\,O(\eta^2)$ part of (10.1) that contains higher-order corrections to the update from the series expansion in the global learning rate η; in §11 we'll explicitly analyze these $O(\eta^2)$ terms and show that they are suppressed by $1/n$. Thus, in the strict infinite-width limit they identically vanish, making the linear truncation exact.

In this section, we'll take a look at what such an infinite-width small-step update entails for the network outputs with $\ell = L$ (§10.1.1) and for preactivations in the final hidden layer with $\ell = L - 1$ (§10.1.2).[1] These analyses will more or less parallel §6.3.2

[1] After reading the next section, it should be clear that the results here are true even at the minimum of the loss at the end of training. We say this now to head off any potential objections of the form, "What if there are some number of steps t for which the quantity $\eta t/n$ is of order one?"

and §6.3.3, where we considered the posterior distribution for infinite-width networks updated via exact Bayesian inference.

10.1.1 No Wiring

Specializing to the network output $z_{i;\delta}^{(L)}$, the update (10.2) simply becomes

$$đz_{i;\delta}^{(L)} = -\eta \sum_{\tilde{\alpha}\in\mathcal{A}} \Theta_{\delta\tilde{\alpha}}^{(L)} \epsilon_{i;\tilde{\alpha}} \,, \tag{10.3}$$

where we recall the now-familiar error factor defined in (7.16) as

$$\epsilon_{i;\tilde{\alpha}} \equiv \frac{\partial \mathcal{L}_{\mathcal{A}}}{\partial z_{i;\tilde{\alpha}}^{(L)}} \,. \tag{10.4}$$

We're going to extensively analyze this update to network outputs in §10.2 and onwards. Here, let us just point out that the update to the i-th feature $z_{i;\delta}^{(L)}(t=1)$ depends only on the i-th component of the error factor $\epsilon_{i;\tilde{\alpha}}$. This mirrors the phenomenon of *no wiring* for the network outputs that we observed in §6.3.2 for the exact Bayesian inference at infinite width.

To be more concrete, for the MSE loss (7.2),

$$\mathcal{L}_{\mathcal{A}} \equiv \frac{1}{2} \sum_{i=1}^{n_L} \sum_{\tilde{\alpha}\in\mathcal{A}} \left(z_{i;\tilde{\alpha}}^{(L)} - y_{i;\tilde{\alpha}} \right)^2 , \tag{10.5}$$

the error factor is simply given by the difference between the true output and the initial output, $\epsilon_{i;\tilde{\alpha}} = z_{i;\tilde{\alpha}}^{(L)} - y_{i;\tilde{\alpha}}$. We thus see that all the output components move independently from each other, and there's no way for correlations between these components to be learned.[2] In addition, since (10.3) is a stochastic equation describing the update to any particular network in the ensemble, there is no wiring for any particular realization of a one-step-into-training infinite-width network.

10.1.2 No Representation Learning

We'll have to work a little harder to analyze the update to the preactivations in the penultimate layer $z_{i;\delta}^{(L-1)}$. To start, we can evaluate the derivative of the loss in the

[2] As another example, we can take a cross-entropy loss of the form $\mathcal{L}_{\mathcal{A}} = -\sum_{j,\tilde{\alpha}} p_{j;\tilde{\alpha}} \log(q_{j;\tilde{\alpha}})$; feel free to flip forward and look at (10.36). In this case we have a target distribution $p_{i;\tilde{\alpha}}$ for which we want to fit the *softmax* distribution – cf. (6.11) – of the network outputs $q_{i;\tilde{\alpha}} \equiv \exp(z_{i;\tilde{\alpha}}^{(L)}) / \big[\sum_k \exp(z_{k;\tilde{\alpha}}^{(L)})\big]$. After noting that $\partial q_{j;\tilde{\alpha}} / \partial z_{i;\tilde{\alpha}}^{(L)} = (\delta_{ij} - q_{i;\tilde{\alpha}}) q_{j;\tilde{\alpha}}$, we find for the error factor

$$\epsilon_{i;\tilde{\alpha}} = \sum_j \frac{\partial \mathcal{L}_{\mathcal{A}}}{\partial q_{j;\tilde{\alpha}}} \frac{\partial q_{j;\tilde{\alpha}}}{\partial z_{i;\tilde{\alpha}}^{(L)}} = -\sum_j \frac{p_{j;\tilde{\alpha}}}{q_{j;\tilde{\alpha}}} q_{j;\tilde{\alpha}} (\delta_{ij} - q_{i;\tilde{\alpha}}) = -p_{i;\tilde{\alpha}} + \Big(\sum_j p_{j;\tilde{\alpha}}\Big) q_{i;\tilde{\alpha}} = q_{i;\tilde{\alpha}} - p_{i;\tilde{\alpha}} \,. \tag{10.6}$$

Therefore, in this case too we see that the error factor $\epsilon_{i;\tilde{\alpha}}$ depends only on the i-th component of the softmax distribution, and no correlation between output components will be generated.

update equation (10.2) using the *backward* equation (8.17):

$$\frac{d\mathcal{L}_{\mathcal{A}}}{dz_{j;\tilde{\alpha}}^{(L-1)}} = \sum_{i=1}^{n_L} \frac{\partial \mathcal{L}_{\mathcal{A}}}{\partial z_{i;\tilde{\alpha}}^{(L)}} \frac{dz_{i;\tilde{\alpha}}^{(L)}}{dz_{j;\tilde{\alpha}}^{(L-1)}} = \sum_{i=1}^{n_L} \frac{\partial \mathcal{L}_{\mathcal{A}}}{\partial z_{i;\tilde{\alpha}}^{(L)}} W_{ij}^{(L)} \sigma'^{(L-1)}_{j;\tilde{\alpha}} \,. \tag{10.7}$$

Substituting this into the update (10.2) at $\ell = L-1$, we get a stochastic equation describing the change in the final hidden-layer representation for any particular network:

$$đz_{j;\delta}^{(L-1)} = -\eta \sum_{i=1}^{n_L} \sum_{\tilde{\alpha}\in\mathcal{A}} \Theta_{\delta\tilde{\alpha}}^{(L-1)} \frac{\partial \mathcal{L}_{\mathcal{A}}}{\partial z_{i;\tilde{\alpha}}^{(L)}} W_{ij}^{(L)} \sigma'^{(L-1)}_{j;\tilde{\alpha}} \,. \tag{10.8}$$

To make progress here, we're going to have to analyze the distribution over such updates.

First, the mean update is given by

$$\mathbb{E}\left[đz_{j;\delta}^{(L-1)}\right] = -\eta \sum_{i=1}^{n_L} \sum_{\tilde{\alpha}\in\mathcal{A}} \Theta_{\delta\tilde{\alpha}}^{(L-1)} \mathbb{E}\left[\frac{\partial \mathcal{L}_{\mathcal{A}}}{\partial z_{i;\tilde{\alpha}}^{(L)}} W_{ij}^{(L)} \sigma'^{(L-1)}_{j;\tilde{\alpha}}\right] . \tag{10.9}$$

This expectation involves an interlayer correlation between the error factor $\partial\mathcal{L}_{\mathcal{A}}/\partial z_{i;\tilde{\alpha}}^{(L)}$ from the L-th layer and the derivative of the activation $\sigma'^{(L-1)}_{j;\tilde{\alpha}}$ from the $(L-1)$-th layer, in addition to a weight insertion $W_{ij}^{(L)}$. From our previous experience we know that such interlayer expectations are suppressed by a factor of $1/n$, vanishing in the strict infinite-width limit. To be extra careful, let's compute the mean explicitly.

To do so, recall our generating function for interlayer correlations (8.53) and specialize to the penultimate layer $\ell = L-1$. (You'll probably want to flip back and remind yourself.) Since we have not previously evaluated the case with a single weight insertion, let's calculate and record it here. Differentiating the generating function once with respect to the source as $\frac{d}{d\mathcal{J}_{ij}}$ and then setting the source to zero, we get

$$\mathbb{E}\left[\mathcal{O}\left(z^{(L)}\right) W_{ij}^{(L)} \mathcal{Q}\left(z^{(L-1)}\right)\right] = \frac{C_W^{(L)}}{n_{L-1}} \sum_{\delta\in\mathcal{D}} \mathbb{E}\left[\left\langle\!\!\left\langle \frac{\partial \mathcal{O}}{\partial z_{i;\delta}^{(L)}} \right\rangle\!\!\right\rangle_{\widehat{G}^{(L)}} \sigma_{j;\delta}^{(L-1)} \mathcal{Q}\left(z^{(L-1)}\right)\right] . \tag{10.10}$$

Applying this formula to the above expression for our update (10.9), we get

$$\begin{aligned}\mathbb{E}\left[đz_{j;\delta}^{(L-1)}\right] &= -\eta \frac{C_W^{(L)}}{n_{L-1}} \sum_{i=1}^{n_L} \sum_{\tilde{\alpha}_1\tilde{\alpha}_2\in\mathcal{A}} \Theta_{\delta\tilde{\alpha}_1}^{(L-1)} \mathbb{E}\left[\left\langle\!\!\left\langle \frac{\partial^2 \mathcal{L}_{\mathcal{A}}}{\partial z_{i;\tilde{\alpha}_1}^{(L)} \partial z_{i;\tilde{\alpha}_2}^{(L)}} \right\rangle\!\!\right\rangle_{\widehat{G}^{(L)}} \sigma_{j;\tilde{\alpha}_2}^{(L-1)} \sigma'^{(L-1)}_{j;\tilde{\alpha}_1}\right] \qquad (10.11)\\ &= -\eta \frac{C_W^{(L)}}{n_{L-1}} \sum_{\tilde{\alpha}_1,\tilde{\alpha}_2\in\mathcal{A}} \Theta_{\delta\tilde{\alpha}_1}^{(L-1)} \sum_{i=1}^{n_L} \left\langle\!\!\left\langle \frac{\partial^2 \mathcal{L}_{\mathcal{A}}}{\partial z_{i;\tilde{\alpha}_1}^{(L)} \partial z_{i;\tilde{\alpha}_2}^{(L)}} \right\rangle\!\!\right\rangle_{K^{(L)}} \left\langle \sigma'_{\tilde{\alpha}_1} \sigma_{\tilde{\alpha}_2} \right\rangle_{K^{(L-1)}} + O\left(\frac{1}{n^2}\right) ,\end{aligned}$$

and see that this expression is manifestly suppressed by $1/n$. Thus, on average across our ensemble, we have found that there's no representation learning in the penultimate layer in the infinite-width limit.

Next, let's consider the variance of the update (10.8):

$$\begin{aligned}
&\mathbb{E}\left[đz^{(L-1)}_{j_1;\delta_1} đz^{(L-1)}_{j_2;\delta_2}\right] \qquad (10.12)\\
=&\eta^2 \sum_{i_1,i_2=1}^{n_L} \sum_{\tilde{\alpha}_1\tilde{\alpha}_2\in\mathcal{A}} \Theta^{(L-1)}_{\delta_1\tilde{\alpha}_1}\Theta^{(L-1)}_{\delta_2\tilde{\alpha}_2}\, \mathbb{E}\left[\frac{\partial \mathcal{L}_{\mathcal{A}}}{\partial z^{(L)}_{i_1;\tilde{\alpha}_1}} \frac{\partial \mathcal{L}_{\mathcal{A}}}{\partial z^{(L)}_{i_2;\tilde{\alpha}_2}} W^{(L)}_{i_1 j_1} W^{(L)}_{i_2 j_2} \sigma'^{(L-1)}_{j_1;\tilde{\alpha}_1} \sigma'^{(L-1)}_{j_2;\tilde{\alpha}_2}\right]\\
=&\delta_{j_1 j_2}\, \eta^2 \frac{C^{(L)}_W}{n_{L-1}} \sum_{\tilde{\alpha}_1\tilde{\alpha}_2\in\mathcal{A}} \Theta^{(L-1)}_{\delta_1\tilde{\alpha}_1}\Theta^{(L-1)}_{\delta_2\tilde{\alpha}_2} \sum_{i=1}^{n_L} \left\langle\!\!\left\langle \frac{\partial \mathcal{L}_{\mathcal{A}}}{\partial z^{(L)}_{i_1;\tilde{\alpha}_1}} \frac{\partial \mathcal{L}_{\mathcal{A}}}{\partial z^{(L)}_{i_2;\tilde{\alpha}_2}} \right\rangle\!\!\right\rangle_{K^{(L)}} \left\langle \sigma'_{\tilde{\alpha}_1}\sigma'_{\tilde{\alpha}_2}\right\rangle_{K^{(L-1)}} + O\left(\frac{1}{n^2}\right) .
\end{aligned}$$

In the last step, we applied our interlayer correlation formula with two weight insertions (8.55) and then picked up the leading contribution.[3] With this, we see that the covariance of the update,

$$\mathrm{Cov}\left[đz^{(L-1)}_{j_1;\delta_1},\, đz^{(L-1)}_{j_2;\delta_2}\right] \equiv \mathbb{E}\left[đz^{(L-1)}_{j_1;\delta_1} đz^{(L-1)}_{j_2;\delta_2}\right] - \mathbb{E}\left[đz^{(L-1)}_{j_1;\delta_1}\right]\mathbb{E}\left[đz^{(L-1)}_{j_2;\delta_2}\right] = O\left(\frac{1}{n}\right) , \qquad (10.13)$$

is manifestly suppressed by $1/n$, vanishing in the strict infinite-width limit. Since the distribution of updates to the penultimate-layer preactivations has a vanishing mean and covariance, we conclude that the distributions before and after the learning update are equal: mirroring what we found for exact Bayesian inference in §6.3.3, there's no representation learning for gradient-based learning in the infinite-width limit.[4]

10.2 A Giant Leap

That's one small step for [a] machine, one giant leap for AI.

Neil AI-Strong

In the last section, we started to understand training for infinite-width networks by taking a small step of gradient descent. Of course, what we'd actually like is to understand the behavior of fully-trained networks at the minimum of their losses. Naturally, we could continue by taking many many small steps until our networks are fully trained, and indeed this is how networks are typically trained in practice.

That said, in §10.2.1 we'll first show that we can actually fully train infinite-width networks in *one* theoretical gradient-descent step. That is, we can take a *giant leap* right to the minimum of the loss. We'll then explain in §10.2.2 that the theoretical minimum we've found by our giant leap is the same minimum we would have found in practice by taking many steps of gradient descent, or even by using *stochastic gradient descent* with

[3] Note that the second term in (8.55) is of order $O\left(1/n^2\right)$ and hence subleading.

[4] You can check the higher-order connected correlators of the update distribution, if you'd like. However, as we already said before we started these computations, these sorts of interlayer correlations are naturally suppressed by factors of $1/n$, and so will be the higher-order connected correlators.

decreasing learning rates. This equivalence makes our giant leap a powerful theoretical tool. After a brief aside about the cross-entropy loss in §10.2.3, finally in §10.2.4 we'll see how our fully-trained infinite-width networks make predictions on previously unseen examples, though a detailed analyses of these test-set predictions will be postponed until the following section.

10.2.1 Newton's Method

Our first goal is to find a single step such that the network outputs equal the true outputs,

$$z^{(L)}_{i;\tilde{\alpha}}(t=1) = y_{i;\tilde{\alpha}}\,, \tag{10.14}$$

for all samples $x_{\tilde{\alpha}}$ in the training set $\tilde{\alpha} \in \mathcal{A}$. This condition will be our definition of *fully trained*, and it's easy to see that such a condition will minimize the training loss for any of the loss functions that we've described. Recalling the gradient-descent update for neural-network outputs (10.3) and rearranging terms, we see that our giant-leap update must satisfy

$$z^{(L)}_{i;\tilde{\alpha}_1} - y_{i;\tilde{\alpha}_1} = \eta \sum_{\tilde{\alpha}_2 \in \mathcal{A}} \widetilde{\Theta}^{(L)}_{\tilde{\alpha}_1\tilde{\alpha}_2} \frac{\partial \mathcal{L}_{\mathcal{A}}}{\partial z^{(L)}_{i;\tilde{\alpha}_2}} \tag{10.15}$$

for the network to be fully trained.

As a reminder, our convention is that quantities without an explicit step argument are evaluated at the point of initialization $t = 0$; in particular, the constraint (10.15) is written solely in terms of the quantities at initialization. Additionally, note that the tilde on $\widetilde{\Theta}^{(L)}_{\tilde{\alpha}_1\tilde{\alpha}_2}$ emphasizes that it's a $N_{\mathcal{A}} \times N_{\mathcal{A}}$-dimensional submatrix of the full frozen NTK matrix $\Theta^{(L)}_{\delta_1\delta_2}$ evaluated on pairs of inputs $(x_{\tilde{\alpha}_1}, x_{\tilde{\alpha}_2})$ in the training set $\mathcal{A}$ *only*. This emphasis will soon prove itself useful, as it did before in §6.

How can we satisfy our giant-leap condition (10.15)? Since the left-hand side is exactly the error factor of the *MSE loss* (10.5), let's first specialize to the MSE loss. Plugging in (10.5) for $\mathcal{L}_{\mathcal{A}}$, we get a concrete equation to solve:

$$z^{(L)}_{i;\tilde{\alpha}_1} - y_{i;\tilde{\alpha}_1} = \sum_{\tilde{\alpha}_2 \in \mathcal{A}} \eta \widetilde{\Theta}^{(L)}_{\tilde{\alpha}_1\tilde{\alpha}_2} \left(z^{(L)}_{i;\tilde{\alpha}_2} - y_{i;\tilde{\alpha}_2} \right) . \tag{10.16}$$

However, for generic neural networks, the frozen NTK $\Theta^{(L)}_{\tilde{\alpha}_1\tilde{\alpha}_2}$ will have nonzero off-diagonal components mixing different sample indices. This unfortunately means that the condition (10.16) is impossible to satisfy by the tuning of the single global learning rate η.

Said another way, the issue is that our global learning rate η is a *scalar*, but here we need it to be *tensor* in order to undo the mixing of the sample indices by the frozen NTK. To enable this, we need to further generalize gradient descent. In our first extension of the gradient descent algorithm (7.11) – discussed under the heading *Tensorial Gradient Descent* – we introduced a *learning-rate tensor* on parameter space,

$$\eta \to \eta \lambda_{\mu\nu}\,, \tag{10.17}$$

which let us mediate how each model parameter individually contributes to the gradient-descent update of the others and let us take steps with unequal magnitudes in various directions in parameter space. The consequence of having such a learning-rate tensor was integrated into the definition of the NTK and then informed our analyses in §7–§9.[5]

Now, we need to further extend this generalization to sample indices as

$$\eta\lambda_{\mu\nu} \to \eta\lambda_{\mu\nu}\kappa^{\tilde{\alpha}_1\tilde{\alpha}_2} \,, \tag{10.18}$$

where we have introduced a new symmetric matrix $\kappa^{\tilde{\alpha}_1\tilde{\alpha}_2}$ that we will call the **Newton tensor**. This enables us to take an anisotropic step in sample space as well. Specifically, we extend the parameter update equation (7.11) to

$$đ\theta_\mu \equiv \theta_\mu(t=1) - \theta_\mu(t=0) = -\sum_{\nu,\tilde{\alpha}_1,\tilde{\alpha}_2,i} \eta\lambda_{\mu\nu}\kappa^{\tilde{\alpha}_1\tilde{\alpha}_2} \frac{dz^{(L)}_{i;\tilde{\alpha}_1}}{d\theta_\nu} \frac{\partial \mathcal{L}_{\mathcal{A}}}{\partial z^{(L)}_{i;\tilde{\alpha}_2}} \,, \tag{10.19}$$

which we will call a **second-order update**.[6] Plugging this second-order update into our expansion for the network outputs, we get

$$\begin{aligned} z^{(L)}_{i;\delta_1}(t=1) &= z^{(L)}_{i;\delta_1} + \sum_\mu \frac{dz^{(L)}_{i;\delta_1}}{d\theta_\mu} đ\theta_\mu + O\left(\frac{1}{n}\right) \\ &= z^{(L)}_{i;\delta_1} - \eta \sum_{\tilde{\alpha}_2,\tilde{\alpha}_3 \in \mathcal{A}} \Theta^{(L)}_{\delta_1\tilde{\alpha}_2}\kappa^{\tilde{\alpha}_2\tilde{\alpha}_3} \frac{\partial \mathcal{L}_{\mathcal{A}}}{\partial z^{(L)}_{i;\tilde{\alpha}_3}} + O\left(\frac{1}{n}\right) . \end{aligned} \tag{10.20}$$

Substituting this update into our fully-trained condition (10.14) and still using the MSE loss, we get a new constraint

$$z^{(L)}_{i;\tilde{\alpha}_1} - y_{i;\tilde{\alpha}_1} = \sum_{\tilde{\alpha}_2,\tilde{\alpha}_3 \in \mathcal{A}} \left(\eta\widetilde{\Theta}^{(L)}_{\tilde{\alpha}_1\tilde{\alpha}_2}\kappa^{\tilde{\alpha}_2\tilde{\alpha}_3}\right)\left(z^{(L)}_{i;\tilde{\alpha}_3} - y_{i;\tilde{\alpha}_3}\right) . \tag{10.21}$$

We'll satisfy this constraint shortly.

For a different perspective on this new second-order update, rather than modifying the optimization algorithm, we can instead find the same constraint (10.21) by adopting a different loss. Consider a *generalized* MSE loss

$$\mathcal{L}_{\mathcal{A},\,\kappa}(\theta) = \frac{1}{2}\sum_{i=1}^{n_L} \sum_{\tilde{\alpha}_1,\tilde{\alpha}_2 \in \mathcal{A}} \kappa^{\tilde{\alpha}_1\tilde{\alpha}_2}\left(z^{(L)}_{i;\tilde{\alpha}_1} - y_{i;\tilde{\alpha}_1}\right)\left(z^{(L)}_{i;\tilde{\alpha}_2} - y_{i;\tilde{\alpha}_2}\right) , \tag{10.22}$$

[5] Most importantly, this let us scale the effective learning rate differently for the biases and weights; we saw in §8.0 and then in §9.4 that this was essential for ensuring that both parameter groups get properly trained. Also, please remember that, even when written generally as $\lambda_{\mu\nu}$, our learning-rate tensor is restricted such that it does not mix parameters from different layers.

[6] The name of this update descends from the fact that similar updates are used to define optimization algorithms that incorporate information from the second derivative of the loss. Such algorithms are generally called *second-order methods*. We will show shortly that this new algorithm minimizes the loss just as well (better, actually).

where here the Newton tensor $\kappa^{\tilde{\alpha}_1\tilde{\alpha}_2}$ acts as a *metric* on sample space.[7] For this loss the derivative with respect to the network output – i.e. the error factor – is now given by

$$\frac{\partial \mathcal{L}_{\mathcal{A}}}{\partial z^{(L)}_{i;\tilde{\alpha}_2}} = \sum_{\tilde{\alpha}_3 \in \mathcal{A}} \kappa^{\tilde{\alpha}_2\tilde{\alpha}_3} \left(z^{(L)}_{i;\tilde{\alpha}_3} - y_{i;\tilde{\alpha}_3} \right) . \tag{10.23}$$

Substituting this error factor into our condition for being fully trained (10.15), we find the same constraint (10.21) using a standard gradient-descent update with our generalized MSE loss (10.22) as we did just before using our second-order update (10.19) with the standard MSE loss. Either perspective is a valid way to think about our theoretical optimization and, as we will explain more generally in §10.2.2, any of these choices of algorithms and losses will lead to the same fully-trained network.

Now, let's find the solution to our giant-leap constraint (10.21). By inspection, this is satisfiable if we can set the term in the first parenthesis to the identity matrix

$$\sum_{\tilde{\alpha}_2 \in \mathcal{A}} \eta \widetilde{\Theta}^{(L)}_{\tilde{\alpha}_1\tilde{\alpha}_2} \kappa^{\tilde{\alpha}_2\tilde{\alpha}_3} = \delta_{\tilde{\alpha}_1}^{\ \tilde{\alpha}_3} , \tag{10.24}$$

which we can ensure by setting the product of the global learning rate and the Newton tensor as

$$\eta \kappa^{\tilde{\alpha}_1\tilde{\alpha}_2} = \widetilde{\Theta}^{\tilde{\alpha}_1\tilde{\alpha}_2}_{(L)} . \tag{10.25}$$

Here, the object on the right-hand side is the *inverse* of the $N_{\mathcal{A}} \times N_{\mathcal{A}}$-dimensional submatrix $\widetilde{\Theta}^{(L)}_{\tilde{\alpha}_1\tilde{\alpha}_2}$, which as a reminder is evaluated on pairs of inputs $(x_{\tilde{\alpha}_1}, x_{\tilde{\alpha}_2})$ in the training set $\mathcal{A}$ *only*, and is defined via the equation

$$\sum_{\tilde{\alpha}_2 \in \mathcal{A}} \widetilde{\Theta}^{\tilde{\alpha}_1\tilde{\alpha}_2}_{(L)} \widetilde{\Theta}^{(L)}_{\tilde{\alpha}_2\tilde{\alpha}_3} = \delta^{\tilde{\alpha}_1}_{\ \tilde{\alpha}_3} . \tag{10.26}$$

Similarly to our work in §6.3 on infinite-width Bayesian inference, the decoration of these submatrices with tildes is useful in order to clearly distinguish these submatrices from submatrices that also involve the *test set* $\mathcal{B}$. Also, as before for the kernel and its inverse, we will always denote the NTK inverse by an object with sample indices *raised*.

The algorithm with the particular choice (10.25) is known as **Newton's method** (which acausally explains why we called $\kappa^{\tilde{\alpha}_1\tilde{\alpha}_2}$ the *Newton tensor*).[8] With it, we can simply write down a solution that fully trains the network in one step:

$$\theta^{\star}_{\mu} = \theta_{\mu}(t=0) - \sum_{\nu,\tilde{\alpha}_1,\tilde{\alpha}_2,i} \lambda_{\mu\nu} \frac{dz^{(L)}_{i;\tilde{\alpha}_1}}{d\theta_{\nu}} \widetilde{\Theta}^{\tilde{\alpha}_1\tilde{\alpha}_2}_{(L)} \left(z^{(L)}_{i;\tilde{\alpha}_2} - y_{i;\tilde{\alpha}_2} \right) . \tag{10.30}$$

[7] Similarly, we could have taken the perspective that the learning-rate tensor $\lambda_{\mu\nu}$ acts as a metric on *parameter space*.

Note also that with this interpretation, the standard MSE loss is just the generalized MSE loss with the Euclidean metric $\kappa^{\tilde{\alpha}_1\tilde{\alpha}_2} \to \delta^{\tilde{\alpha}_1\tilde{\alpha}_2}$. In some sense, it's more pleasing to write it this way if you're familiar with general relativity; writing the Newton tensor with sample indices *raised* allows us to adopt a rule of only summing over sample indices when they come in a raised-lowered pair. Similarly, note that the insertion of the Newton tensor in our second-order update (10.19) follows this pattern as well.

[8] *Newton's method* is a numerical method for finding a zero of a function. For simplicity of presentation, let's take a single-variable function $g(x)$ and suppose that we want to find a solution to the equation $g(x_\star) = 0$; note that this is equivalent to extremizing a function $L(x)$ whose derivative is $g(x)$, i.e. $L'(x) =$

In particular, this is exactly what we'd find by setting the gradient of the loss to zero and solving for the optimal parameters as in (7.7). As we explained back there, such a direct and explicit solution to an optimization problem is only available in special cases, and it turns out that this is precisely the case at infinite width.[9]

Plugging the Newton's method update in our expansion for the network outputs (10.20), we then find the fully-trained network output for a general input $\delta \in \mathcal{D}$:

$$z^{(L)}_{i;\delta}(t=1) = z^{(L)}_{i;\delta} - \sum_{\tilde{\alpha}_1,\tilde{\alpha}_2 \in \mathcal{A}} \Theta^{(L)}_{\delta\tilde{\alpha}_1} \widetilde{\Theta}^{\tilde{\alpha}_1\tilde{\alpha}_2}_{(L)} \left(z^{(L)}_{i;\tilde{\alpha}_2} - y_{i;\tilde{\alpha}_2} \right) . \tag{10.31}$$

In particular, for samples in the training set $\mathcal{A}$, the network output equals the true output $z^{(L)}_{i;\tilde{\alpha}}(t=1) = y_{i;\tilde{\alpha}}$, satisfying our condition for the network to be fully trained (10.14). In other words, our network has perfectly memorized the entire training set.[10] As such, this solution (10.30) also minimizes *any* loss $\mathcal{L}_{\mathcal{A}}(\theta)$ that is minimized by setting the network outputs to the true outputs $z^{(L)}(x_{\tilde{\alpha}};\theta) = y_{i;\tilde{\alpha}}$:

$$\theta^{\star}_{\text{Newton}} = \arg\min_{\theta} \mathcal{L}_{\mathcal{A}}(\theta) . \tag{10.32}$$

This means that regardless of whether we used the standard MSE loss (10.5) or the generalized MSE loss (10.22) (or an entirely different loss as long as it has a minimum

---

$g(x)$. Newton's method instructs us to start with some guess x_0 and then iterate as

$$x_{t+1} = x_t - \frac{g(x_t)}{g'(x_t)} = x_t - \frac{L'(x_t)}{L''(x_t)} . \tag{10.27}$$

This algorithm is based on making a linear approximation $g'(x_t) \approx [g(x_{t+1}) - g(x_t)]/(x_{t+1} - x_t)$ and then solving for $g(x_{t+1}) = 0$. In general, one needs to iterate (10.27) for several steps in order to get a good approximate solution $x_\star$. When the function is linear as $g(x) = a(x - x_\star)$, however, we get

$$x_1 = x_0 - \frac{a(x_0 - x_\star)}{a} = x_\star , \tag{10.28}$$

for any starting point x_0. Hence Newton's method can land right on the solution in one step, just like our giant leap (10.30) did.

The right-hand side of (10.27) offers another perspective: Newton's method is gradient descent with a "loss" $L(x)$ and a learning rate set as $\eta_t = 1/L''(x_t)$. To see why this is a good choice for the learning rate, let's choose a generic learning rate $x_{t+1} = x_t - \eta_t L'(x_t)$ and Taylor-expand the updated loss $L(x_{t+1})$ to the second order in η_t:

$$L(x_{t+1}) = L(x_t) - \eta_t L'(x_t)^2 + \frac{\eta_t^2}{2} L'(x_t)^2 L''(x_t) + O\left(\eta_t^3\right) . \tag{10.29}$$

Optimizing the learning rate, we see that the truncated expression on the right-hand side is minimized when $\eta_t = 1/L''(x_t)$. In particular, for a quadratic function $L(x) = a(x - x_\star)^2/2$ this truncation is exact, and Newton's method again reaches the minimum in one step. This also makes it clear why optimization algorithms based on Newton's method fall in the class of *second-order methods*: each iteration uses second-order information from the function – the second derivative $L''(x)$ – to set the locally optimal learning rate. Our giant leap expressed in (10.30) and (10.31) is doing exactly that – successfully – for the parameter optimization and for the function approximation, respectively.

[9] We'll later show in §∞.2 that perturbative solutions are possible at finite width.

[10] Since infinite-width networks have infinite parameters, it shouldn't be surprising that in this limit the network can memorize the finite training set.

at $z^{(L)}(x_{\tilde{\alpha}};\theta) = y_{i;\tilde{\alpha}}$), our solution (10.31) will faithfully describe the minimum.[11]

10.2.2 Algorithm Independence

Now let's discuss a related – and by now well anticipated – property of the infinite-width limit: given a particular initialization $z^{(L)}_{i;\delta}(t=0)$ and the frozen NTK $\Theta^{(L)}_{\delta\tilde{\alpha}_1}$, we'll always get to exactly the same minimum (10.31), whether we get there by one giant leap or we get there by a sequence of many small steps. That is, at infinite width we have **algorithm independence**.

Let's suppose that we have taken $T-1$ steps towards the minimum with a global learning rate $\eta(t)$, a Newton tensor $\kappa^{\tilde{\alpha}_1\tilde{\alpha}_2}(t)$, and loss $\mathcal{L}_{\mathcal{A}}(t)$, where these quantities will in general depend on the step t. Different choices of $\eta(t)$, $\kappa^{\tilde{\alpha}_1\tilde{\alpha}_2}(t)$, and $\mathcal{L}_{\mathcal{A}}(t)$ will lead to different optimization algorithms; included in this class are Newton's method, gradient descent, and stochastic gradient descent (SGD).[12] Iterating the update (10.20), the network outputs accumulate the changes as

$$z^{(L)}_{i;\delta}(T-1) = z^{(L)}_{i;\delta}(t=0) - \sum_{t=0}^{T-2} \sum_{\tilde{\alpha}_1,\tilde{\alpha}_2} \Theta^{(L)}_{\delta\tilde{\alpha}_1}\, \eta(t)\, \kappa^{\tilde{\alpha}_1\tilde{\alpha}_2}(t)\, \epsilon_{i;\tilde{\alpha}_2}(t)\,, \tag{10.33}$$

where $\epsilon_{i;\tilde{\alpha}_2}(t) \equiv \partial\mathcal{L}_{\mathcal{A}}(t)/\partial z^{(L)}_{i;\tilde{\alpha}_2}$ is the error factor for the training loss $\mathcal{L}_{\mathcal{A}}(t)$ evaluated with respect to the network output $z^{(L)}_{i;\tilde{\alpha}}(t)$ at step t.

Let's then suppose that in the next step $t=T$ that we reach the true minimum. We can ensure this by taking a Newton step from $z^{(L)}_{i;\delta}(T-1)$ such that

$$z^{(L)}_{i;\delta}(T) = z^{(L)}_{i;\delta}(T-1) - \sum_{\tilde{\alpha}_1,\tilde{\alpha}_2} \Theta^{(L)}_{\delta\tilde{\alpha}_1} \widetilde{\Theta}^{\tilde{\alpha}_1\tilde{\alpha}_2}_{(L)} \left[z^{(L)}_{i;\tilde{\alpha}_2}(T-1) - y_{i;\tilde{\alpha}_2} \right], \tag{10.34}$$

where here we set $\eta(T-1)\,\kappa^{\tilde{\alpha}_1\tilde{\alpha}_2}(T-1) = \widetilde{\Theta}^{\tilde{\alpha}_1\tilde{\alpha}_2}_{(L)}$ and chose the standard MSE loss at $t=T-1$ with $\epsilon_{i;\tilde{\alpha}_2}(T-1) = z^{(L)}_{i;\tilde{\alpha}_2}(T-1) - y_{i;\tilde{\alpha}_2}$.[13] Plugging in our expression for the

[11] Note that the solution (10.30) depends on the network output at initialization $z^{(L)}_{i;\delta}$, which ultimately depend on the initialization of the parameters $\theta_{\text{init}} = \theta(t=0)$. For different initializations, we will reach different solutions (10.30), each of which will minimize the loss given that particular initialization θ_{init}. We'll have more to say about this in §10.2.4.

[12] To see how this includes SGD (7.10), note that we can either restrict the loss to be a summation over a different *batch* $\mathcal{S}_t \subset \mathcal{A}$ at each step t as $\mathcal{L}_{\mathcal{A}}(t) = \mathcal{L}_{\mathcal{S}_t}$, or equivalently we can choose the Newton tensor $\kappa^{\tilde{\alpha}_1\tilde{\alpha}_2}(t)$ to project onto the subset $\mathcal{S}_t$.

[13] Note that if we had already reached a minimum at step $T-1$, then this last Newton's step in (10.34) would give no change to the network outputs: $z^{(L)}_{i;\delta}(T) = z^{(L)}_{i;\delta}(T-1)$. Thus, our argument also applies to any algorithm that already reached a minimum with $T-1$ other steps, and we do not have to actually apply the Newton step in practice. We'll address this point again in the next-to-next footnote.

network outputs after the first $T-1$ steps (10.33), we see

$$\begin{aligned} z^{(L)}_{i;\delta}(T) =& z^{(L)}_{i;\delta}(t=0) - \sum_{t=0}^{T-2} \sum_{\tilde{\alpha}_1,\tilde{\alpha}_2} \Theta^{(L)}_{\delta\tilde{\alpha}_1}\, \eta(t)\, \kappa^{\tilde{\alpha}_1\tilde{\alpha}_2}(t)\, \epsilon_{i;\tilde{\alpha}_2}(t) \\ & - \sum_{\tilde{\alpha}_1,\tilde{\alpha}_2} \Theta^{(L)}_{\delta\tilde{\alpha}_1} \widetilde{\Theta}^{\tilde{\alpha}_1\tilde{\alpha}_2}_{(L)} \left\{ \left[z^{(L)}_{i;\tilde{\alpha}_2}(t=0) - \sum_{t=0}^{T-2} \sum_{\tilde{\alpha}_3,\tilde{\alpha}_4} \widetilde{\Theta}^{(L)}_{\tilde{\alpha}_2\tilde{\alpha}_3}\, \eta(t)\, \kappa^{\tilde{\alpha}_3\tilde{\alpha}_4}(t)\, \epsilon_{i;\tilde{\alpha}_4}(t) \right] - y_{i;\tilde{\alpha}_2} \right\} \\ =& z^{(L)}_{i;\delta}(t=0) - \sum_{\tilde{\alpha}_1,\tilde{\alpha}_2 \in \mathcal{A}} \Theta^{(L)}_{\delta\tilde{\alpha}_1} \widetilde{\Theta}^{\tilde{\alpha}_1\tilde{\alpha}_2}_{(L)} \left[z^{(L)}_{i;\tilde{\alpha}_2}(t=0) - y_{i;\tilde{\alpha}_2} \right] , \end{aligned} \tag{10.35}$$

where to go from the second to the third line we made use of the defining equation for the NTK submatrix inverse (10.26), thus enabling the cancellation. What this result shows is that all the details of the training algorithm in the previous steps $\{\eta(t), \kappa^{\tilde{\alpha}_1\tilde{\alpha}_2}(t), \mathcal{L}_{\mathcal{A}}(t)\}_{t=0,\ldots,T-2}$ were erased: the network output $z^{(L)}_{i;\delta}(T)$ after our final step $t=T$ here in (10.35) is exactly the same as the network output reached after one giant leap (10.31).[14]

Thus, in the infinite-width limit the fully-trained solution is determined by *(i)* the frozen NTK $\Theta^{(L)}_{\delta\tilde{\alpha}}$, with its details depending on the *training hyperparameters* in the learning-rate tensor $\lambda_{\mu\nu}$, *(ii)* the initial newtork outputs $z^{(L)}_{i;\delta}(t=0)$, with its distribution depending on the *initialization hyperparameters*, and *(iii)* the true outputs $y_{i;\tilde{\alpha}}$ for the training set $\mathcal{A}$. It doesn't matter which loss function we used, e.g. MSE or cross-entropy, how many steps we took to get to the minimum, or whether we used gradient descent or SGD.[15] Said another way, *algorithm independence* means that these hyperparameters

[14] In §∞.2.2, we'll explicitly analyze the dynamics of another optimization algorithm – many many steps of vanilla gradient descent (7.11) – and evaluate its corresponding fully-trained solution. As expected by algorithm independence (10.35), in the infinite-width limit this solution agrees completely with other solutions obtained by different training algorithms.

[15] Some additional comments that didn't make the cut for the main body:

- Newton's method is often impractical to implement directly, since we have to compute the inverse of the frozen NTK submatrix evaluated on the entire training set, $\widetilde{\Theta}^{\tilde{\alpha}_1\tilde{\alpha}_2}_{(L)}$, similar to how we had to invert the kernel for Bayesian inference in §6.3.2. Unlike the case of exact Bayesian inference where we were considering the feasibility of the learning algorithm, here the point is that Newton's method is a theoretical tool that lets us describe a fully-trained extremely-wide network, even if the network was trained very practically by a many-step version of (stochastic) gradient descent.
- Often theorists will take the limit of a very small step size and approximate the optimization dynamics with an ordinary differential equation (ODE). Such an approximation is sometimes misleading, and here we see that it's entirely unnecessary.
- For SGD to actually converge to a minimum, you need to decrease the learning rate over the course of training, otherwise the network will fluctuate around, but never actually reach, the minimum. Intuitively, this is because at each step the optimization problem does not include the entire training set.
- A curious reader might wonder what happens if one cannot take a final step according to Newton's method, for instance because it's impractical to invert the frozen NTK submatrix. In fact, if you're already close to the minimum at $t=T-1$, i.e. if you're essentially at the end of training, then the final step to land exactly on the minimum will be extremely small, and the solution (10.35) is a very good approximation of the network before taking this last theoretical jump.

and training set uniquely specify the statistics of fully-trained networks in the infinite-width limit; Newton's method is just a nice theoretical trick to leap right to the solution. In this way, we can use the giant-leap solution (10.31) to study the outcome of all these different optimization algorithms, which is what we'll do after a brief aside about the cross-entropy loss.

10.2.3 *Aside*: Cross-Entropy Loss

Let's take a brief aside to bring the *cross-entropy loss* out of the footnotes and into the main body. In general, the cross-entropy loss for some dataset $\mathcal{D}$ is defined as

$$\mathcal{L}_{\mathcal{D}} = -\sum_{\delta\in\mathcal{D}}\sum_{i=1}^{n_{\text{out}}} p(i|x_\delta)\log[q(i|x_\delta)]\ , \tag{10.36}$$

where $p(i|x_\delta)$ is a discrete distribution over the components i of the true output

$$p(i|x_\delta) \equiv \frac{\exp[y_{i;\delta}]}{\sum_{j=1}^{n_{\text{out}}}\exp[y_{j;\delta}]}\,, \tag{10.37}$$

and $q(i|x_\delta)$ is similarly a discrete distribution over the components i of the network's output

$$q(i|x_\delta) \equiv \frac{\exp\left[z^{(L)}_{i;\delta}(t)\right]}{\sum_{j=1}^{n_{\text{out}}}\exp\left[z^{(L)}_{i;\delta}(t)\right]}\,. \tag{10.38}$$

As we mentioned when discussing the categorical hypothesis in the context of Bayesian model fitting in §6.2.1, the discrete distribution used for (10.37) and (10.38) is sometimes referred to as the *softmax* (6.11). The cross-entropy loss (10.36) is a natural measure of the closeness of discrete distributions such as (10.37) and (10.38).[16]

In particular, cross-entropy loss is the appropriate choice for *classification*, when we want to sort the input x into one of n_{out} different classes or categories. Accordingly, the softmax distribution (10.38) transforms the model's output vector with n_{out} real components into a discrete probability distribution. In contrast, the MSE loss is the appropriate choice for *regression*, when the function we want to learn is a vector of real numbers.[17] Importantly, when the initialization hyperparameters are tuned to criticality

[16]The proper measure of closeness of distributions is really the *Kullback–Leibler (KL) divergence* (A.12), which we will describe in detail in Appendix A. However, the KL divergence $KL[p\,||\,q]$ and the cross-entropy loss (10.36) only differ by a $z^{(L)}$-independent constant, the entropy $\mathcal{S}[p(i|x_\delta)] = -\sum_{\delta\in\mathcal{D}}\sum_{i=1}^{n_{\text{out}}} p(i|x_\delta)\log[p(i|x_\delta)]$ to be exact, and thus the use of one versus the other is identical under any gradient-based learning algorithm. Note also the lack of exchange symmetry in either loss between p and q. The choice in (10.36) is purposeful and reflects the fact that an untrained model is on a different footing than the true distribution from which observations arise, analogous to the asymmetry between the prior and posterior in Bayesian inference.

[17]We can think of each loss as descending from a different Bayesian hypothesis, cf. our discussion of the uncertain hypothesis and the MSE loss (6.10) and the categorical hypothesis and the softmax distribution (6.11) in the context of Bayesian model fitting in §6.2.1.

and the training hyperparameters are selected according to the learning rate equivalence principle, *both losses will be completely well behaved during training.*

When using the cross-entropy loss, typically the true outputs for the training set are given in terms of the softmax values $p(i|x_{\tilde{\alpha}})$ rather than in terms of continuous vectors $y_{i;\tilde{\alpha}}$. Even more typically, the values $p(i|x_{\tilde{\alpha}})$ specify a particular *label*, $i = i^\star_{\tilde{\alpha}}$, with *absolute certainty*, $p(i^\star_{\tilde{\alpha}}|x_{\tilde{\alpha}}) = 1$, while the rest of the components vanish, $p(i|x_{\tilde{\alpha}}) = 0$ for $i \neq i^\star_{\tilde{\alpha}}$; this is known as *hard labeling* or **one-hot encoding**, and puts the true value of the network output $y_{i^\star_{\tilde{\alpha}};\tilde{\alpha}}$ at infinity. In such case, no finite amount of training will actually reach the minimum, and in practice as you approach such a minimum the generalization of the network becomes worse and worse. To remedy this, **early stopping** of the training algorithm is used as a regularization technique to effectively get finite targets $y_{i^\star_{\tilde{\alpha}};\tilde{\alpha}}$.[18]

Now let's specialize to the current context of training neural networks in the infinite-width limit (and assume some kind of regularization is used as described above). In the last section, we noted that any loss that's minimized by setting the network outputs to the true outputs for the training set, $z^{(L)}(x_{\tilde{\alpha}};\theta) = y_{i;\tilde{\alpha}}$, is described at the minimum by the Newton's method giant-leap solution (10.32). It's easy to check that the cross-entropy loss (10.36) is minimized when $q(i|x_\delta) = p(i|x_\delta)$, and a quick inspection of (10.37) and (10.38) shows that this is obtained by the condition $z^{(L)}(x_{\tilde{\alpha}};\theta) = y_{i;\tilde{\alpha}}$.

We do need to make one additional important remark for the cross-entropy loss. Since in this setting we specify the true output in terms of a softmax $p(i|x_{\tilde{\alpha}})$ rather than an n_L-component vector of real numbers $y_{i;\tilde{\alpha}}$, there is an ambiguity in how to set the network output $z^{(L)}_{i;\tilde{\alpha}}$: any component-independent shift $y_{i;\tilde{\alpha}} \to y_{i;\tilde{\alpha}} + c_{\tilde{\alpha}}$ keeps the target distribution $p(i|x_\delta)$ invariant. However, since in this case we care not about the network outputs $z^{(L)}_{i;\delta}$, but rather their softmax $q(i|x_\delta)$ (10.38), this ambiguity doesn't matter in the end. In particular, a shift $y_{i;\tilde{\alpha}} \to y_{i;\tilde{\alpha}} + c_{\tilde{\alpha}}$ in the giant-leap solution (10.35) shifts all the output components by the same amount for each input x_δ, $\sum_{\tilde{\alpha}_1,\tilde{\alpha}_2\in\mathcal{A}} \Theta^{(L)}_{\delta\tilde{\alpha}_1} \widetilde{\Theta}^{\tilde{\alpha}_1\tilde{\alpha}_2}_{(L)} c_{\tilde{\alpha}_2}$, leading to the same softmax $q(i|x_\delta)$. Thus, we see explicitly that our solution (10.35) unambiguously describes networks fully-trained according to the cross-entropy loss.

10.2.4 Kernel Prediction

After an intelligence – artificial or otherwise – undergoes an intense memorization session, often that intelligence is then subjected to *test* with unseen problems in order to probe its actual understanding. In the context of machine learning, we typically evaluate our model's understanding by asking it to make predictions on novel inputs $x_{\dot{\beta}}$ from the *test set* $\dot{\beta} \in \mathcal{B}$.

In the infinite-width limit, the predictions of a fully-trained MLP are governed by

[18] Alternatively we can also explicitly pick a target distribution over the output classes with multiple nonzero components $p(i|x_{\tilde{\alpha}})$, which is known as *soft labeling*. This can be implemented as a regularization technique called *label smoothing*, where $p(i^\star_{\tilde{\alpha}}|x_{\tilde{\alpha}}) = 1-\epsilon$ and $p(i \neq i^\star_{\tilde{\alpha}}|x_{\tilde{\alpha}}) = \epsilon/(n_{\text{out}}-1)$, or as *knowledge distillation*, mentioned in footnote 28 of §6, when you actually want to learn such a distribution over output classes.

the stochastic equation

$$z^{(L)}_{i;\dot\beta}(T) = z^{(L)}_{i;\dot\beta} - \sum_{\tilde\alpha_1,\tilde\alpha_2\in\mathcal{A}} \Theta^{(L)}_{\dot\beta\tilde\alpha_1}\, \widetilde{\Theta}^{\tilde\alpha_1\tilde\alpha_2}_{(L)} \left(z^{(L)}_{i;\tilde\alpha_2} - y_{i;\tilde\alpha_2}\right) , \tag{10.39}$$

whether we train the model in one step (10.31) or in many steps (10.35), and regardless of the choice of loss function or any other details of the learning algorithm.[19] For clarity, note that the inverse frozen NTK $\widetilde{\Theta}^{\tilde\alpha_1\tilde\alpha_2}_{(L)}$ is taken with respect to the $N_{\mathcal{A}}$-by-$N_{\mathcal{A}}$ training-set submatrix only, while the frozen NTK $\Theta^{(L)}_{\dot\beta\tilde\alpha_1}$ is an *off-diagonal* block of the full frozen NTK, connecting an element of the training set to an element of the test set. Note also that the network outputs $z^{(L)}_{i;\delta}$ on the right-hand side of the equation are evaluated at initialization: once again, observables without any step argument should be assumed to be evaluated at initialization, while observables with a step argument T should be assumed to be evaluated at the end of training.

The stochastic equation (10.39) describes the predictions of a particular instantiation of a fully-trained neural network. The stochasticity arises from the fact that the prediction (10.39) depends on the network outputs at initialization $z^{(L)}_{i;\delta}$, which themselves depend on the particular realization of the initialized parameters $\theta_{\text{init}} \equiv \theta(t=0)$. Since we already know that such a network has completely memorized the training set so that $z^{(L)}_{i;\tilde\alpha}(T) = y_{i;\tilde\alpha}$, the stochasticity here means that any given network in the ensemble can potentially make different predictions on elements of the test set.

With that in mind, let us now compute the full distribution over such test-set predictions for our entire ensemble of fully-trained networks. Inspecting (10.39), we see that the prediction $z^{(L)}_{i;\dot\beta}(T)$ is a simple linear transformations of the Gaussian-distributed initial outputs $z^{(L)}_{i;\delta}$ and thus will itself be Gaussian. The mean prediction is simply given by

$$m^{\infty}_{i;\dot\beta} \equiv \mathbb{E}\left[z^{(L)}_{i;\dot\beta}(T)\right] = \sum_{\tilde\alpha_1,\tilde\alpha_2\in\mathcal{A}} \Theta^{(L)}_{\dot\beta\tilde\alpha_1}\, \widetilde{\Theta}^{\tilde\alpha_1\tilde\alpha_2}_{(L)} y_{i;\tilde\alpha_2} \, . \tag{10.40}$$

This expression is entirely analogous to the infinite-width posterior mean prediction for exact Bayesian inference (6.64), with a simple replacement of all types of frozen neural tangent kernels with kernels: $\Theta^{(L)} \to K^{(L)}$. (More on this soon.) Meanwhile, the covariance of the prediction (10.39) is given by

$$\begin{aligned}
&\text{Cov}\left[z^{(L)}_{i_1;\dot\beta_1}(T),\, z^{(L)}_{i_2;\dot\beta_2}(T)\right] \equiv \mathbb{E}\left[z^{(L)}_{i_1;\dot\beta_1}(T)\, z^{(L)}_{i_2;\dot\beta_2}(T)\right] - m^{\infty}_{i_1;\dot\beta_1}\, m^{\infty}_{i_2;\dot\beta_2} \\
=&\delta_{i_1 i_2}\Bigg[K^{(L)}_{\dot\beta_1\dot\beta_2} - \sum_{\tilde\alpha_1,\tilde\alpha_2\in\mathcal{A}} \Theta^{(L)}_{\dot\beta_2\tilde\alpha_1}\, \widetilde{\Theta}^{\tilde\alpha_1\tilde\alpha_2}_{(L)} K^{(L)}_{\dot\beta_1\tilde\alpha_2} - \sum_{\tilde\alpha_1,\tilde\alpha_2\in\mathcal{A}} \Theta^{(L)}_{\dot\beta_1\tilde\alpha_1}\, \widetilde{\Theta}^{\tilde\alpha_1\tilde\alpha_2}_{(L)} K^{(L)}_{\dot\beta_2\tilde\alpha_2} \\
&\quad + \sum_{\tilde\alpha_1,\tilde\alpha_2,\tilde\alpha_3,\tilde\alpha_4\in\mathcal{A}} \Theta^{(L)}_{\dot\beta_1\tilde\alpha_1}\, \widetilde{\Theta}^{\tilde\alpha_1\tilde\alpha_2}_{(L)} \Theta^{(L)}_{\dot\beta_2\tilde\alpha_3}\, \widetilde{\Theta}^{\tilde\alpha_3\tilde\alpha_4}_{(L)} K^{(L)}_{\tilde\alpha_4\tilde\alpha_2}\Bigg] .
\end{aligned} \tag{10.41}$$

[19] Note that our analyses of §10.1 apply just as much to a small step as they do to a giant leap: the fully-trained infinite-width network has neither wiring in the vectorial components of the network output (§10.1.1) nor representation learning (§10.1.2).

While this expression is somewhat complicated looking, involving both kernels and frozen NTKs, it similarly reduces to the Bayesian infinite-width posterior covariance (6.57) with the substitution $\Theta^{(L)} \to K^{(L)}$.[20]

This ensemble of network predictions (10.39), completely specified by its mean (10.40) and covariance (10.41), defines a kind of **generalized posterior distribution**. This distribution comprises a complete closed-form solution for our ensemble of infinite-width networks at the end of training, regardless of the path that we take to get there. The mean of the distribution is the prediction of the network *averaged* over instantiations, while the covariance quantifies the instantiation-to-instantiation fluctuations of these predictions.

Indeed, it is sensible to identify the ensemble of trained networks as a kind of posterior distribution, if you recall our discussion of approximation methods for Bayesian inference in §6.2.1: minimizing a training loss $\mathcal{L}_{\mathcal{A}}(\theta)$ gives the maximum likelihood estimation (MLE) of the model parameters (6.21), which we now identify with our fully-trained solution (10.32) $\theta^{\star}_{\text{MLE}} = \theta^{\star}_{\text{Newton}}$. Further recalling the content of footnote 11 in §6.2.1, for wide networks the minimum of the loss is not unique, and the MLE approach will give a family of minima parameterized by the initialization: $\theta^{\star}_{\text{MLE}}(\theta_{\text{init}})$.[21] This lack of uniqueness ultimately stems from the lingering dependence of the trained network prediction $z^{(L)}_{i;\beta}(T)$ on the initial function output $z^{(L)}_{i;\delta}$, which stochastically varies from instantiation to instantiation, cf. our prediction (10.39). Considering the ensemble over instantiations of θ_{init}, we now see exactly how this generalized distribution with the mean (10.40) and covariance (10.41) depends on the training hyperparameters $\lambda_b^{(\ell)}$ and $\lambda_W^{(\ell)}$, initialization hyperparameters $C_b^{(\ell)}$ and $C_W^{(\ell)}$, and the training data $(x_{\tilde{\alpha}}, y_{\tilde{\alpha}})_{\tilde{\alpha}\in\mathcal{A}}$.

To be a little pedantic for a paragraph, the covariance in the generalized posterior distribution (10.41) really has a different interpretation than the posterior covariance (6.57) we computed for exact Bayesian inference at infinite width. In the setting of gradient-

[20] The fact that exact Bayesian inference and gradient descent in general make different predictions is indicative of the fact that – for general hyperparameter settings – they are actually different learning algorithms.

[21] As per that same footnote, we could also try to analyze the MAP estimate (6.22) in the context of infinite-width gradient-based learning with the addition of a regularization term of the form $\sum_{\mu=1}^{P} a_\mu \theta_\mu^2$ to the loss. If you start this analysis, you'll immediately find that the gradient-descent update $\text{đ}\theta_\mu$ includes an additional term $-2\eta \sum_\nu \lambda_{\mu\nu}\, a_\nu \theta_\nu$, and after some reflection you'll likely also realize the need to define a new stochastic tensor,

$$\widehat{\mathcal{R}}^{(\ell)}_{i;\delta} \equiv \sum_{\mu,\nu} \lambda_{\mu\nu}\, a_\mu \theta_\mu \frac{dz^{(\ell)}_{i;\delta}}{d\theta_\nu}\,, \tag{10.42}$$

which has a stochastic iteration given by

$$\widehat{\mathcal{R}}^{(\ell+1)}_{i;\delta} = a_b^{(\ell+1)} \lambda_b^{(\ell+1)} b_i^{(\ell+1)} + a_W^{(\ell+1)} \frac{\lambda_W^{(\ell+1)}}{n_\ell} \sum_{j=1}^{n_\ell} W_{ij}^{(\ell+1)} \sigma^{(\ell)}_{j;\delta} + \sum_{j=1}^{n_\ell} W_{ij}^{(\ell+1)} \sigma'^{(\ell)}_{j;\delta} \widehat{\mathcal{R}}^{(\ell)}_{j;\delta}\,, \tag{10.43}$$

and whose cross-correlation with the preactivations you'll want to compute. Here, you will have defined separate layer-dependent bias and weight regularizations for the coefficients a_μ analogous to what we did for the learning-rate tensor in (8.6), and you may want to work out the interplay between these regularization hyperpameters and initialization hyperparameters for extra credit.

based learning, the covariance of the output encodes the variation in the predictions among networks in the ensemble, each corresponding to different parameter settings that still minimize the training loss $\mathcal{L}_{\mathcal{A}}(\theta)$. In the setting of exact Bayesian inference, the covariance encodes our intrinsic uncertainty about unseen data and a small uncertainty can serve as a measure of confidence in our prediction. Thus, these covariances arise for different reasons and are epistemologically quite different in nature.

However, if we can be somewhat pragmatic for a sentence, when you have multiple trained models it's not entirely unreasonable to try to think of this generalized posterior covariance (10.41) as a measure of confidence as well.

That One Place Where Gradient Descent = Exact Bayesian Inference

Just before, we casually noticed that if we replaced all frozen *neural tangent kernels* with *kernels*, $\Theta^{(L)} \to K^{(L)}$, then the generalized posterior distribution based on (10.39) reduces to the exact Bayesian posterior distribution (6.66). Let us now show how we can actually implement such a substitution in the context of gradient-based learning with a particular choice of training hyperparameters.

To see how to do this, let's put side-by-side the recursion that defines the output-layer kernel (4.118) and the recursion that defines the output-layer frozen NTK (9.5):

$$K^{(L)}_{\delta_1\delta_2} = C_b^{(L)} + C_W^{(L)} \left\langle \sigma_{\delta_1} \sigma_{\delta_2} \right\rangle_{K^{(L-1)}} , \tag{10.44}$$

$$\Theta^{(L)}_{\delta_1\delta_2} = \lambda_b^{(L)} + \lambda_W^{(L)} \left\langle \sigma_{\delta_1} \sigma_{\delta_2} \right\rangle_{K^{(L-1)}} + C_W^{(L)} \left\langle \sigma'_{\delta_1} \sigma'_{\delta_2} \right\rangle_{K^{(L-1)}} \Theta^{(L-1)}_{\delta_1\delta_2} . \tag{10.45}$$

By inspection, it's immediately clear that setting the final-layer learning rates as

$$\lambda_b^{(L)} = C_b^{(L)}, \qquad \lambda_W^{(L)} = C_W^{(L)} , \tag{10.46}$$

gives us what we want, almost:

$$\Theta^{(L)}_{\delta_1\delta_2} = K^{(L)}_{\delta_1\delta_2} + C_W^{(L)} \left\langle \sigma'_{\delta_1} \sigma'_{\delta_2} \right\rangle_{K^{(L-1)}} \Theta^{(L-1)}_{\delta_1\delta_2} . \tag{10.47}$$

To get rid of that pesky last term, we need a way to make the penultimate-layer frozen NTK $\Theta^{(L-1)}_{\delta_1\delta_2}$vanish. To ensure this, we can simply set all the other training hyperparameters to zero:

$$\lambda_b^{(\ell)} = 0 , \qquad \lambda_W^{(\ell)} = 0 , \qquad \text{for} \quad \ell < L . \tag{10.48}$$

Combined with the initial condition for the NTK recursion (8.23), this ensures that $\Theta^{(\ell)}_{\delta_1\delta_2} = 0$ for $\ell < L$, including $\ell = L-1$. Hence, this particular configuration of training hyperparameters sets

$$\Theta^{(L)}_{\delta_1\delta_2} = K^{(L)}_{\delta_1\delta_2} , \tag{10.49}$$

giving us what we wanted, exactly.

In words, this choice of the training hyperparameters (10.46) and (10.48) means that we are training only the biases and weights in the last layer. This establishes that, in the infinite-width limit, exact Bayesian inference is actually a very special case of an ensemble of networks trained with gradient-based learning.

In practice, this means that one could train an ensemble of networks with gradient descent using the training hyperparameters choices (10.46) and (10.48) in order to implement a very good approximation of exact Bayesian inference. (The approximation would become exact if you had an infinite number of networks in your ensemble.) On the one hand, unlike the exact version of Bayesian inference presented in §6.3, in this case we no longer need to explicitly store or invert the kernel. On the other hand, we may need to fully train a large number of very wide networks for our ensemble to give a good approximation, which may again be expensive in terms of computation and memory.

Interestingly, by explicitly turning off the learning in the hidden layers, we are significantly changing the features used to compute our predictions. In particular, in this case the NTK only has contributions from the biases and weights in the final layer. Intuitively, what's happening here is that we're taking random features in the penultimate layer $\sigma\left(z_i^{(L-1)}\right)$ and then explicitly training the biases $b^{(L)}$ and weights $W^{(L)}$ to fit the best possible *linear model* of these random features.[22]

10.3 Generalization

As remarked before, ultimately we care about how well a model performs on a previously unseen test set $\mathcal{B}$ as compared to the training set $\mathcal{A}$. Some fully-trained networks will *generalize* to these new examples better than others, depending strongly on their initialization and training hyperparameters.

To assess this, we can compute the **generalization error**:

$$\mathcal{E} \equiv \mathcal{L}_{\mathcal{B}} - \mathcal{L}_{\mathcal{A}} \,. \tag{10.50}$$

The generalization error is a quantitative measure of how well a network is really approximating the desired function.[23] Specifically, if the training loss is small but the test loss is large such that there's significant generalization error, then the network isn't really a good model of $f(x)$; instead it's just a lookup table of the values of $f(x)$ when x is taken from the training set $\mathcal{A}$. This is known as **overfitting**. In contrast, if the training and test losses are both small such that there's little generalization error, then we expect that our model is going beyond simple memorization.[24] As such, the generalization error is often considered to be the main quantitative measure of success of a machine learning model.

In the infinite-width limit, we saw in the last section that we can easily set the training loss to zero $\mathcal{L}_{\mathcal{A}} = 0$ for any particular network. Thus, in the current context

[22] We'll explain in §10.4 that the general version of gradient-based learning in the infinite-width limit is also a *linear model* of random features, but is constructed from a larger set of such features encompassing all the hidden layers.

[23] Without loss of generality, in the following discussion we'll implicitly assume that the minimal value of the loss is zero.

[24] If the generalization error $\mathcal{E}$ is small but the training loss $\mathcal{L}_{\mathcal{A}}$ is large, then the model is said to be *underfitting*. This situation is not really relevant for very wide networks since, as we've already explained, we can fully train them to achieve zero training loss.

the generalization error is completely assessed by the test loss,

$$\mathcal{E} = \mathcal{L}_{\mathcal{B}}\,, \tag{10.51}$$

and our current goal is to understand the statistics of the test error in order to characterize how infinite-width networks generalize.

For wide networks we know that there are many configurations of the model parameters that will minimize the training loss and memorize the training data. Some of these configurations might generalize well leading to a small test loss, while some might overfit leading to a large test loss. Thus, the statistics of the generalization error are determined by the statistics of these configurations, which are in turn determined by our initialization and training hyperparameters.

The mean generalization error captures the generalization properties of the ensemble of networks, and its variance tells us how the instantiation-to-instantiation fluctuations lead some particular networks to generalize better than others. Understanding these statistics will inform how we should pick our hyperparameters to achieve the best generalization performance on average as well as to ensure that the typical behavior of any fully-trained network is likely to be close to the average.[25]

With that in mind, let's first evaluate the MSE test loss, averaged over an ensemble of fully-trained networks:

$$\begin{aligned}
\mathbb{E}\left[\mathcal{L}_{\mathcal{B}}(T)\right] =& \mathbb{E}\left[\frac{1}{2}\sum_{i=1}^{n_L}\sum_{\dot\beta\in\mathcal{B}}\left(z^{(L)}_{i;\dot\beta}(T) - y_{i;\dot\beta}\right)^2\right] \\
=& \mathbb{E}\left[\frac{1}{2}\sum_{i=1}^{n_L}\sum_{\dot\beta\in\mathcal{B}}\left(z^{(L)}_{i;\dot\beta}(T) - m^{\infty}_{i;\dot\beta} + m^{\infty}_{i;\dot\beta} - y_{i;\dot\beta}\right)^2\right] \\
=& \frac{1}{2}\sum_{\dot\beta\in\mathcal{B}}\left\{\sum_{i=1}^{n_L}\left(m^{\infty}_{i;\dot\beta} - y_{i;\dot\beta}\right)^2 + \sum_{i=1}^{n_L}\mathrm{Cov}\left[z^{(L)}_{i;\dot\beta}(T),\, z^{(L)}_{i;\dot\beta}(T)\right]\right\}\,.
\end{aligned} \tag{10.52}$$

In the second line, we added and subtracted the infinite-width mean prediction (10.40), and to get to the third line we noted that the cross terms cancel under the expectation.

This decomposition (10.52) illustrates a type of generalized **bias-variance tradeoff**: the first term, the *bias*, measures the deviation of the mean prediction of the ensemble $m^{\infty}_{i;\dot\beta}$ from the true output $y_{i;\dot\beta}$; the second term, the *variance* – or specifically the covariance of the generalized posterior distribution (10.41) – measures the instantiation-to-instantiation fluctuations of that prediction across different models in our ensemble.

[25] In some applications of machine learning, unseen examples are further divided into two types of datasets, *(i)* a *validation set*, used generally for model selection and often specifically for tuning hyperparameters, and *(ii)* a *test set*, used to assess the generalization properties of a particular trained model. Despite our liberal usage of the term *test set*, here, as we analytically compute the statistics of the loss on unseen examples and then use them to tune the hyperparameters, what we have is really closer in meaning to a *validation set*, as we are tuning an ensemble of models rather than assessing any particular one.

The reason why this is called a *tradeoff* is that typically different settings of the hyperparameters will decrease one term at the cost of increasing the other, making the modeler have to choose between improving one at the cost of the other.[26]

For more general losses, we can Taylor expand the test loss around the mean prediction as

$$\begin{aligned}\mathcal{L}_{\mathcal{B}} =& \mathcal{L}_{\mathcal{B}}(m^\infty) + \sum_{i,\dot\beta} \left.\frac{\partial \mathcal{L}_{\mathcal{B}}}{\partial z^{(L)}_{i;\dot\beta}}\right|_{z^{(L)}=m^\infty} \left(z^{(L)}_{i;\dot\beta}(T) - m^\infty_{i;\dot\beta}\right) \\ &+ \frac{1}{2}\sum_{i_1,i_2,\dot\beta_1,\dot\beta_2} \left.\frac{\partial^2 \mathcal{L}_{\mathcal{B}}}{\partial z^{(L)}_{i_1;\dot\beta_1}\partial z^{(L)}_{i_2;\dot\beta_2}}\right|_{z^{(L)}=m^\infty} \left(z^{(L)}_{i_1;\dot\beta_1}(T) - m^\infty_{i_1;\dot\beta_1}\right)\left(z^{(L)}_{i_2;\dot\beta_2}(T) - m^\infty_{i_2;\dot\beta_2}\right) + \ldots ,\end{aligned} \tag{10.53}$$

where we've denoted the test loss evaluated at the mean prediction as

$$\mathcal{L}_{\mathcal{B}}(m^\infty) \equiv \mathcal{L}_{\mathcal{B}}\left(z^{(L)} = m^\infty\right) . \tag{10.54}$$

Performing the expectation over the ensemble and noting that $\mathbb{E}\left[z^{(L)}_{i;\dot\beta}(T) - m^\infty_{i;\dot\beta}\right] = 0$ by definition (10.40), we get

$$\mathbb{E}\left[\mathcal{L}_{\mathcal{B}}\right] = \mathcal{L}_{\mathcal{B}}(m^\infty) + \frac{1}{2}\sum_{i_1,i_2,\dot\beta_1,\dot\beta_2} \left.\frac{\partial^2 \mathcal{L}_{\mathcal{B}}}{\partial z^{(L)}_{i_1;\dot\beta_1}\partial z^{(L)}_{i_2;\dot\beta_2}}\right|_{z^{(L)}=m^\infty} \mathrm{Cov}\left[z^{(L)}_{i_1;\dot\beta_1}(T),\, z^{(L)}_{i_2;\dot\beta_2}(T)\right] + \ldots . \tag{10.55}$$

Here again we find a generalized bias-variance decomposition: the first term $\mathcal{L}_{\mathcal{B}}(m^\infty)$ is the bias, measuring the deviation of the mean prediction from the true output on the test set, and the second term is the variance, measuring the instantiation-to-instantiation uncertainty as the trace of the covariance multiplied by the Hessian of the loss with respect to the network outputs. Thus, for *any* choice of loss function, these bias and variance terms will give a good proxy for the generalization error $\mathcal{E}$ so long as our models are making predictions that are close to the mean prediction.

Now, let's see how to compute these bias and variance terms in a few different setups. In most common cases in practice the loss is *extensive* or additive in samples, i.e. $\mathcal{L}_{\mathcal{B}} = \sum_{\dot\beta\in\mathcal{B}} \mathcal{L}_{\dot\beta}$, and we can consider the test loss evaluated on one test sample at a time. Thus, for the purpose of our analysis here, the question is: for a given test input, how many training examples are relevant for making a prediction?

In §10.3.1, we'll compute the bias and variance terms of the generalization error (10.55) around one training sample using our δ expansion introduced in §5, giving another lens into hyperparameter tuning and criticality for our two universality classes.

[26] The reason why we call it *generalized* bias-variance tradeoff is that, in the *standard* bias-variance tradeoff, the expectation is over different realizations of the training set $\mathcal{A}$ rather than over different initializations of the model parameters θ_μ. In that typical setting we have only a single model, and the *bias* characterizes how well that model can be trained on each different training set – with a large bias indicative of *underfitting* – while the *variance* characterizes the fluctuations of that model's performance over the different training sets – with a large variance indicative of *overfitting*.

However, this view will be somewhat limited by the restriction of our training set to one sample.

In §10.3.2, we'll enlarge our training set to include two samples. Rather than computing the generalization error itself, here we'll be able to explore directly a different aspect of generalization: how a fully-trained network either interpolates or extrapolates to make predictions.

10.3.1 Bias-Variance Tradeoff and Criticality

Let us index one training sample by $\tilde{\alpha} = +$ and a nearby test sample by $\dot{\beta} = -$; let us also focus on the output layer $\ell = L$ and temporarily drop the layer index from the frozen NTK, $\Theta_{\delta_1\delta_2} \equiv \Theta^{(L)}_{\delta_1\delta_2}$, until later when we need to discuss the depth dependence of various NTK components.

The bias term in the generalization error is determined by the deviation of the mean prediction (10.40) from the true output:

$$\begin{aligned} m^{\infty}_{i;-} - y_{i;-} &= \frac{\Theta_{-+}}{\Theta_{++}} y_{i;+} - y_{i;-} \\ &= (y_{i;+} - y_{i;-}) + \left(\frac{\Theta_{-+}}{\Theta_{++}} - 1\right) y_{i;+} \,. \end{aligned} \tag{10.56}$$

In this expression, the first term is the true difference in the function outputs on the two different inputs, $f(x_+) - f(x_-)$, while the second term is a similar (expected) difference between our predicted output on x_- and the learned true output on x_+, $z_-(T) - z_+(T)$.[27] Note the opposite ordering of $+$ and $-$ in these two terms: if our prediction is exactly correct, these two terms are equal in magnitude and opposite in sign, and the bias term in the generalization error will vanish.

With that in mind, the quantity in parenthesis in the second factor of the bias term (10.56),

$$\frac{\Theta_{-+}}{\Theta_{++}} - 1 \,, \tag{10.57}$$

serves as a natural measure of *robustness* since it characterizes how sensitively our prediction changes – i.e. how $|z_-(T) - z_+(T)|$ grows – with corresponding small changes in the input. A model that isn't robust will often be incorrect, making predictions that vary greatly from the network output on nearby training points, while too robust a model will not have much flexibility in its output. Since a priori we don't know what type of function we are going to approximate, we naturally would want to pick hyperparameters that include a class of networks that are robust, but not overly inflexible.[28]

[27] In general, the bias term $\mathcal{L}_{\mathcal{B}}(m^{\infty})$ in the generalization error (10.55) depends on the details of the loss. Of course, we can expand $\mathcal{L}_{\mathcal{B}}(m^{\infty})$ around the true output $y_{i;-}$, and the expansion will depend on the difference $m^{\infty}_{i;-} - y_{i;-}$ (10.56). For the MSE loss, the bias is precisely the square of this difference. For the cross-entropy loss, it is more natural to expand in terms of the difference $\overline{q}(i|x_-) - p(i|x_-)$, with $\overline{q}(i|x_\delta) \equiv \exp\left[m^{\infty}_{i;\delta}\right] / \sum_{j=1}^{n_{\text{out}}} \exp\left[m^{\infty}_{j;\delta}\right]$.

[28] A dedicated reader might notice the parallel with §6.3.1 where we argued for criticality by considering the evidence for two inputs with differing true outputs, $f(x_+) - f(x_-) \neq 0$, which called for similar flexibility in choice of function approximators.

Now, since we're considering a test input that's nearby our training input, that should remind you of our δ expansion from our criticality analysis in §5.1.[29] Specifically, we can expand the frozen NTK in our $\gamma^{[a]}$ basis as we did for the kernel in (5.15),

$$\Theta_{\pm\pm} = \Theta_{[0]} \pm \Theta_{[1]} + \Theta_{[2]} \,, \qquad \Theta_{\pm\mp} = \Theta_{[0]} - \Theta_{[2]} \,, \tag{10.58}$$

and make δ expansions similar to the ones we did for the kernel in (5.22)–(5.24),

$$\Theta_{[0]} = \Theta_{00} + \delta\delta\Theta_{[0]} + O\Big(\delta^4\Big) \,, \tag{10.59}$$

$$\Theta_{[1]} = \delta\Theta_{[1]} + O\Big(\delta^3\Big) \,, \tag{10.60}$$

$$\Theta_{[2]} = \delta\delta\Theta_{[2]} + O\Big(\delta^4\Big) \,, \tag{10.61}$$

where the expansion is taken around the midpoint frozen NTK Θ_{00} evaluated on the midpoint input $x_{i;0} \equiv (x_{i;+} + x_{i;-})/2$.

For simplicity of our presentation, let's now assume that the two inputs have the same norm $\sum_{i=1}^{n_0} x_{i;+}^2 = \sum_{i=1}^{n_0} x_{i;-}^2$, so that $K_{[1]} = 0$ and $\Theta_{[1]} = (\Theta_{++} - \Theta_{--})/2 = 0$. With this simplification, plugging the decomposition (10.58) and then the expansions (10.59) and (10.61) into the expression for our robustness measure (10.57), we get

$$\frac{\Theta_{-+}}{\Theta_{++}} - 1 = \left(\frac{\Theta_{[0]} - \Theta_{[2]}}{\Theta_{[0]} + \Theta_{[2]}} - 1\right) = -2\frac{\delta\delta\Theta_{[2]}}{\Theta_{00}} + O\Big(\delta^4\Big) \,. \tag{10.62}$$

Thus, we see that the ratio $\delta\delta\Theta_{[2]}/\Theta_{00}$ captures the robustness of predictions for nearby test inputs. We'll analyze its depth dependence for two universality classes shortly.

Having covered the bias term in the generalization error, let's next consider the variance term. The loss-independent piece of the variance is given by the covariance of the generalized posterior distribution (10.41). Evaluating (10.41) for a single training sample $\tilde{\alpha} = +$, using our decompositions for the kernel (5.15) and frozen NTK (10.58), and then using expansions (5.22), (5.24), (10.59), and (10.61), we find

$$\begin{aligned}
&\mathrm{Cov}\Big[z_{i;-}^{(L)}(T),\, z_{i;-}^{(L)}(T)\Big] \qquad\qquad (10.63)\\
=&K_{--} - 2\frac{\Theta_{-+}}{\Theta_{++}}K_{-+} + \left(\frac{\Theta_{-+}}{\Theta_{++}}\right)^2 K_{++}\\
=&K_{[0]} + K_{[2]} - 2\left(\frac{\Theta_{[0]} - \Theta_{[2]}}{\Theta_{[0]} + \Theta_{[2]}}\right)\Big(K_{[0]} - K_{[2]}\Big) + \left(\frac{\Theta_{[0]} - \Theta_{[2]}}{\Theta_{[0]} + \Theta_{[2]}}\right)^2\Big(K_{[0]} + K_{[2]}\Big)\\
=&4\delta\delta K_{[2]} + O\Big(\delta^4\Big) \,.
\end{aligned}$$

Thus, to leading order the variance term depends only on the perpendicular perturbation of the kernel $\delta\delta K_{[2]}$.

[29] The following applies to smooth activation functions and is intended to give the general picture. We will give an analysis particular to nonlinear scale-invariant activation functions later when discussing them in particular.

At this point, we know everything there is to know about how $\delta\delta K_{[2]}$ behaves as a function of depth for our universality classes (cf. §5.3 and §5.5). On the one hand, we could pick initialization hyperparameters such that $\delta\delta K_{[2]}$ grows exponentially with depth. However, with this choice the variance term will grow very quickly, leading to large fluctuations in model predictions between different realizations. On the other hand, we could pick initialization hyperparameters that decay exponentially with depth, leading to a quickly vanishing variance term and very overconfident predictions. However, we will soon see that this overconfidence comes at a cost: an exponentially vanishing perpendicular perturbation $\delta\delta K_{[2]}$ implies an exponentially vanishing frozen NTK component $\delta\delta\Theta_{[2]}$ and thus a vanishing robustness measure (10.62) signaling extreme inflexibility. In particular, we will have learned a constant function that's always equal to y_+, regardless of the input.

This is precisely the generalized bias-variance tradeoff that we described above: if we try to set the variance to zero by having $\delta\delta K_{[2]}$ vanish exponentially, then the vanishing of $\delta\delta\Theta_{[2]}$ will cause our bias to be larger for generic inputs and consequently the network will not be able to generalize in a nontrivial manner. Vice versa, making the function too flexible with large $\delta\delta\Theta_{[2]}$ will make the model predictions not only too sensitive to small changes in the input through $\delta\delta\Theta_{[2]}$, but also will cause large fluctuations in that prediction from realization to realization through $\delta\delta K_{[2]}$.

Of course, we know that there's a third option: we could pick our criticality condition $\chi_\perp(K^\star) = 1$. This setting of the initialization hyperparameters has the potential to balance the bias-variance tradeoff, leading to the best outcome without a priori knowing anything more about the underling dataset we're trying to model.

What about our other criticality condition $\chi_\parallel(K^\star) = 1$? Recall from our discussion of the exploding and vanishing gradient problem in §9.4 that the parallel susceptibility $\chi_\parallel$ affects the way in which the midpoint frozen NTK Θ_{00} receives contributions from different layers. As the midpoint frozen NTK Θ_{00} enters in the robustness measure as in (10.62), this suggests that it also plays an important role in generalization. In fact, we will see soon in §10.4 that ensuring equal contributions from all layers is another way of saying that we use the greatest set of features available to us in making a prediction. Thus, it stands to reason that also picking the criticality condition $\chi_\parallel(K^\star) = 1$ in conjunction with the condition $\chi_\perp(K^\star) = 1$ is a natural choice for generalization, in

addition to all our other evidence for such a choice.[30]

Now, returning to the bias part of the generalization error, to complete our analysis we'll need to solve a recursion for the $\delta\delta\Theta_{[2]}$ component of the frozen NTK recursion (9.5), reprinted here in full:

$$\Theta^{(\ell+1)}_{\delta_1\delta_2} = \lambda_b^{(\ell+1)} + \lambda_W^{(\ell+1)} \left\langle \sigma_{\delta_1}\sigma_{\delta_2} \right\rangle_{K^{(\ell)}} + C_W \left\langle \sigma'_{\delta_1}\sigma'_{\delta_2} \right\rangle_{K^{(\ell)}} \Theta^{(\ell)}_{\delta_1\delta_2} \,. \tag{10.67}$$

Let's first work this out for the $K^\star = 0$ universality class, and then we'll consider the scale-invariant universality class for which we'll need to make use of our finite-angle results from §5.5. Either way, this should be child's play for us at this point.[31]

$K^\star = 0$ Universality Class

Recall (5.44) from much much earlier describing the decomposition of the Gaussian expectation of two activations in the $\gamma^{[a]}$ basis. With the parallel perturbation turned

[30] Just like in footnote 23 of §6.3.1, additional justification comes from the consideration of two inputs with unequal norms: $\sum_{i=1}^{n_0} x_{i;+}^2 \neq \sum_{i=1}^{n_0} x_{i;-}^2$. In such a case, the robustness measure (10.62) is given by

$$\frac{\Theta_{-+}}{\Theta_{++}} - 1 = -\frac{\delta\Theta_{[1]}}{\Theta_{00}} - 2\frac{\delta\delta\Theta_{[2]}}{\Theta_{00}} + \left(\frac{\delta\Theta_{[1]}}{\Theta_{00}}\right)^2 + O\big(\delta^3\big) \,, \tag{10.64}$$

and the covariance is given by

$$\mathrm{Cov}\Big[z^{(L)}_{i;-}(T),\, z^{(L)}_{i;-}(T)\Big] = 4\delta\delta K_{[2]} - 2\delta K_{[1]}\frac{\delta\Theta_{[1]}}{\Theta_{00}} + K_{00}\left(\frac{\delta\Theta_{[1]}}{\Theta_{00}}\right)^2 + O\big(\delta^3\big) \,. \tag{10.65}$$

First, we see that the kernel components $\delta K_{[1]}$ and K_{00} both contribute, necessitating that we set $\chi_\parallel = 1$ as per our previous discussions. In addition, we will also need to tame the exploding and vanishing problem of $\delta\Theta_{[1]}$.

For the scale-invariant universality class, $\Theta_{[1]} = (\Theta_{++} - \Theta_{--})/2$ has exactly the same depth dependence as the single-input frozen NTK (9.44). In this case, $\chi_\parallel = \chi_\perp \equiv \chi$, and all the exponential explosions and vanishments are mitigated by setting $\chi = 1$.

For the $K^\star = 0$ universality class, we can write a recursion for $\delta\Theta_{[1]}$ by projecting out the $\gamma^{[1]}$ component of the full frozen NTK recursion (10.67) using (5.20):

$$\delta\Theta^{(\ell+1)}_{[1]} = \chi_\perp^{(\ell)}\delta\Theta^{(\ell)}_{[1]} + \left(\frac{\lambda_W^{(\ell+1)}}{C_W}\chi_\parallel^{(\ell)} + \frac{C_W}{K_{00}^{(\ell)}}\left\langle z\sigma'\sigma'' \right\rangle_{K_{00}^{(\ell)}} \Theta_{00}^{(\ell)}\right)\delta K^{(\ell)}_{[1]} \,, \tag{10.66}$$

i.e. with a derivation almost isomorphic to the one below for $\delta\delta\Theta_{[2]}$ (10.70). We in particular see that $\delta K^{(\ell)}_{[1]}$ contributes to $\delta\Theta^{(\ell)}_{[1]}$ – which can be thought of as the Bayesian contribution per our last discussion in §10.2.4 – and its exploding and vanishing problem is mitigated by setting $\chi_\parallel(K^\star) = 1$: cf. (5.47). (At this point you may find it useful to re-read and re-flect on the last paragraph of footnote 23 in §6.3.1.) You can further study the depth dependence of $\delta\Theta^{(\ell)}_{[1]}$ at criticality and find that $\delta\Theta^{(\ell)}_{[1]}$ decays faster than $\Theta^{(\ell)}_{00}$ and $\delta\delta\Theta^{(\ell)}_{[2]}$, thus reducing the problem back to the one studied in the main text.

[31] An even more childish play would be studying the Bayesian version of generalization error by setting the training hyperparameters according to (10.46) and (10.48), such that $\Theta^{(L)}_{\delta_1\delta_2} = K^{(L)}_{\delta_1\delta_2}$. In this case, we know exactly how the bias and variance terms of the generalization error behave. This is a very particular setting of the training hyperparameters and unlikely to be optimal in general (cf. our discussion of the differences between the frozen NTK and Bayesian kernel in terms of feature functions in §10.4). Indeed for the scale-invariant universality class, we'll explicitly see around (10.94) that exact Bayesian inference has inferior asymptotic behavior than the more general gradient-based learning.

off, $K^{(\ell)}_{[1]} = 0$, this expansion reads

$$\langle \sigma_{\delta_1} \sigma_{\delta_2} \rangle_{K^{(\ell)}} = \left[\langle \sigma\sigma \rangle_{K^{(\ell)}_{00}} + O\left(\delta^2\right) \right] \gamma^{[0]}_{\delta_1\delta_2} + \left[\delta\delta K^{(\ell)}_{[2]} \langle \sigma'\sigma' \rangle_{K^{(\ell)}_{00}} + O\left(\delta^4\right) \right] \gamma^{[2]}_{\delta_1\delta_2} . \quad (10.68)$$

With a replacement $\sigma \to \sigma'$, we have a similar decomposition for the Gaussian expectation of the derivatives of activations:

$$\langle \sigma'_{\delta_1} \sigma'_{\delta_2} \rangle_{K^{(\ell)}} = \left[\langle \sigma'\sigma' \rangle_{K^{(\ell)}_{00}} + O\left(\delta^2\right) \right] \gamma^{[0]}_{\delta_1\delta_2} + \left[\delta\delta K^{(\ell)}_{[2]} \langle \sigma''\sigma'' \rangle_{K^{(\ell)}_{00}} + O\left(\delta^4\right) \right] \gamma^{[2]}_{\delta_1\delta_2} . \quad (10.69)$$

Plugging these expansions into the full frozen NTK recursion (10.67) and using the component-wise identities $\gamma^{[0]}_{\delta_1\delta_2}\gamma^{[0]}_{\delta_1\delta_2} = \gamma^{[2]}_{\delta_1\delta_2}\gamma^{[2]}_{\delta_1\delta_2} = \gamma^{[0]}_{\delta_1\delta_2}$ and $\gamma^{[0]}_{\delta_1\delta_2}\gamma^{[2]}_{\delta_1\delta_2} = \gamma^{[2]}_{\delta_1\delta_2}$, we get

$$\delta\delta\Theta^{(\ell+1)}_{[2]} = \chi^{(\ell)}_{\perp} \delta\delta\Theta^{(\ell)}_{[2]} + \left(\frac{\lambda^{(\ell+1)}_W}{C_W} \chi^{(\ell)}_{\perp} + C_W \langle \sigma''\sigma'' \rangle_{K^{(\ell)}_{00}} \Theta^{(\ell)}_{00} \right) \delta\delta K^{(\ell)}_{[2]} , \quad (10.70)$$

where we've recalled the definition of the perpendicular susceptibility (5.51), $\chi^{(\ell)}_{\perp} = C_W \langle \sigma'\sigma' \rangle_{K^{(\ell)}_{00}}$.[32]

We learned long ago that the perpendicular susceptibility governs the behavior of the perpendicular perturbation $\delta\delta K^{(\ell)}_{[2]}$, and we see from (10.70) that it also controls the behavior of the frozen NTK component $\delta\delta\Theta^{(\ell)}_{[2]}$. As we alluded to before, the exponential decay/growth of $\delta\delta K^{(\ell)}_{[2]}$ and $\delta\delta\Theta^{(\ell)}_{[2]}$ are thusly linked. In particular, trying to eliminate the variance term of the generalization error by letting $\delta\delta K^{(\ell)}_{[2]}$ exponentially decay will also cause $\delta\delta\Theta^{(\ell)}_{[2]}$ to exponentially decay, making the model prediction constant, inflexible, and highly biased.

Given this and our previous discussion on the role of the parallel susceptibility $\chi_{\parallel}$, let's now tune to criticality $\chi_{\parallel}(K^\star) = \chi_{\perp}(K^\star) = 1$ and evaluate the depth dependence of $\delta\delta\Theta^{(\ell)}_{[2]}$. For the $K^\star = 0$ universality class, criticality (5.90) is found by tuning $C_b = 0$ and $C_W = \frac{1}{\sigma_1^2}$. With these settings, we recall the large-ℓ asymptotic solutions from (5.92) and (5.99)

$$K^{(\ell)}_{00} = \left[\frac{1}{(-a_1)} \right] \frac{1}{\ell} + \ldots , \qquad \delta\delta K^{(\ell)}_{[2]} = \frac{\delta^2}{\ell^{p_\perp}} + \ldots , \quad (10.71)$$

where $p_\perp \equiv b_1/a_1$, and the activation-function dependent constants a_1 and b_1 were defined in (5.86) and (5.88), and δ^2 is a constant related to the initial separation of the inputs but isn't fixed by the asymptotic analysis. Also recall from the more recent past (9.76) that we can asymptotically expand the perpendicular susceptibility as

$$\chi^{(\ell)}_{\perp} = 1 - \frac{p_\perp}{\ell} + \ldots . \quad (10.72)$$

[32] More generally there are terms proportional to $\left(\delta K_{[1]}\right)^2$ and $\delta K_{[1]}\delta\Theta_{[1]}$ in this recursion for $\delta\delta\Theta_{[2]}$; however, when training and test inputs have equal norm, $\delta K_{[1]} = 0$, these terms vanish.

Similarly, by a simple Gaussian integral we can evaluate the following Gaussian expectation,

$$\left\langle \sigma''\sigma'' \right\rangle_{K_{00}^{(\ell)}} = \sigma_2^2 + O\Big(K_{00}^{(\ell)}\Big) = \sigma_2^2 + O\left(\frac{1}{\ell}\right) , \tag{10.73}$$

remembering our notation $\sigma_2 \equiv \sigma''(0)$.

Next, we also have to make a choice about the training hyperparameters $\lambda_b^{(\ell)}$ and $\lambda_W^{(\ell)}$. Indeed, the depth scaling of the generalization error will depend on these hyperparameters, a fact that should not be surprising: we expect the selection of our relative learning rates to affect the performance of our model. Let's first follow the guidance of §9.4 where we discussed an *equivalence principle* for learning rates, and set these training hyperparameters according to (9.95), i.e. (9.70) multiplied by $L^{p_\perp - 1}$:

$$\lambda_b^{(\ell)} = \widetilde{\lambda}_b \left(\frac{1}{\ell}\right)^{p_\perp} L^{p_\perp - 1} , \qquad \lambda_W^{(\ell)} = \widetilde{\lambda}_W \left(\frac{L}{\ell}\right)^{p_\perp - 1} . \tag{10.74}$$

With such a choice, we have an asymptotic solution for the midpoint frozen NTK, which is the same solution as in (9.71) up to a multiplication by $L^{p_\perp - 1}$:

$$\Theta_{00}^{(\ell)} = \left[\widetilde{\lambda}_b + \frac{\widetilde{\lambda}_W \sigma_1^2}{(-a_1)}\right] \left(\frac{L}{\ell}\right)^{p_\perp - 1} + \ldots . \tag{10.75}$$

Further plugging these results (10.71)–(10.75) into (10.70), we get

$$\begin{aligned} \delta\delta\Theta_{[2]}^{(\ell+1)} = &\left[1 - \frac{p_\perp}{\ell} + \ldots\right] \delta\delta\Theta_{[2]}^{(\ell)} \\ &+ \delta^2 \left\{ \widetilde{\lambda}_W \sigma_1^2 + \frac{\sigma_2^2}{\sigma_1^2} \left[\widetilde{\lambda}_b + \frac{\widetilde{\lambda}_W \sigma_1^2}{(-a_1)}\right] \right\} L^{p_\perp - 1} \left(\frac{1}{\ell}\right)^{2p_\perp - 1} + \ldots . \end{aligned} \tag{10.76}$$

With our usual methods, we can solve this recursion in the asymptotically large-ℓ limit with

$$\delta\delta\Theta_{[2]}^{(\ell)} = \delta^2 \frac{L^{p_\perp - 1}}{(2 - p_\perp)} \left\{ \widetilde{\lambda}_W \sigma_1^2 + \frac{\sigma_2^2}{\sigma_1^2} \left[\widetilde{\lambda}_b + \frac{\widetilde{\lambda}_W \sigma_1^2}{(-a_1)}\right] \right\} \left(\frac{1}{\ell}\right)^{2p_\perp - 2} + \ldots . \tag{10.77}$$

Finally, taking the ratio of the midpoint frozen NTK (10.75) and the perpendicular perturbation (10.77) and evaluating at the output layer $\ell = L$, we find the overall network depth dependence for our robustness measure:

$$\frac{-2\delta\delta\Theta_{[2]}^{(L)}}{\Theta_{00}^{(L)}} = \frac{2\delta^2}{(p_\perp - 2)} \left\{ \frac{\widetilde{\lambda}_W \sigma_1^2}{\left[\widetilde{\lambda}_b + (\widetilde{\lambda}_W \sigma_1^2)/(-a_1)\right]} + \frac{\sigma_2^2}{\sigma_1^2} \right\} L^{1 - p_\perp} \propto L^{1 - p_\perp} . \tag{10.78}$$

This is astonishing: the desire to keep the robustness measure of order one for very deep networks exactly picks out activation functions in this universality class with $p_\perp = 1$![33]

[33] Since $p_\perp = b_1/a_1$, cf. (5.99), $b_1 \geq a_1$, cf. (5.86) and (5.88), and $a_1 < 0$, cf. (5.92), these overall imply that $p_\perp \leq 1$ for any $K^\star = 0$ activation function. In particular, for a non-odd activation function with $\sigma_2 \neq 0$, the exponent for perpendicular perturbations is strictly less than one, $p_\perp < 1$, and thus the bias term in the generalization error (10.56) will grow with network depth L.

Such a condition is satisfied by any odd $K^\star = 0$ activation function, such as `tanh` and `sin`, both of which we've been discussing prominently throughout the book.

Now, let's zoom out for a moment and reflect on these calculations more broadly. Somewhat miraculously, the theoretically-motivated tuning of all our hyperparameters – *criticality* for the initialization hyperparameters and the learning rate *equivalence principle* for the training hyperparameters – has led to the most practically-optimal solution for the generalization error in this one-training-one-test setting, keeping the bias-variance tradeoff in check. Even more importantly, these choices and solutions are robust across many different network widths and depths, making them quite useful for experimentation and the scaling up of models.[34] Of course, there really was no miracle: our theoretical principles were practically motivated from the start.

Now that we understand what we *should* do, let's discuss a different choice of training hyperparameters that we *should not* make. Had we not followed the learning rate equivalence principle, perhaps we would have just made both weight and bias learning rates layer independent as $\lambda_b^{(\ell)} = \lambda_b$ and $\lambda_W^{(\ell)} = \lambda_W$. Let's see what happens then, specializing to odd activation functions with $\sigma_2 = 0$ and $p_\perp = 1$ for simplicity. In this case, our general formal solution for the single-input frozen NTK (9.69) reduces to

$$\Theta_{00}^{(\ell)} = \left(\frac{\lambda_b}{2}\right)\ell + \ldots \,, \tag{10.79}$$

with a linear dependence on the layer ℓ, while the same calculation as above with $\sigma_2 = 0$ and $p_\perp = 1$ in mind gives a layer-independent constant asymptotic solution for $\delta\delta\Theta_{[2]}^{(\ell)}$:

$$\delta\delta\Theta_{[2]}^{(\ell)} = \lambda_W \sigma_1^2 \delta^2 + \ldots \,. \tag{10.80}$$

Combined, our robustness measure becomes

$$\frac{-2\delta\delta\Theta_{[2]}^{(L)}}{\Theta_{00}^{(L)}} = \left[\frac{-4\lambda_W \sigma_1^2 \delta^2}{\lambda_b}\right]\frac{1}{L} + \ldots \,, \tag{10.81}$$

which is slowly but surely decaying with the overall depth L of the network. (The consideration of more general activation functions with $p_\perp \neq 1$ doesn't change this conclusion.) Therefore, this choice of the training hyperparameters is polynomially suboptimal compared to the choice based on our equivalence principle.[35]

Reflecting back, when we first discussed the learning rate equivalence principle by staring at our formal solution (9.69), we were motivated by the desire to ensure equal contributions to the NTK from each layer. Then in §9.4 we realized that such choices solve a polynomial version of the exploding and vanishing gradient problem. Here we see

[34] These tunings are more or less still valid even as we relax the infinite-width requirement to allow nonzero aspect ratio, $L/n \ll 1$.

[35] We leave it to the reader to see how disastrous things would be – in terms of our one-training-one-test generalization error – if we had decided not to rescale the weight learning rate by the widths of the previous layer as in (8.6), leading to an even more extreme violation of the learning rate equivalence principle.

the downstream consequences of those choices through the lens of generalization error, giving a solid support for the equivalence principle according to our quantitative measure of training success.

Scale-Invariant Universality Class

To analyze scale-invariant activation functions, we need to use results from our finite-angle analysis in §5.5. In particular, the Gaussian expectation $\langle \sigma'' \sigma'' \rangle$ that appeared in the δ expansion of $\langle \sigma' \sigma' \rangle$ in (10.69) is singular for nonlinear scale-invariant functions due to the kink at the origin, and we promised we'd have to recall our finite-angle results when such a singularity occurs.

Keeping our promise to you, let's recall a bunch of things from that section. First, we decomposed the two-input kernel matrix as (5.146)

$$K^{(\ell)}_{\delta_1\delta_2} = \begin{pmatrix} K^{(\ell)}_{++} & K^{(\ell)}_{+-} \\ K^{(\ell)}_{-+} & K^{(\ell)}_{--} \end{pmatrix} = K^{(\ell)}_d \begin{pmatrix} 1 & \cos\left(\psi^{(\ell)}\right) \\ \cos\left(\psi^{(\ell)}\right) & 1 \end{pmatrix} , \qquad \psi^{(\ell)} \in [0, \pi] , \tag{10.82}$$

with two dynamical variables being the diagonal kernel $K^{(\ell)}_d$ and the polar angle $\psi^{(\ell)}$. With this parametrization in mind, let us reprint a bunch of the previous results that we'll need, (5.60), (5.62), (5.159), and (5.161):

$$\langle \sigma_+ \sigma_+ \rangle_{K^{(\ell)}} = \langle \sigma_- \sigma_- \rangle_{K^{(\ell)}} = A_2 K^{(\ell)}_d , \tag{10.83}$$

$$C_W \langle \sigma'_+ \sigma'_+ \rangle_{K^{(\ell)}} = C_W \langle \sigma'_- \sigma'_- \rangle_{K^{(\ell)}} = C_W A_2 \equiv \chi , \tag{10.84}$$

$$\langle \sigma_+ \sigma_- \rangle_{K^{(\ell)}} = A_2 K^{(\ell)}_d \left\{ \cos\left(\psi^{(\ell)}\right) + \rho \left[\sin\left(\psi^{(\ell)}\right) - \psi^{(\ell)} \cos\left(\psi^{(\ell)}\right) \right] \right\} , \tag{10.85}$$

$$C_W \langle \sigma'_+ \sigma'_- \rangle_{K^{(\ell)}} = \chi (1 - \rho \psi^{(\ell)}) , \tag{10.86}$$

where $A_2 \equiv (a_+^2 + a_-^2)/2$, $\rho \equiv \frac{1}{\pi} \frac{(a_+ - a_-)^2}{(a_+^2 + a_-^2)}$, and a_+ and a_- are the two constants that define the particular activation function (though by now you know that by heart).

Let us now make a similar decomposition for the frozen NTK as

$$\Theta^{(\ell)}_{\delta_1\delta_2} = \begin{pmatrix} \Theta^{(\ell)}_{++} & \Theta^{(\ell)}_{+-} \\ \Theta^{(\ell)}_{-+} & \Theta^{(\ell)}_{--} \end{pmatrix} = \Theta^{(\ell)}_d \begin{pmatrix} 1 & \cos\left(\zeta^{(\ell)}\right) \\ \cos\left(\zeta^{(\ell)}\right) & 1 \end{pmatrix} , \qquad \zeta^{(\ell)} \in [0, \pi] , \tag{10.87}$$

with a diagonal frozen NTK $\Theta^{(\ell)}_d$ and another polar angle $\zeta^{(\ell)}$.

Then, plugging this decomposition (10.87) and recollected results (10.83)–(10.86) into the frozen NTK recursion (10.67), we get coupled recursions for the frozen NTK, casted in our finite-angle parameterization:

$$\Theta^{(\ell+1)}_d = \chi \Theta^{(\ell)}_d + \lambda_b^{(\ell+1)} + \lambda_W^{(\ell+1)} A_2 K^{(\ell)}_d , \tag{10.88}$$

$$\begin{aligned} \Theta^{(\ell+1)}_d \cos\left(\zeta^{(\ell+1)}\right) =& \chi (1 - \rho \psi^{(\ell)}) \Theta^{(\ell)}_d \cos\left(\zeta^{(\ell)}\right) \\ &+ \lambda_b^{(\ell+1)} + \lambda_W^{(\ell+1)} A_2 K^{(\ell)}_d \left\{ \cos\left(\psi^{(\ell)}\right) + \rho \left[\sin\left(\psi^{(\ell)}\right) - \psi^{(\ell)} \cos\left(\psi^{(\ell)}\right) \right] \right\} . \end{aligned} \tag{10.89}$$

We see here in the off-diagonal recursion (10.89) a finite-angle analog of what we saw perturbatively for the $K^\star = 0$ universality class in the infinitesimal-angle recursion (10.70): the polar angle for the kernel $\psi^{(\ell)}$ sources the finite angle for the frozen NTK $\zeta^{(\ell)}$. Said another way, the exponential growth and decay of the kernel angle $\psi^{(\ell)}$ – at least for small enough angle – are linked to the exponential growth and decay of the frozen-NTK angle $\zeta^{(\ell)}$, which are in turn linked to the generalized bias-variance tradeoff.

With that chain of links in mind (as well as parallel discussions of similar issues in almost every other chapter of this book), it's natural that we should set our initialization hyperparameters by tuning to criticality: $\chi = 1$. With this choice, we recall the critical solutions from our finite-angle analysis of the kernel in §5.5:

$$K_d^{(\ell)} = K_d^\star\,, \qquad \psi^{(\ell)} = \left(\frac{3}{\rho}\right)\frac{1}{\ell} + \ldots\,, \tag{10.90}$$

where $K_d^\star$ is exactly constant, set by the first layer. Additionally, having already made the case for the learning rate equivalence principle when discussing the $K^\star = 0$ universality class, let's just simplify our discussion here by setting training hyperparameters according to that equivalence principle for scale-invariant activations (9.94): $\lambda_b^{(\ell)} = \widetilde{\lambda}_b/L$ and $\lambda_W^{(\ell)} = \widetilde{\lambda}_W/L$. With this choice, we see that

$$\Theta_d^{(\ell)} = \left(\widetilde{\lambda}_b + \widetilde{\lambda}_W A_2 K_d^\star\right)\frac{\ell}{L} \tag{10.91}$$

solves the recursion for the diagonal frozen NTK (10.88) with the initial condition (9.7). Importantly, here ℓ refers to a particular layer of the network, while L is the overall network depth.[36]

Plugging our choice of learning rates, our kernel solution (10.90), and NTK solution (10.91) into the finite-angle recursion (10.89), we get after a bit of rearranging

$$\cos\Bigl(\zeta^{(\ell+1)}\Bigr) = \left(1 - \frac{4}{\ell} + \ldots\right)\cos\Bigl(\zeta^{(\ell)}\Bigr) + \left(\frac{1}{\ell} + \ldots\right)\,, \tag{10.92}$$

which we can see easily is solved by an everything-independent constant

$$\cos\Bigl(\zeta^{(\ell)}\Bigr) = \frac{1}{4} + \ldots\,. \tag{10.93}$$

Thus, our robustness measure (10.57) in the bias term of generalization error for nonlinear scale-invariant activation functions is given by a simple order-one number:

$$\frac{\Theta_{-+}^{(L)}}{\Theta_{++}^{(L)}} - 1 = \cos\Bigl(\zeta^{(L)}\Bigr) - 1 = -\frac{3}{4} + \ldots\,. \tag{10.94}$$

Similarly, given that the nearby-input analysis can break down for nonlinear scale-invariant activations, let's use our finite-angle analysis here to also work out the variance

[36] The frozen NTK solution (10.91) is identical to our previous single-input solution (9.44), here we have just rescaled the bias and weight learning rates by the overall depth, $\lambda_b = \widetilde{\lambda}_b/L$ and $\lambda_W = \widetilde{\lambda}_W/L$, as required by the equivalence principle (9.94).

term of the generalization error (10.63); plugging in the asymptotic falloff for the kernel (5.167) and (5.168) as well as using (10.94) for the frozen NTK, we get

$$\begin{aligned}\text{Cov}\Big[z_{i;-}^{(L)}(T),\, z_{i;-}^{(L)}(T)\Big] &= K_{--} - 2\frac{\Theta_{-+}}{\Theta_{++}}K_{-+} + \left(\frac{\Theta_{-+}}{\Theta_{++}}\right)^2 K_{++} \\ &= K_d^\star \left[1 - \cos\Big(\zeta^{(L)}\Big)\right]^2 = \frac{9}{16}K_d^\star + \ldots \,. \end{aligned} \tag{10.95}$$

Unlike the previous case for $K^\star = 0$ activations, these asymptotic results for the generalization error, (10.94) and (10.95), don't depend on the training hyperparameters $\widetilde{\lambda}_b$ and $\widetilde{\lambda}_W$, nor do they depend on a constant like δ^2 that knows about the separation of the test and training points. (However, just as we discussed for $\psi^{(\ell)}$ in §5.5, the depth at which these asymptotic results become valid does depend on δ^2, the input norm, the activation function, and the training hyperparameters.) Nonetheless, again with the correct tuning of our hyperparameters based on the principles of criticality and equivalence, we found a constant bias and variance, giving us the best possible tradeoff when training deep networks with nonlinear scale-invariant activations.[37]

Let us end with the special remark on deep linear networks. For these networks, we use the `linear` activation function with $a_+ = a_-$ and hence have $\rho = 0$. In particular, we saw in §5.5 that not only was the diagonal kernel preserved at criticality, but the polar angle was preserved as well: $K_d^{(\ell)} = K_d^\star$ and $\psi^{(\ell)} = \psi^\star$. Noting this, the off-diagonal recursion for the frozen NTK (10.89) then becomes

$$\Theta_d^{(\ell+1)} \cos\Big(\zeta^{(\ell+1)}\Big) = \Theta_d^{(\ell)} \cos\Big(\zeta^{(\ell)}\Big) + \frac{\widetilde{\lambda}_b}{L} + \frac{\widetilde{\lambda}_W}{L} A_2 K_d^\star \cos(\psi^\star) \,. \tag{10.96}$$

This recursion is exactly solved by

$$\Theta_d^{(\ell)} \cos\Big(\zeta^{(\ell)}\Big) = \Big[\widetilde{\lambda}_b + \widetilde{\lambda}_W A_2 K_d^\star \cos(\psi^\star)\Big] \frac{(\ell-1)}{L} + \Theta_d^{(1)} \cos\Big(\zeta^{(1)}\Big) \,. \tag{10.97}$$

Dividing this result by our solution for the diagonal frozen NTK (10.91) then gives

$$\cos\Big(\zeta^{(\ell)}\Big) = \frac{\widetilde{\lambda}_b + \widetilde{\lambda}_W A_2 K_d^\star \cos(\psi^\star)}{\widetilde{\lambda}_b + \widetilde{\lambda}_W A_2 K_d^\star} + \ldots \,, \tag{10.98}$$

which polynomially asymptotes to a constant. Unlike the case for the nonlinear scale-invariant activation functions, this constant depends on the observables in the first layer in a rather detailed manner, naturally connecting the generalization properties of the network to the input. The real limitation of the deep linear networks becomes immediately apparent upon considering their (in)ability to interpolate/extrapolate, which we'll analyze next for linearly and nonlinearly activated MLPs.

[37] It is worth noting what happens with the special case of exact Bayesian inference where the only nonzero learning rates are in the last layer in order to set $\Theta^{(L)} = K^{(L)}$. In that case, the robustness measure is given by $\cos\big(\psi^{(L)}\big) - 1 = O\big(1/\ell^2\big)$. Given this decay with depth, we see that the restricted Bayesian case is clearly inferior to an ensemble of networks that are fully-trained via gradient descent with uniform learning rates across layers.

10.3.2 Interpolation and Extrapolation

Rather than focusing primarily on our evaluation criteria for successful training, the generalization error, in this subsection we will focus more on the kinds of functions that our trained neural networks actually compute. This analysis will enable us to consider the *inductive bias* of different activation functions and tell us how to relate the properties of those activation functions to the properties of the dataset and function that we're trying to approximate.

In the previous subsection we asked: given the true output $y_{i;+}$ for an input $x_{i;+}$, what does a fully-trained MLP in the infinite-width limit predict for the output of a nearby input $x_{i;-}$? Here, we up the ante and ask: given the true outputs $y_{i;\pm}$ for *two* inputs $x_{i;\pm} = x_{i;0} \pm \frac{\delta x_i}{2}$, what is the prediction for a one-parameter family of test inputs,

$$s x_{i;+} + (1-s) x_{i;-} = x_{i;0} + \frac{(2s-1)}{2} \delta x_i \equiv x_{i;(2s-1)} \,, \tag{10.99}$$

that sit on a line passing through $x_{i;+}$ and $x_{i;-}$? When our parameter s is inside the unit interval $s \in [0,1]$, this is a question about neural-network **interpolation**; for s outside the unit interval, it's a question about **extrapolation**. For general s, let's refer to this collectively as $*$**-polation**.

First we'll perform a little exercise in $*$-polation with deep linear networks to see what networks with `linear` activation functions do. Accordingly, we'll see concretely how deep linear networks approximate a very limited set of functions. Then we'll follow up by assessing smooth nonlinear networks.

Linear $*$-Polation by Deep Linear Networks

There's a very simple way to see how $*$-polation works for deep linear networks. If we recall for a moment (and for one last time) the forward equation for deep linear networks (3.1),

$$z^{(\ell+1)}_{i;\alpha} = b^{(\ell+1)}_i + \sum_{j=1}^{n_\ell} W^{(\ell+1)}_{ij} z^{(\ell)}_{j;\alpha} \,, \tag{10.100}$$

with $z^{(0)}_{j;\alpha} \equiv x_{j;\alpha}$, it's clear that the linear structure in the input (10.99) will be preserved from layer to layer. That is, given an ℓ-th-layer preactivations of the form

$$z^{(\ell)}_{i;(2s-1)} = s z^{(\ell)}_{i;+} + (1-s) z^{(\ell)}_{i;-} \tag{10.101}$$

that has such a linear structure, we then have for the next layer

$$\begin{aligned} z^{(\ell+1)}_{i;(2s-1)} &= b^{(\ell+1)}_i + \sum_{j=1}^{n_\ell} W^{(\ell+1)}_{ij} \left[s z^{(\ell)}_{j;+} + (1-s) z^{(\ell)}_{j;-} \right] \\ &= s \left(b^{(\ell+1)}_i + \sum_{j=1}^{n_\ell} W^{(\ell+1)}_{ij} z^{(\ell)}_{j;+} \right) + (1-s) \left(b^{(\ell+1)}_i + \sum_{j=1}^{n_\ell} W^{(\ell+1)}_{ij} z^{(\ell)}_{j;-} \right) \\ &= s z^{(\ell+1)}_{i;+} + (1-s) z^{(\ell+1)}_{i;-} \,, \end{aligned} \tag{10.102}$$

which still respects the linear structure. This is just a direct consequence of the fact that deep linear networks compute linear functions of their input.

Therefore, for a test input that's a linear sum of our two training points (10.99), the network will output the linear sum of the network outputs on the two individual training points:

$$z^{(L)}_{i;(2s-1)} = s z^{(L)}_{i;+} + (1-s) z^{(L)}_{i;-} \,. \tag{10.103}$$

This equation holds at initialization as well as at the end of training, which means that any particular fully-trained deep linear network will $*$-polate as

$$z^{(L)}_{i;(2s-1)}(T) = s y_{i;+} + (1-s) y_{i;-} \,, \tag{10.104}$$

since the fully-trained network output will equal the true output for any element in the training set: $z^{(L)}_{i;\pm}(T) = y_{i;\pm}$. With this, we see that fully-trained deep linear network *linearly* $*$-polate, no matter what. This is both intuitive and pretty obvious; as deep linear networks perform linear transformations they can only compute linear functions.

Of course, this is exactly what we said when we studied deep linear networks way back in §3. Here, we explicitly see *why* these networks are limited after training, by showing the (limited) way in which they can use training examples to make predictions. Accordingly, if the function you're trying to approximate is a linear function of the input data, then deep linear networks are a great modeling choice. If the function is nonlinear, we'll have to consider nonlinear activation functions. It's not that deep.

Nonlinear $*$-Polation by Smooth Nonlinear Deep Networks

We'll have to work a little harder to see what nonlinear networks do.[38] In the last section, we saw that the output of a fully-trained network is given by the stochastic kernel prediction equation (10.39), which we reprint here for convenience:

$$z^{(L)}_{i;\dot\beta}(T) = z^{(L)}_{i;\dot\beta} - \sum_{\tilde\alpha_1,\tilde\alpha_2 \in \mathcal{A}} \Theta^{(L)}_{\dot\beta\tilde\alpha_1} \widetilde{\Theta}^{\tilde\alpha_1\tilde\alpha_2}_{(L)} \left(z^{(L)}_{i;\tilde\alpha_2} - y_{i;\tilde\alpha_2} \right) . \tag{10.105}$$

Thus, we see that to study $*$-polation more generally we will need to evaluate elements of the frozen NTK between our test and training set, $\Theta^{(L)}_{(2s-1)\pm}$, and also need to invert the two-by-two submatrix of the frozen NTK on the training set only, $\widetilde{\Theta}^{\tilde\alpha_1\tilde\alpha_2}_{(L)}$.

This latter inversion can be easily completed with the standard textbook formula for the inverse of a two-by-two matrix,

$$\widetilde{\Theta}^{\tilde\alpha_1\tilde\alpha_2} = \frac{1}{\Theta_{++}\Theta_{--} - \Theta^2_{+-}} \begin{pmatrix} \Theta_{--} & -\Theta_{+-} \\ -\Theta_{+-} & \Theta_{++} \end{pmatrix} , \tag{10.106}$$

[38] We would have to work even harder to see what nonlinear scale-invariant activation functions do, so here we'll focus on smooth nonlinear activation functions. For such nonlinear scale-invariant activation functions with kinks, since the $*$-polated input $x_{i;(2s-1)}$ does not have the same norm as $x_{i;\pm}$ for $s \neq 0, 1$, we would need to extend the finite-angle analysis from §5.5 to the case of unequal input norms. This is left as a challenge in pedagogy to future deep-learning book authors.

where here we've also used the symmetry $\Theta_{+-} = \Theta_{-+}$ and further dropped the layer indices. For the rest of this section, we will always assume that these frozen NTKs are evaluated at the output layer.

Next, to compute the off-diagonal elements between the training set and the test set $\Theta_{(2s-1)\pm}$, we'll need to generalize our δ expansion a bit. Let's first recall our expressions for the components of the frozen NTK in $\gamma^{[a]}$ basis, (10.58), and then plug in the δ expansion we performed on the two-by-two submatrix $\widetilde{\Theta}_{\tilde{\alpha}_1\tilde{\alpha}_2}$, (10.59)–(10.61), which gives

$$\Theta_{\pm\pm} = \Theta_{00} \pm \delta\Theta_{[1]} + \Big(\delta\delta\Theta_{[2]} + \delta\delta\Theta_{[0]}\Big) + O\Big(\delta^3\Big) \ , \tag{10.107}$$

$$\Theta_{\pm\mp} = \Theta_{00} + \Big(-\delta\delta\Theta_{[2]} + \delta\delta\Theta_{[0]}\Big) + O\Big(\delta^3\Big) \ , \tag{10.108}$$

for the pair of inputs $x_{i;\pm} = x_{i;0} \pm \frac{\delta x_i}{2}$. Let's now consider a pair of perturbed inputs of a more general form

$$x_{i;\epsilon_1} \equiv x_{i;0} + \frac{\epsilon_1}{2}\delta x_i \, , \qquad x_{i;\epsilon_2} \equiv x_{i;0} + \frac{\epsilon_2}{2}\delta x_i \, . \tag{10.109}$$

Note that picking $\epsilon_{1,2}$ from ± 1 reduces them to $x_{i;\pm}$, while the new case of the interest, $\Theta_{(2s-1)\pm}$, corresponds to setting $\epsilon_1 = (2s-1)$ and $\epsilon_2 = \pm 1$. For a generic pair of inputs (10.109), the δ expansion gets modified as

$$\begin{aligned}\Theta_{\epsilon_1\epsilon_2} =& \Theta_{00} + \left(\frac{\epsilon_1+\epsilon_2}{2}\right)\delta\Theta_{[1]} + \left(\frac{\epsilon_1+\epsilon_2}{2}\right)^2\Big(\delta\delta\Theta_{[2]} + \delta\delta\Theta_{[0]}\Big) \\ &+ \left(\frac{\epsilon_1-\epsilon_2}{2}\right)^2\Big(-\delta\delta\Theta_{[2]} + \delta\delta\Theta_{[0]}\Big) + O\Big(\epsilon^3\delta^3\Big) \ .\end{aligned} \tag{10.110}$$

To see why this is the correct expression, note that *(i)* each term has the right scaling with $\epsilon_{1,2}$, *(ii)* for $\epsilon_1 = \epsilon_2 = \pm 1$ we correctly recover the expression for $\Theta_{\epsilon_1\epsilon_2} = \Theta_{\pm\pm}$ (10.107), *(iii)* for $\epsilon_1 = -\epsilon_2 = \pm 1$, we correctly recover the expression for $\Theta_{\epsilon_1\epsilon_2} = \Theta_{\pm\mp}$ (10.108), and *(iv)* the expression is symmetric under $\epsilon_1 \leftrightarrow \epsilon_2$. The frozen NTK component $\Theta_{\epsilon_1\epsilon_2}$ must satisfy these four constraints, and the expression (10.110) is the unique formula that satisfies them all.

Applying this formula to evaluate $\Theta_{(2s-1)\pm}$ and simplifying a bit, we find

$$\Theta_{(2s-1)\pm} = s\Theta_{\pm+} + (1-s)\Theta_{\pm-} - 2s(1-s)\delta\delta\Theta_{[0]} + O\Big(\delta^3\Big) \ . \tag{10.111}$$

As we'll see, the key to nonlinear $*$-polation, at least for nearby inputs, is in the $\delta\delta\Theta_{[0]}$ term.[39] Firstly, it's clear that this term nonlinearly depends on the test input, as evidenced by $s(1-s)$ prefactor. Indeed, you can go back and check that this term identically vanishes for deep linear networks, i.e., for those networks we simply have

[39] N.B. $\delta\delta\Theta_{[0]}$ is very different from $\delta\delta\Theta_{[2]}$: the former is the second term in the expansion of the $\gamma^{[0]}$ component $\Theta_{[0]}$, cf. (10.59), while the latter is the first term in the expansion of the $\gamma^{[2]}$ component $\Theta_{[2]}$, cf. (10.61).

$\Theta_{(2s-1)\pm} = s\Theta_{\pm+} + (1-s)\Theta_{\pm-}$.[40] With that in mind, it also helps to decompose the initial preactivation into linear and nonlinear pieces as

$$z^{(L)}_{i;(2s-1)} = sz^{(L)}_{i;+} + (1-s)z^{(L)}_{i;-} + \left[z^{(L)}_{i;(2s-1)} - sz^{(L)}_{i;+} - (1-s)z^{(L)}_{i;-}\right] . \tag{10.112}$$

Here, the second term vanishes for deep linear networks, as per (10.103), and so in general it captures nonlinearity of the network output at initialization.

Plugging (10.106), (10.111), and (10.112) into our kernel prediction formula (10.105), we see that our fully-trained prediction on the test input $x_{i;(2s-1)} = sx_{i;+} + (1-s)x_{i;-}$ is given by

$$\begin{aligned} &z^{(L)}_{i;(2s-1)}(T) \\ &= \left[z^{(L)}_{i;(2s-1)} - sz^{(L)}_{i;+} - (1-s)z^{(L)}_{i;-}\right] + \left[sy_{i;+} + (1-s)y_{i;-}\right] \\ &\quad - s(1-s)\left[\frac{2\delta\delta\Theta_{[0]}}{\Theta_{00}\delta\delta\Theta_{[2]} - \delta\Theta^2_{[1]}}\right]\Big[2\delta\delta\Theta_{[2]}\left(z^{(L)}_{i;+} + z^{(L)}_{i;-} + y_{i;+} + y_{i;-}\right) \\ &\qquad\qquad - \delta\Theta_{[1]}\left(z^{(L)}_{i;+} - z^{(L)}_{i;-} + y_{i;+} - y_{i;-}\right)\Big] + O\left(\delta^3\right) . \end{aligned} \tag{10.113}$$

Comparing with our linear $*$-polation formula (10.104), we see that both the first and last terms are new: nonlinear networks can *nonlinearly* $*$-polate! Interestingly, the fully-trained $*$-polation for nonlinear activation functions depends on the network output at initialization through the nonlinearity $z^{(L)}_{i;(2s-1)} - sz^{(L)}_{i;+} - (1-s)z^{(L)}_{i;-}$; in contrast, for deep linear networks the $*$-polation only depended on the true output of the training examples.

As a particular illustration of this formula, consider the case when the two training inputs have the same norm. In such a case $\Theta_{[1]} = 0$, and we find a much simpler formula:

$$\begin{aligned} z^{(L)}_{i;(2s-1)}(T) = &\left[z^{(L)}_{i;(2s-1)} - sz^{(L)}_{i;+} - (1-s)z^{(L)}_{i;-}\right] + \left[sy_{i;+} + (1-s)y_{i;-}\right] \\ &- 4s(1-s)\left(\frac{\delta\delta\Theta_{[0]}}{\Theta_{00}}\right)\left(z^{(L)}_{i;+} + z^{(L)}_{i;-} + y_{i;+} + y_{i;-}\right) + O\left(\delta^3\right) . \end{aligned} \tag{10.114}$$

Averaging over our ensemble, this prediction has a mean

$$m^{\infty}_{i;(2s-1)} = sy_{i;+} + (1-s)y_{i;-} - 4s(1-s)\left(\frac{\delta\delta\Theta_{[0]}}{\Theta_{00}}\right)(y_{i;+} + y_{i;-}) + O\left(\delta^3\right) . \tag{10.115}$$

Here the first term in (10.114) that captured the nonlinearity of the network output at initialization vanished under the expectation, and so the nonlinearity of the $*$-polation mean is entirely captured by the dimensionless ratio $\delta\delta\Theta_{[0]}/\Theta_{00}$.

[40] To see this quickly, note that both the first-layer metric (4.8) and the first-layer NTK (8.23) are bilinear in the two inputs, and that such bilinear structure is preserved under the recursions for deep linear networks: $K^{(\ell+1)}_{\delta_1\delta_2} = C_b^{(\ell+1)} + C_W^{(\ell+1)}K^{(\ell)}_{\delta_1\delta_2}$, cf. (4.118), and $\Theta^{(\ell+1)}_{\delta_1\delta_2} = \lambda_b^{(\ell+1)} + \lambda_W^{(\ell+1)}K^{(\ell)}_{\delta_1\delta_2} + C_W^{(\ell+1)}\Theta^{(\ell)}_{\delta_1\delta_2}$, cf. (9.5).

So, what kind of a function is our fully-trained infinite-width nonlinear neural network computing? To assess this, note that the ratio $\delta\delta\Theta_{[0]}/\Theta_{00}$ captures the *curvature* of the $*$-polation in the neighborhood of the training points.[41] This curvature encodes a non-universal *inductive bias* of the activation function and architecture indicating how this class of function approximators will generalize to novel data.

For a given task and dataset, some activation functions might produce a more desired type of $*$-polation. This can be measured directly via the bias term in the generalization error. Substituting in our equal norm expression for the mean (10.115),

$$m^{\infty}_{i;(2s-1)} - y_{i;(2s-1)} = \Big[y_{i;+} + (1-s)y_{i;-} - y_{i;(2s-1)}\Big] \\ - 4s(1-s)\left(\frac{\delta\delta\Theta_{[0]}}{\Theta_{00}}\right)(y_{i;+} + y_{i;-}) + O\Big(\delta^3\Big) \ , \tag{10.117}$$

we see that this generalization error bias decomposes into a comparison between the nonlinearity in the true output – given by the first square brackets – and the network curvature around the midpoint of the true output $(y_{i;+} + y_{i;-})/2$. With this framing, deep linear networks promote a very particular type of inductive bias: only linear functions are computed. More generally, we could (but won't here) compute and solve a recursion for $\delta\delta\Theta_{[0]}$ for any particular activation function in order to learn more about the kinds of functions computed by deep networks with that activation function.

Finally, note that this analysis doesn't make any particular distinction between *interpolation* and *extrapolation*, and also that as $s \to 0, 1$, the $*$-polation (10.113) reduces to $y_{i;\pm}$ with absolute certainty. In fact, in the neighborhood of $s = 0, 1$, the $*$-polation bias (10.117) has much in common with the prediction bias (10.56) and (10.62) that we saw in §10.3.1 when considering a training set consisting of only one training sample. Importantly, it is the most nearby training point that contributes the most to a test point's prediction.

Taken as a guide to thinking about larger training sets, the *local* nature of these predictions is highly suggestive of some ways to make further progress. On the one hand, we might be able to make theoretical progress on more complicated prediction formulae by weighting the predictions given by nearby training points to a given test point, perhaps using an approximation from §10.3.1 when there's only one nearby training point and using $*$-polation (10.113) when there's a nearby pair. On the other hand, we might be able to make practical progress on training-set design – given the network's inductive biases – by using this kind of analysis to inform how best to sample training inputs over the data manifold.

[41] Note that as the two training samples begin to coincide $x_{\pm} \to x_0$, the curvature vanishes quadratically $\delta\delta\Theta_{[0]}/\Theta_{00} = O\big(\delta^2\big)$, and the closer the $*$-polation will be to a linear $*$-polation. Further applying our generalized δ expansion (10.110) to the kernel, we can show that that the variance of the $*$-polation vanishes even more quickly in this coincident limit as

$$\mathbb{E}\left[z^{(L)}_{i;2s-1}(T)z^{(L)}_{i;2s-1}(T)\right] - \left(\mathbb{E}\left[z^{(L)}_{i;2s-1}(T)\right]\right)^2 = O\big(\delta^3\big) \ . \tag{10.116}$$

10.4 Linear Models and Kernel Methods

Before we back off the infinite-width limit, let's take a section to place what we've done in this chapter into the broader context of machine learning. In the next chapter, such a context will help us understand the ways in which deep learning at finite width is qualitatively quite different from its infinite-width counterpart.

In particular, in this section we'll explain a *dual* way of thinking about the class of models that can be described by a kernel prediction formula such as (10.39). On the one hand, kernel predictions can be thought of as being made by $*$-polating the training data using the kernel. On the other hand, we can think of them as the output of a trained model that's linear in its parameters. The former perspective has been more natural to us, given that we always consider an ensemble over the model parameters and then integrate them out. So let's begin by explaining the latter *linear model* perspective.[42]

10.4.1 Linear Models

The simplest linear model – and perhaps the simplest machine learning model – is just a one-layer (i.e. zero-hidden-layer) network

$$z_i(x_\delta; \theta) = b_i + \sum_{j=1}^{n_0} W_{ij} x_{j;\delta} \, . \tag{10.118}$$

While this model is linear in both the parameters $\theta = \{b_i, W_{ij}\}$ and the input $x_{j;\delta}$, the *linear* in *linear model* takes its name from the dependence on the parameters θ and not the input x. In particular, while the components of the input samples $x_{j;\delta}$ sometime can serve as a reasonable set of features for function approximation, in general they do not. Indeed, considering how much ink we've already spilled on representation group flow and representation learning in the context of deep learning, it's natural to expect that we would need to (pre-)process the input data before it's useful for any machine learning task.

One traditional way to fix this, inherited from statistics, is to engineer better features. Such an approach was necessary when computers were less powerful and models had to be much simpler to optimize. For instance, in addition to the features x_j perhaps it would also be useful for the model to take into account features $x_j x_k$ that let us consider the dependence of one component upon another. More generally, we might design a fixed set of **feature functions** $\phi_j(x)$ that's meant to work well for the dataset $\mathcal{D}$ and the underlying task at hand.[43]

In this traditional approach, the hope is that all the complicated modeling work goes into the construction of these feature functions $\phi_j(x)$ and, if we do a good enough job,

[42] The connection between infinite-width networks trained by gradient descent and kernel methods was pointed out in [55] in the context of introducing the NTK. Following that, an extended discussion of such networks as linear models was given in [60].

[43] These type of feature functions are also useful if the input x is something abstract – such as a document of text – and thus needs to be transformed into a numerical vector before it can be processed by a parameterized model.

then its associated **linear model**,

$$z_i(x_\delta;\theta) = b_i + \sum_{j=1}^{n_f} W_{ij}\phi_j(x_\delta) = \sum_{j=0}^{n_f} W_{ij}\phi_j(x_\delta)\,, \tag{10.119}$$

is simple to train, easy to interpret, and performs well on the desired task. Here, we've followed a customary notational reductionism, subsuming the bias vector into the weight matrix by setting $\phi_0(x) \equiv 1$ and $W_{i0} \equiv b_i$. Thus, the output $z_i(x;\theta)$ of a linear model depends linearly on the model parameters θ, consisting of a combined weight matrix W_{ij} of dimension $n_{\text{out}} \times (n_f + 1)$. We can still think of this model as a one-layer neural network, but in this case we pre-process each input with the function $\phi_j(x)$ before passing it through the network.

Now let's explain how to learn the optimal values for weight matrix $W^\star_{ij}$ given a training set $\mathcal{A}$. The most common approach is to minimize the MSE loss

$$\mathcal{L}_A(\theta) = \frac{1}{2}\sum_{\tilde{\alpha}\in\mathcal{A}}\sum_{i=1}^{n_{\text{out}}}\left[y_{i;\tilde{\alpha}} - z_i(x_{\tilde{\alpha}};\theta)\right]^2 = \frac{1}{2}\sum_{\tilde{\alpha}\in\mathcal{A}}\sum_{i=1}^{n_{\text{out}}}\left[y_{i;\tilde{\alpha}} - \sum_{j=0}^{n_f} W_{ij}\phi_j(x_{\tilde{\alpha}})\right]^2 . \tag{10.120}$$

Supervised learning with a linear model is known as **linear regression**, and – as the MSE loss of a linear model is necessarily quadratic in the model parameters – this is another case of an analytically-solvable learning problem (7.7). Taking the derivative of $\mathcal{L}_\mathcal{A}$ with respect to the parameters and setting it to zero, we get an implicit equation that determines the optimal weight matrix $W^\star_{ij}$:

$$\sum_{k=0}^{n_f} W^\star_{ik}\left[\sum_{\tilde{\alpha}\in\mathcal{A}}\phi_k(x_{\tilde{\alpha}})\phi_j(x_{\tilde{\alpha}})\right] = \sum_{\tilde{\alpha}\in\mathcal{A}} y_{i;\tilde{\alpha}}\phi_j(x_{\tilde{\alpha}})\,. \tag{10.121}$$

To solve this equation, let's define a symmetric (n_f+1)-by-(n_f+1) matrix of features,

$$M_{ij} \equiv \sum_{\tilde{\alpha}\in\mathcal{A}} \phi_i(x_{\tilde{\alpha}})\phi_j(x_{\tilde{\alpha}})\,, \tag{10.122}$$

with elements that give a pairwise aggregation of feature functions summed over all the training samples $\tilde{\alpha} \in \mathcal{A}$. Then, applying its inverse to both sides of the implicit expression (10.121), we find a solution:

$$W^\star_{ij} = \sum_{k=0}^{n_f}\sum_{\tilde{\alpha}\in\mathcal{A}} y_{i;\tilde{\alpha}}\phi_k(x_{\tilde{\alpha}})\left(M^{-1}\right)_{kj}\,. \tag{10.123}$$

Notice that the solution depends on the training set, linearly for the true function values $y_{i;\tilde{\alpha}}$ and in a more complicated way on the input features $\phi_k(x_{\tilde{\alpha}})$.[44] Finally, we can

[44] If the number of features (n_f+1) is larger than the size of the training set $N_\mathcal{A}$, then the model is *overparameterized*, and M_{ij} is not uniquely invertible. One scheme to specify the solution is to add a

use this fully-trained linear model with its associated optimal parameters $W^\star_{ij}$ to make predictions on novel test-set inputs $x_{\dot\beta}$ as

$$z_i(x_{\dot\beta};\theta^\star) = \sum_{j=0}^{n_f} W^\star_{ij}\phi_j(x_{\dot\beta})\,, \tag{10.125}$$

giving us a closed-form solution for our linear regression problem. Importantly, after learning is complete we can simply store the optimal parameters $W^\star_{ij}$ and forget about the training data.

10.4.2 Kernel Methods

While this is all very easy, it's less familiar in our book since we typically do not work explicitly with the parameters. To cast our linear model into a more familiar form, let's consider a *dual* expression for the solution. First, let's substitute our expression for the optimal parameters $W^\star_{ij}$, (10.123), into our linear regression solution, (10.125), giving

$$z_i(x_{\dot\beta};\theta^\star) = \sum_{\tilde\alpha\in\mathcal{A}}\left[\sum_{j,k=0}^{n_f}\phi_j(x_{\dot\beta})\left(M^{-1}\right)_{jk}\phi_k(x_{\tilde\alpha})\right]y_{i;\tilde\alpha}\,. \tag{10.126}$$

Note that the expression in the square brackets involves the inversion of an $(n_f+1)\times(n_f+1)$-dimensional matrix M_{ij}, which was required to obtain the optimal parameters $W^\star_{ij}$. This works well if the number of features is small, but if the number of feature functions we defined is very large $n_f \gg 1$, then representing and inverting such a matrix might be computationally difficult.

However, it turns out that we actually don't need to do any of that. To see why, let us introduce a new $N_\mathcal{D}\times N_\mathcal{D}$-dimensional symmetric matrix:

$$k_{\delta_1\delta_2} \equiv k(x_{\delta_1},x_{\delta_2}) \equiv \sum_{i=0}^{n_f}\phi_i(x_{\delta_1})\,\phi_i(x_{\delta_2})\ . \tag{10.127}$$

As an inner product of feature functions, $k_{\delta_1\delta_2}$ is a measure of similarity between two inputs $x_{i;\delta_1}$ and $x_{i;\delta_2}$ in feature space. Such a measure of similarity is called a **kernel**.[45]

regularization term of the form $a\sum_{ij}W^2_{ij}$ to the loss (10.120), cf. footnote 21 in §10.2.4 for a related discussion of regularization for infinite-width networks. In this modified regression problem, we can then invert the regularized matrix

$$M_{ij} = 2a\,\delta_{ij} + \sum_{\tilde\alpha\in\mathcal{A}}\phi_i(x_{\tilde\alpha})\phi_j(x_{\tilde\alpha})\,, \tag{10.124}$$

and send the regulator to zero, $a\to 0^+$, at the end of our calculations. Note that either when the regulator a is kept finite or when we're in the *underparameterized* regime with $(n_f+1)<N_\mathcal{A}$, the linear model will no longer reach zero training loss even when fully optimized.

[45]For instance, in the case of the simplest linear model (10.118), the kernel is just given by the inner product between the two inputs

$$k_{\delta_1\delta_2} \equiv \sum_{i=1}^{n_0} x_{i;\delta_1}x_{i;\delta_2}\,, \tag{10.128}$$

which is often called the *linear kernel*.

In a way that should feel very familiar, we'll also denote an $N_{\mathcal{A}}$-by-$N_{\mathcal{A}}$-dimensional submatrix of the kernel evaluated on the training set as $\widetilde{k}_{\tilde{\alpha}_1\tilde{\alpha}_2}$ with a tilde. This lets us write its inverse as $\widetilde{k}^{\tilde{\alpha}_1\tilde{\alpha}_2}$, which satisfies

$$\sum_{\tilde{\alpha}_2\in\mathcal{A}} \widetilde{k}^{\tilde{\alpha}_1\tilde{\alpha}_2}\widetilde{k}_{\tilde{\alpha}_2\tilde{\alpha}_3} = \delta^{\tilde{\alpha}_1}_{\tilde{\alpha}_3}\,. \tag{10.129}$$

Note that given the definition of the kernel (10.127), for this inverse to exist and for this equation to hold we must be in the *overparameterized* regime with $(n_f+1)\geq N_{\mathcal{A}}$.

Now with this, let's see how we might rearrange the factor in the square brackets of our solution (10.126). Multiplying it by the submatrix $\widetilde{k}_{\tilde{\alpha}\tilde{\alpha}_1}$, we can simplify this factor as

$$\begin{aligned}
&\sum_{\tilde{\alpha}\in\mathcal{A}}\left[\sum_{j,k=0}^{n_f}\phi_j(x_{\dot{\beta}})\left(M^{-1}\right)_{jk}\phi_k(x_{\tilde{\alpha}})\right]\widetilde{k}_{\tilde{\alpha}\tilde{\alpha}_1} \qquad (10.130)\\
&=\sum_{\tilde{\alpha}\in\mathcal{A}}\sum_{j,k=0}^{n_f}\phi_j(x_{\dot{\beta}})\left(M^{-1}\right)_{jk}\phi_k(x_{\tilde{\alpha}})\sum_{i=0}^{n_f}\phi_i(x_{\tilde{\alpha}})\,\phi_i(x_{\tilde{\alpha}_1})\\
&=\sum_{i,j,k=0}^{n_f}\phi_j(x_{\dot{\beta}})\left(M^{-1}\right)_{jk}M_{ki}\,\phi_i(x_{\tilde{\alpha}_1})\\
&=\sum_{i=0}^{n_f}\phi_i(x_{\dot{\beta}})\phi_i(x_{\tilde{\alpha}_1}) = k_{\dot{\beta}\tilde{\alpha}_1}\,.
\end{aligned}$$

To get this result, in the second line we plugged in the definition of the kernel (10.127), in the third line we performed the sum over $\tilde{\alpha}$ using the definition of the feature matrix M_{ij} (10.122), and in the last equality of the fourth line we again used the definition of the kernel. Finally, multiplying the first and last expressions by the inverse submatrix $\widetilde{k}^{\tilde{\alpha}_1\tilde{\alpha}_2}$, we get a new representation for the factor in the square brackets

$$\left[\sum_{j,k=0}^{n_f}\phi_j(x_{\dot{\beta}})\left(M^{-1}\right)_{jk}\phi_k(x_{\tilde{\alpha}_2})\right] = \sum_{\tilde{\alpha}_1\in\mathcal{A}} k_{\dot{\beta}\tilde{\alpha}_1}\widetilde{k}^{\tilde{\alpha}_1\tilde{\alpha}_2}\,, \tag{10.131}$$

which lets us rewrite the prediction of our linear model (10.125) as

$$z_i(x_{\dot{\beta}};\theta^\star) = \sum_{\tilde{\alpha}_1,\tilde{\alpha}_2\in\mathcal{A}} k_{\dot{\beta}\tilde{\alpha}_1}\widetilde{k}^{\tilde{\alpha}_1\tilde{\alpha}_2}y_{i;\tilde{\alpha}_2}\,. \tag{10.132}$$

When the prediction of a linear model is computed in this way, it's known as a *kernel machine* or **kernel methods**.

Note that in this dual expression of the solution, the optimal parameters $W^\star_{ij}$ and the feature functions $\phi_i(x)$ don't appear. Thus, we've successfully exchanged our feature-space quantities, an (n_f+1)-dimensional feature vector and the inverse of an $(n_f+1)\times(n_f+1)$-dimensional matrix, for sample-space quantities, an $N_{\mathcal{A}}$-dimensional vector $k_{\dot{\beta}\tilde{\alpha}_1}$

and the inverse of an $N_{\mathcal{A}} \times N_{\mathcal{A}}$-dimensional matrix $\widetilde{k}_{\tilde{\alpha}_1\tilde{\alpha}_2}$.[46] This works because in our solution (10.132), we actually only care about the inner product of the feature functions – i.e. the kernel – and not the values of the features themselves.

By writing the linear model's prediction in terms of the kernel in (10.132), we can interpret the prediction in terms of direct comparison with previously-seen examples. In particular, this solution computes the similarity of a new test input $x_{\dot{\beta}}$ with all the training examples with $k_{\dot{\beta}\tilde{\alpha}_1}$ and then uses that similarity to linearly weight the true function values from the training set $y_{i;\tilde{\alpha}_2}$ with the sample-space metric $\widetilde{k}^{\tilde{\alpha}_1\tilde{\alpha}_2}$. For this reason, kernel methods are sometimes referred to as *memory-based* methods since they involve memorizing the entire training set.[47] This should be contrasted with the parameterized linear model solution (10.125), where we forget the training set samples and instead just explicitly store the optimal parameter values $W^{\star}_{ij}$.

Finally, note that there's "no wiring" in the prediction: the z_i component of the prediction is entirely determined by the y_i component of the training examples. This is only implicit in the optimal weight matrix $W^{\star}_{ij}$ (10.123) in the linear model solution (10.125) but is explicit in the kernel method solution (10.132). This is one of many ways that such linear models and kernel methods are limited machine learning models.

[46] In some situations, specifying and evaluating the kernel is much simpler than specifying and evaluating the feature functions. For instance, the *Gaussian kernel*, given by

$$k_{\delta_1\delta_2} \equiv \exp\left[-\frac{1}{2\sigma^2}\sum_{i=1}^{n_0}\left(x_{i;\delta_1} - x_{i;\delta_2}\right)^2\right], \tag{10.133}$$

implies an infinite-dimensional feature space, but can be evaluated by simply computing the squared distance between the n_{in}-dimensional input vectors and then exponentiating the result. (To see why the Gaussian kernel implies an infinite number of feature functions, we can express the squared distance as a sum of three inner products and then Taylor expand the exponential in those inner products; the terms in the Taylor expansion give the feature functions.) In this way, we see how computing the kernel can be far easier than representing the features explicitly.

In fact, any algorithm based on a linear kernel (10.128) can be generalized by swapping the simple kernel for a more complicated kernel like the Gaussian kernel (10.133). This is known as the **kernel trick** and is a way to describe in the language of kernel methods how we generalized our simplest linear model (10.118) – that was linear in the input – to the more general linear model (10.119) – that was nonlinear in the input.

[47] Often, for a particular kernel method to be tractable, the model's predictions are made *locally*, incorporating information mostly from the training samples nearest to the test sample of interest. Given the $*$-polation results of last section, it's not hard to imagine how such methods could be made to work well.

A canonical example of such a local method is *k-nearest neighbors* [61, 62], which is a special type of kernel method. By only considering nearby training points, these kinds of local algorithms can skirt some of the impracticality that we've been pointing out for our exact Bayesian inference predictions as well as for our frozen NTK predictions. It would be interesting to extend such local algorithms to the finite-width exact Bayesian inference that we discussed in §6.4 or the finite-width gradient-based learning prediction that we'll discuss in §∞.

10.4.3 Infinite-Width Networks as Linear Models

Surely, the kernel methods' prediction formula (10.132) should seem awfully familiar to you: it is precisely the same as the exact Bayesian mean prediction (6.64) if we identify the kernel methods' kernel $k_{\delta_1\delta_2}$ with the Bayesian kernel $K^{(L)}_{\delta_1\delta_2}$, and it is exactly the same as the (neural tangent) kernel mean prediction (10.40) if we identify the kernel methods' kernel $k_{\delta_1\delta_2}$ with the frozen neural tangent kernel $\Theta^{(L)}_{\delta_1\delta_2}$.[48] This finally provides a justification for the names of these objects as well as for the name of the current chapter.

Indeed, there is a very direct connection between these traditional linear models and kernel methods on the one hand and our (neural tangent) kernel learning of infinite-width models on the other hand. First, let's discuss the simpler case of the Bayesian kernel. As pointed out in §10.2.4, this choice corresponds to treating only output-layer biases $b_i^{(L)}$ and weights $W_{ij}^{(L)}$ as trainable model parameters, and so the network output at initialization is given by

$$z_i^{(L)}\left(x_\delta;\theta\right) = b_i^{(L)} + \sum_{j=1}^{n_{L-1}} W_{ij}^{(L)} \sigma_{j;\delta}^{(L-1)} = \sum_{j=0}^{n_{L-1}} W_{ij}^{(L)} \sigma_{j;\delta}^{(L-1)} \,, \tag{10.134}$$

where on the right-hand side we defined $\sigma_{0;\delta}^{(L-1)} \equiv 1$ and $W_{i0}^{(L)} \equiv b_i^{(L)}$. This is almost the same as our linear model (10.119) except that the feature functions are *random*: $\widehat{\phi}_{j;\delta} \equiv \sigma_{j;\delta}^{(L-1)}$. In particular, here we've *hatted* these feature functions to emphasize that they depend on the parameters in the hidden layers that are sampled from the initialization distribution at the beginning and then *fixed*; this is sometimes called a **random feature model**.

In this case, the kernel methods' notion of the kernel is also stochastic

$$\begin{aligned} \widehat{k}_{\delta_1\delta_2} =& C_b^{(L)} \widehat{\phi}_{0;\delta_1} \widehat{\phi}_{0;\delta_2} + \frac{C_W^{(L)}}{n_{L-1}} \sum_{j=1}^{n_{L-1}} \widehat{\phi}_{j;\delta_1} \widehat{\phi}_{j;\delta_2} \\ =& C_b^{(L)} + C_W^{(L)} \left(\frac{1}{n_{L-1}} \sum_{j=1}^{n_{L-1}} \sigma_{j;\delta_1}^{(L-1)} \sigma_{j;\delta_2}^{(L-1)} \right) , \end{aligned} \tag{10.135}$$

where in the first line we have re-weighted the terms in the feature sum in the definition of the kernel (10.127) by $C_b^{(L)}$ and $C_W^{(L)}/n_L$.[49] Note that we called this object (10.135) the *stochastic metric* (4.70) when studying the RG flow of preactivations. Now, taking

[48] You may have noticed that our stochastic (neural tangent) kernel prediction formula (10.39) also depended on the network output at initialization and had a nonzero covariance. This is related to our earlier discussion in footnote 44 that, when we're in the *overparameterized* regime with $(n_f + 1) > N_{\mathcal{A}}$, as is especially the case when we have an infinite number of features, the $(n_f + 1)$-by-$(n_f + 1)$ matrix matrix $M_{ij} \equiv \sum_{\tilde{\alpha}\in\mathcal{A}} \phi_i(x_{\tilde{\alpha}})\phi_j(x_{\tilde{\alpha}})$ (10.122) does not have a unique inverse. Thus, in this regime, the optimal weight matrix $W^\star$ is not unique: if we don't use the regulation trick (10.124) to uniquely pick out one of the solutions, the prediction in the dual kernel description will have a dependence on the model's initialization.

[49] A more general definition of the kernel methods' kernel (10.127) allows us to weight the contribution

an expectation over the initialization ensemble, in the infinite-width limit we have

$$\mathbb{E}\left[\widehat{k}_{\delta_1\delta_2}\right] = C_b^{(L)} + C_W^{(L)} \left\langle \sigma_{\delta_1}\sigma_{\delta_2} \right\rangle_{K^{(L-1)}} = K^{(L)}_{\delta_1\delta_2} \,, \tag{10.137}$$

where in the last equality we used the recursion for the kernel (10.44).[50] In this way, we see how we can interpret exact Bayesian inference at infinite width as a simple linear model (10.134) of fixed random features.

Now, let's give a linear model interpretation to gradient-based learning at infinite width. Since a linear model is linear in its parameters, we can more generally define the **random features** by

$$\widehat{\phi}_{i,\mu}(x_\delta) \equiv \frac{dz^{(L)}_{i;\delta}}{d\theta_\mu} \,. \tag{10.138}$$

To be clear, this derivative is evaluated at initialization and these features are thus fixed in the infinite-width limit. Explicitly, for an MLP the random features are given by

$$\widehat{\phi}_{i,W^{(\ell)}_{k_1k_2}}(x_\delta) = \left(\sum_{j_{L-1},\ldots,j_{\ell+1}} W^{(L)}_{ij_{L-1}} \sigma'^{(L-1)}_{j_{L-1};\delta} \cdots W^{(\ell+1)}_{j_{\ell+1}k_1} \sigma'^{(\ell)}_{k_1;\delta} \right) \sigma^{(\ell-1)}_{k_2;\delta} \,, \tag{10.139}$$

with the bias component given by setting $\sigma^{(\ell)}_{0;\delta} \equiv 1$ and $W^{(\ell)}_{k0} \equiv b^{(\ell)}_k$. As is apparent from this expression, these features are stochastic, depending on the specific values of the biases and weights at initialization. Note also that the Bayesian linear model (10.134) only uses a subset of these features, $\widehat{\phi}_{i,W^{(L)}_{ij}} = \sigma^{(L-1)}_{j;\delta}$, and thus is a much more limited and less expressive model.

Note further that these feature functions $\widehat{\phi}_{i,\mu}(x_\delta)$ are related to, but not exactly equivalent to, the previous notion of feature we gave when we discussed representation group flow in §4.6. In that case, our ℓ-th-layer features corresponds to ℓ-th-layer preactivations $z^{(\ell)}_{i;\delta}$ or activations $\sigma^{(\ell)}_{i;\delta}$. However, here we see that the random feature functions (10.139) are proportional to $(\ell-1)$-th-layer activations $\sigma^{(\ell-1)}_{i;\delta}$ but are also multiplied by objects from deeper layers.

As should be clear, the stochastic kernel associated with these features,

$$\widehat{k}_{ij;\delta_1\delta_2} \equiv \sum_{\mu,\nu} \lambda_{\mu\nu}\, \widehat{\phi}_{i,\mu}(x_{\delta_1}) \widehat{\phi}_{j,\nu}(x_{\delta_2}) = \sum_{\mu,\nu} \lambda_{\mu\nu} \frac{dz^{(L)}_{i;\delta_1}}{d\theta_\mu} \frac{dz^{(L)}_{j;\delta_2}}{d\theta_\nu} \equiv \widehat{H}^{(L)}_{ij;\delta_1\delta_2} \,, \tag{10.140}$$

---

of each pair of feature functions as

$$k_{\delta_1\delta_2} \equiv \sum_{i,j=0}^{n_f} c_{ij}\, \phi_i(x_{\delta_1})\, \phi_j(x_{\delta_2}) \,. \tag{10.136}$$

[50] Note alternatively, by the central limit theorem, that the stochastic kernel $\widehat{k}_{\delta_1\delta_2}$ will be equal to the kernel $K^{(L)}_{\delta_1\delta_2}$ in the infinite-width limit without explicitly averaging over initializations. This *self-averaging* of the kernel is equivalent to the fact that the connected four-point correlator vanishes at infinite width. Here we see that such self-averaging of the kernel can also be thought of as arising from a sum over an infinite number of random features.

is just the L-th-layer stochastic NTK (8.5).[51] Here, we have taken advantage of our more general definition of the kernel methods' kernel (10.136) to incorporate the learning-rate tensor $\lambda_{\mu\nu}$ into the expression. Accordingly, at infinite width the NTK is frozen and diagonal in final layer neural indices, giving

$$k_{\delta_1\delta_2} \equiv \sum_{\mu,\nu} \lambda_{\mu\nu}\, \widehat{\phi}_{i,\mu}(x_{\delta_1})\, \widehat{\phi}_{i,\nu}(x_{\delta_2}) = \sum_{\mu,\nu} \lambda_{\mu\nu} \frac{dz^{(L)}_{i;\delta_1}}{d\theta_\mu} \frac{dz^{(L)}_{i;\delta_2}}{d\theta_\nu} \equiv \Theta^{(L)}_{\delta_1\delta_2} \,. \tag{10.141}$$

In this way, we see that at infinite width the fully-trained mean network output is just a linear model based on random features (10.138). In this sense, infinite-width neural networks are rather shallow in terms of model complexity, however deep they may appear.

Looking back, when we discussed the linear model at the beginning of this section, we had to introduce feature functions $\phi_i(x)$, designed using our knowledge of the task and data at hand, as a way to (pre-)process the input. This way, the parametric model that we learn is really simple, i.e. linear.[52] We then reinterpreted this fit linear model in terms of its associated kernel, which itself has a natural interpretation as measuring similarity between our designed features.

However, for infinite-width networks we didn't *design* the frozen NTK, and its associated features are *random*. Instead, the network is defined by the architecture, hyperparameters, and biases and weights, and the path from those variables to the NTK and kernel prediction is filled with calculations. So the abstraction of the actual neural network seems like a very odd way to design a kernel.

Indeed, we've just learned that infinite-width neural networks can only make predictions that are linear in the true outputs from the training set; they are linear models that can only compute linear combinations of random features. Of course, deep learning is exciting because it works on problems where classic machine learning methods have failed; it works in cases where we don't know how to design feature functions or kernels, or doing so would be too complicated. For neural networks to go beyond the kernel methods, they need to be able to *learn* useful features *from the data*, not just make use of a complicated linear combination of random features.

---

[51] However please note importantly that at finite width the stochastic neural tangent *kernel* $\widehat{k}_{ij;\delta_1\delta_2} = \widehat{H}^{(L)}_{i_1 i_2;\delta_1\delta_2}$ is not fixed during training – learning useful features from the data – and randomly varies across initializations; hence – despite its name – it is not actually a kernel.

[52] Please don't confuse our discussion of *linear models* here and our discussion of linear vs. nonlinear functions as in §10.3.2.

A linear model is a model that's linear in the model parameters (10.119) and has a dual kernel description that's linear in the true outputs $y_{i;\tilde{\alpha}}$ in the training set $\mathcal{A}$ (10.132); as we have seen, linear models are very simple and easy to solve analytically. As is clear from the definition (10.119), a linear model in general will be *nonlinear* in the inputs x for general nonlinear feature functions $\phi_i(x)$. Accordingly, linear models can compute nonlinear functions of their input. We saw this explicitly when we worked out how nonlinear $*$-polation works for smooth nonlinear networks in (10.113).

In contrast, a deep linear network is a neural network that uses a `linear` activation function. Such networks compute linear functions of their input and thus may only linearly $*$-polate between training points, cf. (10.104). However, since they are *not* linear models (for $L > 1$), the function they compute depends nonlinearly on the model parameters. Accordingly, their training dynamics can be somewhat complicated, and at finite width they even exhibit representation learning.

Luckily, finite-width networks are not kernel methods. Please now turn to the next chapter to find out exactly what they are instead.

Chapter 11

Representation Learning

> *It can scarcely be denied that the supreme goal of all theory is to make the irreducible basic elements as simple and as few as possible without having to surrender the adequate representation of a single datum of experience.*
>
> Albert Einstein in a 1933 lecture, "On the Method of Theoretical Physics," [63].

Last chapter, we understood that linear models cannot learn features from data. Thus, the infinite-width limit is too simple to provide an adequate representation of deep learning; in order to include its irreducible basic element – representation learning – it is *qualitatively* important to study finite-width networks.

In the first half of this chapter, we'll analyze the leading correction to the gradient-descent update to the network output by extending its Taylor expansion to second order in the global learning rate η. After further seeing that a similar contribution arises in the first-order Taylor expansion of the update to the NTK, we'll then show that this correction is a finite-width effect. This upgrade of the NTK from fixed to dynamical indicates that for finite-width networks, the feature functions that comprise the NTK are themselves learning from the data over the course of training.

Unfortunately, the complete $O(1/n)$ contribution to the dynamics further includes terms that arise from Taylor expanding the update to the network output to third order in the global learning rate η, and similarly Taylor expanding the update to the NTK to second order in η. While it's *necessary* to include these contributions in order to actually compute the distribution of fully-trained finite-width networks, the $O(\eta^2)$ expansion of the network output and the $O(\eta)$ expansion of the NTK is *sufficient* to qualitatively investigate the mechanism for representation learning in these models.

With that in mind, in order to separate the pedagogy of representation learning from the messy phenomenological details of the real MLP, we'll spend the second half of this chapter focusing on a simplified model that's equivalent to this $O(\eta^2)$ truncation and gives a minimal qualitative picture of representation learning. These minimal models that we discuss form a valid and potentially useful class of machine learning models that perform representation learning, though annoyingly finite-width MLPs are not in

this class. Listening carefully to these real MLPs, we'll spend all of next chapter (§∞) working out their $O(1/n)$ training dynamics in full intricate detail.

To begin, in §11.1 we'll work out this second-order-in-η contribution to the update to preactivations and the first-order-in-η contribution to the update to the NTK. This lets us source all representation learning from a single irreducible basic element, the *differential of the neural tangent kernel* (dNTK): just as the NTK governs the leading-order-in-η dynamics of the preactivations, the dNTK governs the leading-order-in-η dynamics of the NTK.

After thusly identifying the dNTK as a driver of representation learning, in §11.2 we'll recursively determine its correlation with the preactivations as a function of network layer ℓ. In detail, we'll first derive a stochastic forward equation for the dNTK and then evaluate the remaining recursions needed to determine the statistics of the joint preactivation-NTK-dNTK distribution at initialization. As such, this section mirrors the structure of our *RG-flow* analysis in §4 and §8. Importantly, we'll see that all the statistics involving the dNTK are $O(1/n)$ and thus only contribute at finite width.

In §11.3, we'll apply the principles of criticality and universality to analyze the new dNTK recursions. Since all of our hyperparameters have already been fixed by the parallel analysis of the preactivations in §5 – fixing the initialization hyperparameters – and the NTK in §9 – fixing the training hyperparameters – our focus here will be on evaluating the depth and width scaling of the dNTK statistics with these fixed hyperparameters. As you might guess, we'll find across our two universality classes (§11.3.1 and §11.3.2) that the effect of the dNTK – and therefore one source of representation learning – is proportional to our effective theory cutoff, the depth-to-width ratio L/n.

Having now firmly established that the NTK evolves at finite width – and having worked out an important contribution to its dynamics – in §11.4, we'll take a step back and look for a broader context, mirroring our discussion in §10.4 for infinite-width networks. To that end, in §11.4.1 we'll introduce a class of *nonlinear models* – with a particular focus on the *quadratic model* – and thus minimally extend the traditional workhorse of machine learning, the linear model. This quadratic model provides a *minimal model* of representation learning, independent of any neural-network abstraction. Moreover, these models are simple and completely analyzable, and yet are able to capture the essence of representation learning.

After solving the implied *nearly-linear quadratic regression* problem, in §11.4.2 we'll further provide a dual description of the quadratic model solution, which we'll call *nearly-kernel methods*. This will let us identify an object that corresponds to the dNTK in this minimal setting, and show us how to make test-set predictions with a data-dependent *trained* kernel. Overall, we hope this framework will be of further theoretical and practical interest as a new class of nearly-simple machine learning models that learn representations.

At this point, the connection between these nearly-kernel methods and finite-width networks – at least at order η^2 – will be nearly manifest, and in §11.4.3 we'll make it explicit. By doing so, we'll understand precisely how deep learning is a *non-minimal* model of representation learning. Ultimately, we'll conclude that the power of deep

learning is the *deep* – the inductive bias of the network architecture induced by the layer-to-layer RG flow – providing a particularly good choice of initial features as a starting point for *learning*. These observations will be quite helpful for us in interpreting our somewhat messy finite-width solution in the following chapter.

11.1 Differential of the Neural Tangent Kernel

Recall that in the first step of gradient descent, the change in the ℓ-th-layer parameters of any particular network is given by (7.11)

$$\text{đ}\theta_\mu^{(\ell)} \equiv \theta_\mu^{(\ell)}(t=1) - \theta_\mu^{(\ell)}(t=0) = -\eta \sum_\nu \lambda_{\mu\nu}^{(\ell)} \left(\sum_{k=1}^{n_L} \sum_{\tilde{\alpha}\in\mathcal{A}} \frac{\partial \mathcal{L}_\mathcal{A}}{\partial z_{k;\tilde{\alpha}}^{(L)}} \frac{dz_{k;\tilde{\alpha}}^{(L)}}{d\theta_\nu^{(\ell)}} \right) . \tag{11.1}$$

In this section, it will be helpful to specify explicitly which layer each parameter comes from. In particular, here $\theta_\mu^{(\ell)}$ denotes either an ℓ-th-layer bias $\theta_\mu^{(\ell)} \equiv b_i^{(\ell)}$ or an ℓ-th-layer weight $\theta_\mu^{(\ell)} \equiv W_{ij}^{(\ell)}$, and the ℓ-th-layer model-parameter indices μ, ν run over all the components of the bias vector $b_i^{(\ell)}$ *and* the weight matrix $W_{ij}^{(\ell)}$ in the ℓ-th layer *only*. Additionally, to emphasize that the learning-rate tensor $\lambda_{\mu\nu}^{(\ell)}$ only connects the parameters within a given layer ℓ, we've decorated it with a layer index for clarity. For now we'll let $\lambda_{\mu\nu}^{(\ell)}$ act arbitrarily within a layer, though ultimately we'll be interested in the case where it's diagonal, with two training hyperparameters per layer, $\lambda_b^{(\ell)}$ and $\lambda_W^{(\ell)}$, as usual.

As a further reminder, quantities without any explicit step argument are taken to be evaluated at initialization – though sometimes we may also explicitly denote $t = 0$ for extra emphasis – and our sample-index notation is *alpha-with-tilde* for the inputs in the training set, $\tilde{\alpha} \in \mathcal{A}$, *beta-with-dot* for inputs in the test set, $\dot{\beta} \in \mathcal{B}$, and *delta-with-no-decoration* for inputs that could be in either set, $\delta \in \mathcal{D} = \mathcal{A} \cup \mathcal{B}$.

Now, to go beyond the infinite-width limit, we'll need to expand the change in ℓ-th-layer preactivations to *second order* in the parameter update:

$$\begin{aligned} \text{đ}z_{i;\delta}^{(\ell)} \equiv & z_{i;\delta}^{(\ell)}(t=1) - z_{i;\delta}^{(\ell)}(t=0) \\ = & \sum_{\ell_1=1}^{\ell} \sum_\mu \frac{dz_{i;\delta}^{(\ell)}}{d\theta_\mu^{(\ell_1)}} \text{đ}\theta_\mu^{(\ell_1)} + \frac{1}{2} \sum_{\ell_1,\ell_2=1}^{\ell} \sum_{\mu_1,\mu_2} \frac{d^2 z_{i;\delta}^{(\ell)}}{d\theta_{\mu_1}^{(\ell_1)} d\theta_{\mu_2}^{(\ell_2)}} \text{đ}\theta_{\mu_1}^{(\ell_1)} \text{đ}\theta_{\mu_2}^{(\ell_2)} + \ldots . \end{aligned} \tag{11.2}$$

Note that the ℓ-th-layer preactivations $z_{i;\delta}^{(\ell)}$ cannot depend on model parameters $\theta_\mu^{(\ell')}$ from layers ℓ' that are deeper than the ℓ-th layer. Thus, when $\ell' > \ell$, we have $dz_{i;\delta}^{(\ell)}/d\theta_\mu^{(\ell')} = 0$, and so we truncated our layer sums in the above expression at ℓ.

Next, we are going to slightly rewrite the parameter update equation (11.1) for the parameters $\theta_\mu^{(\ell_a)}$ appearing in our preactivation expansion (11.2), i.e. for those parameters in layers $\ell_a \leq \ell$ that contribute. To do so, we'll make use of the chain rule to

decompose the derivative of the output-layer preactivations $z^{(L)}_{k;\tilde{\alpha}}$ with respect to the ℓ_a-th-layer model parameters as

$$\frac{dz^{(L)}_{k;\tilde{\alpha}}}{d\theta^{(\ell_a)}_{\nu}} = \sum_{j} \frac{dz^{(L)}_{k;\tilde{\alpha}}}{dz^{(\ell)}_{j;\tilde{\alpha}}} \frac{dz^{(\ell)}_{j;\tilde{\alpha}}}{d\theta^{(\ell_a)}_{\nu}} \,, \tag{11.3}$$

for an intermediate layer ℓ such that $\ell_a \leq \ell$. Using this decomposition, we can rewrite our parameter update (11.1) as

$$đ\theta^{(\ell_a)}_{\mu} = -\eta \sum_{\nu} \lambda^{(\ell_a)}_{\mu\nu} \left(\sum_{j,k,\tilde{\alpha}} \frac{\partial \mathcal{L}_{\mathcal{A}}}{\partial z^{(L)}_{k;\tilde{\alpha}}} \frac{dz^{(L)}_{k;\tilde{\alpha}}}{dz^{(\ell)}_{j;\tilde{\alpha}}} \frac{dz^{(\ell)}_{j;\tilde{\alpha}}}{d\theta^{(\ell_a)}_{\nu}} \right) = -\eta \sum_{\nu,j,\tilde{\alpha}} \lambda^{(\ell_a)}_{\mu\nu} \, \epsilon^{(\ell)}_{j;\tilde{\alpha}} \frac{dz^{(\ell)}_{j;\tilde{\alpha}}}{d\theta^{(\ell_a)}_{\nu}} \,, \tag{11.4}$$

where in the last equality we introduced an *ℓ-th-layer error factor*:

$$\epsilon^{(\ell)}_{j;\tilde{\alpha}} \equiv \sum_{k=1}^{n_L} \frac{\partial \mathcal{L}_{\mathcal{A}}}{\partial z^{(L)}_{k;\tilde{\alpha}}} \frac{dz^{(L)}_{k;\tilde{\alpha}}}{dz^{(\ell)}_{j;\tilde{\alpha}}} = \frac{d\mathcal{L}_{\mathcal{A}}}{dz^{(\ell)}_{j;\tilde{\alpha}}} \,. \tag{11.5}$$

Substituting this form of the parameter update (11.4) into the ℓ-th-layer preactivation update (11.2), our second-order expansion becomes

$$\begin{aligned} đz^{(\ell)}_{i;\delta} = &- \eta \sum_{j,\tilde{\alpha}} \left(\sum_{\ell_1=1}^{\ell} \sum_{\mu,\nu} \lambda^{(\ell_1)}_{\mu\nu} \frac{dz^{(\ell)}_{i;\delta}}{d\theta^{(\ell_1)}_{\mu}} \frac{dz^{(\ell)}_{j;\tilde{\alpha}}}{d\theta^{(\ell_1)}_{\nu}} \right) \epsilon^{(\ell)}_{j;\tilde{\alpha}} \\ &+ \frac{\eta^2}{2} \sum_{j_1,j_2,\tilde{\alpha}_1,\tilde{\alpha}_2} \left(\sum_{\ell_1,\ell_2=1}^{\ell} \sum_{\substack{\mu_1,\nu_1,\\ \mu_2,\nu_2}} \lambda^{(\ell_1)}_{\mu_1\nu_1} \lambda^{(\ell_2)}_{\mu_2\nu_2} \frac{d^2 z^{(\ell)}_{i;\delta}}{d\theta^{(\ell_1)}_{\mu_1} d\theta^{(\ell_2)}_{\mu_2}} \frac{dz^{(\ell)}_{j_1;\tilde{\alpha}_1}}{d\theta^{(\ell_1)}_{\nu_1}} \frac{dz^{(\ell)}_{j_2;\tilde{\alpha}_2}}{d\theta^{(\ell_2)}_{\nu_2}} \right) \epsilon^{(\ell)}_{j_1;\tilde{\alpha}_1} \epsilon^{(\ell)}_{j_2;\tilde{\alpha}_2} \\ &+ \ldots \,, \end{aligned} \tag{11.6}$$

which is *quadratic* in such error factors. Here, it was essential that we treated the parameters in a per-layer manner and that each learning-rate tensor $\lambda^{(\ell_a)}_{\mu\nu}$ was restricted to a single layer ℓ_a; had we not done that, our decomposition (11.4) and update equation (11.6) would have been far more complicated.

Naturally, the object in the first parenthesis of the update equation (11.6) is the stochastic *ℓ-th-layer NTK*(8.5)

$$\widehat{H}^{(\ell)}_{i_1 i_2;\delta_1\delta_2} \equiv \sum_{\ell_1=1}^{\ell} \sum_{\mu,\nu} \lambda^{(\ell_1)}_{\mu\nu} \frac{dz^{(\ell)}_{i_1;\delta_1}}{d\theta^{(\ell_1)}_{\mu}} \frac{dz^{(\ell)}_{i_2;\delta_2}}{d\theta^{(\ell_1)}_{\nu}} \,, \tag{11.7}$$

as you know quite well by now, though in this version of the definition we represent the sum over layers explicitly and the sum over parameter indices μ, ν runs per layer.

In contrast, the object in the second parenthesis is new.[1] Let's call this object the stochastic **ℓ-th-layer differential of the neural tangent kernel** (dNTK) and

[1] This object first appeared, unnamed, in both [64] and [65] around the same time. Here, we'll compute its recursion, determine its scaling with depth, and emphasize its physical importance by highlighting its connection to representation learning.

symbolize it as

$$\widehat{\mathrm{d}H}^{(\ell)}_{i_0 i_1 i_2;\delta_0\delta_1\delta_2} \equiv \sum_{\ell_1,\ell_2=1}^{\ell} \sum_{\substack{\mu_1,\nu_1,\\ \mu_2,\nu_2}} \lambda^{(\ell_1)}_{\mu_1\nu_1}\lambda^{(\ell_2)}_{\mu_2\nu_2} \frac{d^2 z^{(\ell)}_{i_0;\delta_0}}{d\theta^{(\ell_1)}_{\mu_1} d\theta^{(\ell_2)}_{\mu_2}} \frac{dz^{(\ell)}_{i_1;\delta_1}}{d\theta^{(\ell_1)}_{\nu_1}} \frac{dz^{(\ell)}_{i_2;\delta_2}}{d\theta^{(\ell_2)}_{\nu_2}} \,. \tag{11.8}$$

Here, the hats on both the NTK and the dNTK remind us that these objects are stochastic, depending on the particular realization of the model parameters at initialization. Also, from its definition note that the dNTK is symmetric in its second and third paired set of indices $(i_1, \delta_1) \leftrightarrow (i_2, \delta_2)$, while the first neural-sample index (i_0, δ_0) is distinguished from the other two.

Using both definitions (11.7) and (11.8), our second-order expansion (11.6) can be more compactly written as

$$đz^{(\ell)}_{i;\delta} = -\eta \sum_{j,\tilde{\alpha}} \widehat{H}^{(\ell)}_{ij;\delta\tilde{\alpha}} \epsilon^{(\ell)}_{j;\tilde{\alpha}} + \frac{\eta^2}{2} \sum_{j_1,j_2,\tilde{\alpha}_1,\tilde{\alpha}_2} \widehat{\mathrm{d}H}^{(\ell)}_{ij_1j_2;\delta\tilde{\alpha}_1\tilde{\alpha}_2} \epsilon^{(\ell)}_{j_1;\tilde{\alpha}_1} \epsilon^{(\ell)}_{j_2;\tilde{\alpha}_2} + \ldots \,. \tag{11.9}$$

In other words, we have a power series in error factors. To ultimately understand how the preactivations evolve under gradient descent at leading order in $1/n$, we'll actually need to extend this expansion to order η^3, which in turn will require that we introduce a few additional tensors. Rather than worry about that now, we'll put it off to §∞. Regardless of those additional higher-order terms, from (11.9) we already see that we'll need to know the joint statistics of the preactivations – encoding the error factors $\epsilon^{(\ell)}_{j;\tilde{\alpha}}$ – the NTK $\widehat{H}^{(\ell)}_{ij;\delta\tilde{\alpha}}$, and the dNTK $\widehat{\mathrm{d}H}^{(\ell)}_{ij_1j_2;\delta\tilde{\alpha}_1\tilde{\alpha}_2}$.

Finally, as an explanation for our choice of name and symbol for the dNTK, consider the leading-order update to the ℓ-th-layer NTK after a step of gradient descent:

$$\begin{aligned}
đH^{(\ell)}_{i_1i_2;\delta_1\delta_2} \equiv& H^{(\ell)}_{i_1i_2;\delta_1\delta_2}(t=1) - H^{(\ell)}_{i_1i_2;\delta_1\delta_2}(t=0) \\
=& \sum_{\ell_1=1}^{\ell} \sum_{\mu_1} \frac{dH^{(\ell)}_{i_1i_2;\delta_1\delta_2}}{d\theta^{(\ell_1)}_{\mu_1}} đ\theta^{(\ell_1)}_{\mu_1} + \ldots \\
=& -\eta \sum_{\ell_1=1}^{\ell} \sum_{\mu_1} \left[\frac{d}{d\theta^{(\ell_1)}_{\mu_1}} \left(\sum_{\ell_2=1}^{\ell} \sum_{\mu_2,\nu_2} \lambda^{(\ell_2)}_{\mu_2\nu_2} \frac{dz^{(\ell)}_{i_1;\delta_1}}{d\theta^{(\ell_2)}_{\mu_2}} \frac{dz^{(\ell)}_{i_2;\delta_2}}{d\theta^{(\ell_2)}_{\nu_2}} \right) \right] \left[\sum_{\nu_1} \lambda^{(\ell_1)}_{\mu_1\nu_1} \sum_{j,\tilde{\alpha}} \frac{dz^{(\ell)}_{j;\tilde{\alpha}}}{d\theta^{(\ell_1)}_{\nu_1}} \epsilon^{(\ell)}_{j;\tilde{\alpha}} \right] + \ldots \\
=& -\eta \sum_{j,\tilde{\alpha}} \left[\sum_{\ell_1,\ell_2=1}^{\ell} \sum_{\substack{\mu_1,\nu_1,\\ \mu_2,\nu_2}} \lambda^{(\ell_1)}_{\mu_1\nu_1}\lambda^{(\ell_2)}_{\mu_2\nu_2} \frac{d^2 z^{(\ell)}_{i_1;\delta_1}}{d\theta^{(\ell_1)}_{\mu_1} d\theta^{(\ell_2)}_{\mu_2}} \frac{dz^{(\ell)}_{i_2;\delta_2}}{d\theta^{(\ell_2)}_{\nu_2}} \frac{dz^{(\ell)}_{j;\tilde{\alpha}}}{d\theta^{(\ell_1)}_{\nu_1}} \right] \epsilon^{(\ell)}_{j;\tilde{\alpha}} \\
& -\eta \sum_{j,\tilde{\alpha}} \left[\sum_{\ell_1,\ell_2=1}^{\ell} \sum_{\substack{\mu_1,\nu_1,\\ \mu_2,\nu_2}} \lambda^{(\ell_1)}_{\mu_1\nu_1}\lambda^{(\ell_2)}_{\mu_2\nu_2} \frac{dz^{(\ell)}_{i_1;\delta_1}}{d\theta^{(\ell_2)}_{\mu_2}} \frac{d^2 z^{(\ell)}_{i_2;\delta_2}}{d\theta^{(\ell_1)}_{\mu_1} d\theta^{(\ell_2)}_{\nu_2}} \frac{dz^{(\ell)}_{j;\tilde{\alpha}}}{d\theta^{(\ell_1)}_{\nu_1}} \right] \epsilon^{(\ell)}_{j;\tilde{\alpha}} + \ldots \\
=& -\eta \sum_{j,\tilde{\alpha}} \left(\widehat{\mathrm{d}H}^{(\ell)}_{i_1i_2j;\delta_1\delta_2\tilde{\alpha}} + \widehat{\mathrm{d}H}^{(\ell)}_{i_2i_1j;\delta_2\delta_1\tilde{\alpha}} \right) \epsilon^{(\ell)}_{j;\tilde{\alpha}} + \ldots \,.
\end{aligned} \tag{11.10}$$

Here, in the third line we inserted the definition of NTK (11.7) and the parameter update (11.4) for $\ell_1 \leq \ell$, and on the final line we used the definition of the dNTK (11.8). Thus we see that the dNTK – when multiplied by the global learning rate and contracted with an ℓ-th-layer error factor – gives the update to the ℓ-th-layer NTK after a step of gradient descent.[2]

Since we know that the infinite-width NTK is *frozen* $\widehat{H}^{(\ell)} \to \Theta^{(\ell)}$, the relation between the NTK update and the dNTK implies that the dNTK must be a finite-width effect, vanishing in the strict infinite-width limit $\widehat{\mathrm{d}H}^{(\ell)} \to 0$. Similarly, at infinite width we truncated the preactivation updates (11.9) to be linear in the global learning rate η, cf. (10.2). In the next section, we will verify all of this by computing the dNTK recursively and showing explicitly that $\widehat{\mathrm{d}H}^{(\ell)} = O(1/n)$.

11.2 RG Flow of the dNTK

As its title suggests, the structure of this section parallels §4 – where we worked out the layer-to-layer representation group (RG) flow of the preactivation distribution $p\Big(z^{(\ell)}\Big|\mathcal{D}\Big)$ – and §8 – where we worked out the layer-to-layer RG flow of the NTK-preactivation joint distribution $p\Big(z^{(\ell)}, \widehat{H}^{(\ell)}\Big|\mathcal{D}\Big)$. Specifically, we will now work out the effective ℓ-th-layer joint distribution of the preactivations, the NTK, and the dNTK:

$$p\Big(z^{(\ell)}, \widehat{H}^{(\ell)}, \widehat{\mathrm{d}H}^{(\ell)}\Big|\mathcal{D}\Big)\,. \tag{11.11}$$

This analysis is important for two reasons: *(i)* firstly, understanding the statistics of this ℓ-th-layer joint distribution at order $1/n$ is a necessary prerequisite for understanding the leading nontrivial finite-width corrections for deep neural networks trained with gradient-based learning; *(ii)* secondly, in §11.4 we will see that a nonvanishing dNTK is sufficient for a network to exhibit representation learning, and thus by showing that the dNTK is of order $1/n$, we will firmly establish that the leading-order finite-width effective theory is able to describe this essential property of deep learning.

Zeroth, we'll establish the stochastic iteration equation for the dNTK (§11.2.0). Then, beginning our statistical analysis, first we'll see that the dNTK vanishes identically in the first layer (§11.2.1). Second, we'll see that there's a nontrivial cross correlation between the dNTK and the preactivations in the second layer (§11.2.2). Third and finally, we'll work out a general recursion that controls the accumulation of such dNTK-preactivation cross correlations in deeper layers (§11.2.3).

[2] Please don't confuse our italicized, crossed, and unhatted notation, $\not{d}H^{(\ell)}_{i_1 i_2;\delta_1\delta_2}$, representing the first update to the NTK, with our unitalicized, uncrossed, and hatted notation, $\widehat{\mathrm{d}H}^{(\ell)}_{i_0 i_1 i_2;\delta_0\delta_1\delta_2}$, representing the dNTK. In this chapter we will focus on the statistics of the dNTK, and we will not use this notation when evaluating the NTK dynamics in the following chapter.

11.2.0 Forward Equation for the dNTK

Just as we needed to derive a stochastic forward iteration equation for the NTK (8.12) in §8.0 before working out recursions for its statistics, here we'll derive such an equation for the dNTK.

Let's start by writing out the definition of the dNTK (11.8) at layer $(\ell+1)$:

$$\widehat{\mathrm{d}H}^{(\ell+1)}_{i_0 i_1 i_2;\delta_0\delta_1\delta_2} \equiv \sum_{\ell_1,\ell_2=1}^{\ell+1} \left[\sum_{\substack{\mu_1,\nu_1,\\ \mu_2,\nu_2}} \lambda^{(\ell_1)}_{\mu_1\nu_1} \lambda^{(\ell_2)}_{\mu_2\nu_2} \frac{d^2 z^{(\ell+1)}_{i_0;\delta_0}}{d\theta^{(\ell_1)}_{\mu_1} d\theta^{(\ell_2)}_{\mu_2}} \frac{dz^{(\ell+1)}_{i_1;\delta_1}}{d\theta^{(\ell_1)}_{\nu_1}} \frac{dz^{(\ell+1)}_{i_2;\delta_2}}{d\theta^{(\ell_2)}_{\nu_2}} \right] . \tag{11.12}$$

To determine its forward equation, we need to explicitly evaluate the derivatives with respect to the $(\ell+1)$-th-layer parameters and also rewrite all the $(\ell+1)$-th-layer quantities in terms of the ℓ-th-layer quantities using the chain rule. Depending on the values of ℓ_1 and ℓ_2, there are thus three cases to consider for the double summation over layers.

First, when both layers are maximal $\ell_1 = \ell_2 = \ell+1$ there is no contribution. Recalling for one final time the preactivation forward equation,

$$z^{(\ell+1)}_{i;\delta} = b^{(\ell+1)}_i + \sum_{j=1}^{n_\ell} W^{(\ell+1)}_{ij} \sigma^{(\ell)}_{j;\delta} , \tag{11.13}$$

we see that the $(\ell+1)$-th-layer preactivations are always linear in the $(\ell+1)$-th-layer model parameters $\theta^{(\ell+1)}_\mu$. Thus, in this case the second derivative in the dNTK definition (11.12) will vanish.

Second, when $\ell_1 = \ell+1$ and $\ell_2 < \ell+1$, there is a contribution from the $(\ell+1)$-th-layer weights but not from the $(\ell+1)$-th-layer biases. Considering the bias $\theta^{(\ell_1)}_{\mu_1} = b^{(\ell+1)}_j$, the $(\ell+1)$-th-layer derivative gives a Kronecker delta

$$\frac{dz^{(\ell+1)}_{i;\delta}}{db^{(\ell+1)}_j} = \delta_{ij} , \tag{11.14}$$

and so the second derivative again vanishes

$$\frac{d^2 z^{(\ell+1)}_{i;\delta}}{db^{(\ell+1)}_j d\theta^{(\ell_2)}_{\mu_2}} = 0 . \tag{11.15}$$

Instead considering the weight matrix $\theta^{(\ell_1)}_{\mu_1} = W^{(\ell+1)}_{jk}$, the $(\ell+1)$-th-layer derivative is not a constant

$$\frac{dz^{(\ell+1)}_{i;\delta}}{dW^{(\ell+1)}_{jk}} = \delta_{ij} \sigma^{(\ell)}_{k;\delta} . \tag{11.16}$$

Thus, the second derivative evaluates to something nontrivial

$$\frac{d^2 z^{(\ell+1)}_{i;\delta}}{dW^{(\ell+1)}_{jk} d\theta^{(\ell_2)}_{\mu_2}} = \delta_{ij} \sigma'^{(\ell)}_{k;\delta} \frac{dz^{(\ell)}_{k;\delta}}{d\theta^{(\ell_2)}_{\mu_2}} , \tag{11.17}$$

while the remaining first derivative gives

$$\frac{dz^{(\ell+1)}_{i;\delta}}{d\theta^{(\ell_2)}_{\nu_2}} = \sum_k W^{(\ell+1)}_{ik} \sigma'^{(\ell)}_{k;\delta} \frac{dz^{(\ell)}_{k;\delta}}{d\theta^{(\ell_2)}_{\nu_2}} \,, \tag{11.18}$$

with the use of the chain rule. Plugging in these three derivative evaluations (11.16), (11.17), and (11.18) to evaluate terms in the dNTK definition (11.12) with $\ell_1 = \ell + 1$ and $\ell_2 < \ell + 1$, we find

$$\begin{aligned}
&\sum_{\ell_2=1}^{\ell} \sum_{j,k} \lambda_{W^{(\ell+1)}_{jk} W^{(\ell+1)}_{jk}} \sum_{\mu_2,\nu_2} \lambda^{(\ell_2)}_{\mu_2\nu_2} \frac{d^2 z^{(\ell+1)}_{i_0;\delta_0}}{dW^{(\ell+1)}_{jk} d\theta^{(\ell_2)}_{\mu_2}} \frac{dz^{(\ell+1)}_{i_1;\delta_1}}{dW^{(\ell+1)}_{jk}} \frac{dz^{(\ell+1)}_{i_2;\delta_2}}{d\theta^{(\ell_2)}_{\nu_2}} \\
=&\frac{\lambda^{(\ell+1)}_W}{n_\ell} \sum_{\ell_2=1}^{\ell} \sum_{j,k} \sum_{\mu_2,\nu_2} \lambda^{(\ell_2)}_{\mu_2\nu_2} \left(\delta_{i_0 j}\, \sigma'^{(\ell)}_{k;\delta_0} \frac{dz^{(\ell)}_{k;\delta_0}}{d\theta^{(\ell_2)}_{\mu_2}} \right) \left(\delta_{i_1 j}\, \sigma^{(\ell)}_{k;\delta_1} \right) \left(\sum_{k_2} W^{(\ell+1)}_{i_2 k_2} \sigma'^{(\ell)}_{k_2;\delta_2} \frac{dz^{(\ell)}_{k_2;\delta_2}}{d\theta^{(\ell_2)}_{\nu_2}} \right) \\
=&\frac{\lambda^{(\ell+1)}_W}{n_\ell} \delta_{i_0 i_1} \sum_{k_0,k_2} W^{(\ell+1)}_{i_2 k_2} \sigma'^{(\ell)}_{k_0;\delta_0} \sigma^{(\ell)}_{k_0;\delta_1} \sigma'^{(\ell)}_{k_2;\delta_2} \widehat{H}^{(\ell)}_{k_0 k_2;\delta_0\delta_2} \,.
\end{aligned} \tag{11.19}$$

To get this result, on the second line we implemented our choice of a single intralayer learning rate for the weights (8.6),

$$\lambda_{W^{(\ell+1)}_{j_1 k_1} W^{(\ell+1)}_{j_2 k_2}} = \delta_{j_1 j_2} \delta_{k_1 k_2} \frac{\lambda^{(\ell+1)}_W}{n_\ell} \,, \tag{11.20}$$

importantly rescaled by n_ℓ, and on the third line we used the definition of the stochastic NTK (11.7) and relabeled a dummy index. By symmetry, there must be a similar contribution when instead $\ell_2 = \ell + 1$ and $\ell_1 < \ell + 1$. This term is given by (11.19) after swapping neural-sample index pairs $(i_1, \delta_1) \leftrightarrow (i_2, \delta_2)$.

Third and finally, when both $\ell_1 < \ell+1$ and $\ell_2 < \ell+1$ both the biases and the weights contribute to the second derivative. When $\theta^{(\ell_1)}_\mu$ and $\theta^{(\ell_2)}_\nu$ are not from the $(\ell + 1)$-th layer, we computed their first derivative in (11.18), and their second derivative is given by

$$\frac{d^2 z^{(\ell+1)}_{i;\delta}}{d\theta^{(\ell_1)}_{\mu_1} d\theta^{(\ell_2)}_{\mu_2}} = \sum_k W^{(\ell+1)}_{ik} \sigma''^{(\ell)}_{k;\delta} \frac{dz^{(\ell)}_{k;\delta}}{d\theta^{(\ell_1)}_{\mu_1}} \frac{dz^{(\ell)}_{k;\delta}}{d\theta^{(\ell_2)}_{\mu_2}} + \sum_k W^{(\ell+1)}_{ik} \sigma'^{(\ell)}_{k;\delta} \frac{d^2 z^{(\ell)}_{k;\delta}}{d\theta^{(\ell_1)}_{\mu_1} d\theta^{(\ell_2)}_{\mu_2}} \,. \tag{11.21}$$

Multiplying these second derivative terms by the learning-rate tensors $\lambda^{(\ell_1)}_{\mu_1\nu_1}\, \lambda^{(\ell_2)}_{\mu_2\nu_2}$ and by the appropriate first derivatives (11.18), and implementing all the sums over ℓ_1, ℓ_2, μ_1, ν_1, μ_2, ν_2 in the dNTK definition (11.12), the first term from (11.21) gives a contribution of

$$\sum_{k_0,k_1,k_2} W^{(\ell+1)}_{i_0 k_0} W^{(\ell+1)}_{i_1 k_1} W^{(\ell+1)}_{i_2 k_2} \sigma''^{(\ell)}_{k_0;\delta_0} \sigma'^{(\ell)}_{k_1;\delta_1} \sigma'^{(\ell)}_{k_2;\delta_2} \widehat{H}^{(\ell)}_{k_0 k_1;\delta_0\delta_1} \widehat{H}^{(\ell)}_{k_0 k_2;\delta_0\delta_2} \,, \tag{11.22}$$

where we made use of the NTK definition (11.7) twice, while the second term from (11.21) gives a contribution of

$$\sum_{k_0,k_1,k_2} W^{(\ell+1)}_{i_0 k_0} W^{(\ell+1)}_{i_1 k_1} W^{(\ell+1)}_{i_2 k_2} \sigma'^{(\ell)}_{k_0;\delta_0} \sigma'^{(\ell)}_{k_1;\delta_1} \sigma'^{(\ell)}_{k_2;\delta_2} \widehat{\mathrm{d}H}^{(\ell)}_{k_0 k_1 k_2;\delta_0\delta_1\delta_2} \,, \tag{11.23}$$

where we made use of the dNTK definition (11.8) once.

Combining our three types of contributions (11.19), (11.22), and (11.23), we get a rather involved stochastic iteration equation:

$$\begin{aligned}\widehat{\mathrm{d}H}^{(\ell+1)}_{i_0 i_1 i_2;\delta_0\delta_1\delta_2} &= \sum_{k_0,k_1,k_2} W^{(\ell+1)}_{i_0 k_0} W^{(\ell+1)}_{i_1 k_1} W^{(\ell+1)}_{i_2 k_2} \sigma'^{(\ell)}_{k_0;\delta_0} \sigma'^{(\ell)}_{k_1;\delta_1} \sigma'^{(\ell)}_{k_2;\delta_2} \widehat{\mathrm{d}H}^{(\ell)}_{k_0 k_1 k_2;\delta_0\delta_1\delta_2} \\ &\quad + \sum_{k_0,k_1,k_2} W^{(\ell+1)}_{i_0 k_0} W^{(\ell+1)}_{i_1 k_1} W^{(\ell+1)}_{i_2 k_2} \sigma''^{(\ell)}_{k_0;\delta_0} \sigma'^{(\ell)}_{k_1;\delta_1} \sigma'^{(\ell)}_{k_2;\delta_2} \widehat{H}^{(\ell)}_{k_0 k_1;\delta_0\delta_1} \widehat{H}^{(\ell)}_{k_0 k_2;\delta_0\delta_2} \\ &\quad + \frac{\lambda_W^{(\ell+1)}}{n_\ell} \delta_{i_0 i_1} \sum_{k_0,k_2} W^{(\ell+1)}_{i_2 k_2} \sigma'^{(\ell)}_{k_0;\delta_0} \sigma^{(\ell)}_{k_0;\delta_1} \sigma'^{(\ell)}_{k_2;\delta_2} \widehat{H}^{(\ell)}_{k_0 k_2;\delta_0\delta_2} \\ &\quad + \frac{\lambda_W^{(\ell+1)}}{n_\ell} \delta_{i_0 i_2} \sum_{k_0,k_1} W^{(\ell+1)}_{i_1 k_1} \sigma'^{(\ell)}_{k_0;\delta_0} \sigma'^{(\ell)}_{k_1;\delta_1} \sigma^{(\ell)}_{k_0;\delta_2} \widehat{H}^{(\ell)}_{k_0 k_1;\delta_0\delta_1} \,. \end{aligned} \tag{11.24}$$

This is the **forward equation for the dNTK**, and we're next going to work out the recursions that determine its statistics.

11.2.1 First Layer: Zero dNTK

Recall from §4.1 and §8.1 that at initialization the first-layer preactivations,

$$z^{(1)}_{i;\delta} \equiv b^{(1)}_i + \sum_{k=1}^{n_0} W^{(1)}_{ik} x_{k;\delta} \,, \tag{11.25}$$

are distributed according to a zero-mean Gaussian distribution (4.23) and that the NTK $\widehat{H}^{(1)}_{i_1 i_2;\delta_1\delta_2} = \delta_{i_1 i_2} H^{(1)}_{\delta_1\delta_2}$ is deterministic (8.23).

As we discussed just before, since the preactivations are linear in the model parameters, their second derivative must vanish. Thus, the dNTK trivially vanishes in the first layer:

$$\widehat{\mathrm{d}H}^{(1)}_{i_0 i_1 i_2;\delta_0\delta_1\delta_2} \equiv \sum_{\substack{\mu_1,\nu_1,\\ \mu_2,\nu_2}} \lambda^{(1)}_{\mu_1\nu_1} \lambda^{(1)}_{\mu_2\nu_2} \frac{d^2 z^{(1)}_{i_0;\delta_0}}{d\theta^{(1)}_{\mu_1} d\theta^{(1)}_{\mu_2}} \frac{dz^{(1)}_{i_1;\delta_1}}{d\theta^{(1)}_{\nu_1}} \frac{dz^{(1)}_{i_2;\delta_2}}{d\theta^{(1)}_{\nu_2}} = 0 \,. \tag{11.26}$$

This gives the initial condition for our recursions.

Note that this result should have been expected as the first-layer NTK (8.23) is independent of the model parameters, and thus cannot change with any training. As we saw before for the first-layer preactivations and first-layer NTK – zero-mean Gaussian and fixed, respectively – this first-layer result for the dNTK will be representative of its infinite-width limit for all layers.

11.2.2 Second Layer: Nonzero dNTK

Now, let's analyze the dNTK (11.24) in the second layer. Remembering again that the first-layer NTK (8.23) is deterministic and diagonal in its neural indices as $\widehat{H}^{(1)}_{i_1 i_2;\delta_1\delta_2} =$

$\delta_{i_1 i_2} H^{(1)}_{\delta_1\delta_2}$, and remembering for the first time that the dNTK vanishes in the first layer $\widehat{\mathrm{d}H}^{(1)}_{i_0 i_1 i_2;\delta_0\delta_1\delta_2} = 0$ from (11.26), the forward equation (11.24) in the second layer simplifies to

$$\begin{aligned}\widehat{\mathrm{d}H}^{(2)}_{i_0 i_1 i_2;\delta_0\delta_1\delta_2} =& H^{(1)}_{\delta_0\delta_1} H^{(1)}_{\delta_0\delta_2} \sum_{k=1}^{n_1} W^{(2)}_{i_0 k} W^{(2)}_{i_1 k} W^{(2)}_{i_2 k} \sigma''^{(1)}_{k;\delta_0} \sigma'^{(1)}_{k;\delta_1} \sigma'^{(1)}_{k;\delta_2} \\ &+ \delta_{i_0 i_1} \frac{\lambda^{(2)}_W}{n_1} H^{(1)}_{\delta_0\delta_2} \sum_{k=1}^{n_1} W^{(2)}_{i_2 k} \sigma'^{(1)}_{k;\delta_0} \sigma^{(1)}_{k;\delta_1} \sigma'^{(1)}_{k;\delta_2} \\ &+ \delta_{i_0 i_2} \frac{\lambda^{(2)}_W}{n_1} H^{(1)}_{\delta_0\delta_1} \sum_{k=1}^{n_1} W^{(2)}_{i_1 k} \sigma'^{(1)}_{k;\delta_0} \sigma'^{(1)}_{k;\delta_1} \sigma^{(1)}_{k;\delta_2} \,. \end{aligned} \tag{11.27}$$

Interestingly, since each term has an odd number of weights, the mean of the dNTK will vanish, and we'll have to look at cross correlations to find leading dNTK statistics that are non-vanishing.

The simplest cross correlation is with a single preactivation. Considering the product of the second-layer dNTK (11.27) with second-layer preactivations,

$$z^{(2)}_{i;\delta} = b^{(2)}_i + \sum_{k=1}^{n_1} W^{(2)}_{ik} \sigma^{(1)}_{k;\delta} \,, \tag{11.28}$$

and taking an expectation, we find

$$\begin{aligned}&\mathbb{E}\left[\widehat{\mathrm{d}H}^{(2)}_{i_0 i_1 i_2;\delta_0\delta_1\delta_2} z^{(2)}_{i_3;\delta_3}\right] \\ =& H^{(1)}_{\delta_0\delta_1} H^{(1)}_{\delta_0\delta_2} \left(\frac{C^{(2)}_W}{n_1}\right)^2 (\delta_{i_0 i_3}\delta_{i_1 i_2} + \delta_{i_0 i_1}\delta_{i_2 i_3} + \delta_{i_0 i_2}\delta_{i_1 i_3}) \sum_{k=1}^{n_1} \mathbb{E}\left[\sigma''^{(1)}_{k;\delta_0} \sigma'^{(1)}_{k;\delta_1} \sigma'^{(1)}_{k;\delta_2} \sigma^{(1)}_{k;\delta_3}\right] \\ &+ \frac{\lambda^{(2)}_W}{n_1} H^{(1)}_{\delta_0\delta_2} \delta_{i_0 i_1}\delta_{i_2 i_3} \frac{C^{(2)}_W}{n_1} \sum_{k=1}^{n_1} \mathbb{E}\left[\sigma'^{(1)}_{k;\delta_0} \sigma^{(1)}_{k;\delta_1} \sigma'^{(1)}_{k;\delta_2} \sigma^{(1)}_{k;\delta_3}\right] \\ &+ \frac{\lambda^{(2)}_W}{n_1} H^{(1)}_{\delta_0\delta_1} \delta_{i_0 i_2}\delta_{i_1 i_3} \frac{C^{(2)}_W}{n_1} \sum_{k=1}^{n_1} \mathbb{E}\left[\sigma'^{(1)}_{k;\delta_0} \sigma'^{(1)}_{k;\delta_1} \sigma^{(1)}_{k;\delta_2} \sigma^{(1)}_{k;\delta_3}\right] \\ =& \frac{1}{n_1} (\delta_{i_0 i_3}\delta_{i_1 i_2} + \delta_{i_0 i_1}\delta_{i_2 i_3} + \delta_{i_0 i_2}\delta_{i_1 i_3}) C^{(2)}_W H^{(1)}_{\delta_0\delta_1} C^{(2)}_W H^{(1)}_{\delta_0\delta_2} \left\langle \sigma''_{\delta_0} \sigma'_{\delta_1} \sigma'_{\delta_2} \sigma_{\delta_3} \right\rangle_{G^{(1)}} \\ &+ \frac{1}{n_1} \delta_{i_0 i_1}\delta_{i_2 i_3} \lambda^{(2)}_W C^{(2)}_W H^{(1)}_{\delta_0\delta_2} \left\langle \sigma'_{\delta_0} \sigma_{\delta_1} \sigma'_{\delta_2} \sigma_{\delta_3} \right\rangle_{G^{(1)}} \\ &+ \frac{1}{n_1} \delta_{i_0 i_2}\delta_{i_1 i_3} \lambda^{(2)}_W C^{(2)}_W H^{(1)}_{\delta_0\delta_1} \left\langle \sigma'_{\delta_0} \sigma'_{\delta_1} \sigma_{\delta_2} \sigma_{\delta_3} \right\rangle_{G^{(1)}} \end{aligned} \tag{11.29}$$

To get this final result, in the first equality we dropped the bias term from (11.28), since it vanishes under the expectation, and performed various Wick contractions of the weights using $\mathbb{E}\left[W^{(2)}_{i_1 j_1} W^{(2)}_{i_2 j_2}\right] = \delta_{i_1 i_2}\delta_{j_1 j_2} C^{(2)}_W / n_1$ (2.20). For the second equality, we remembered that the first-layer preactivation distribution is a zero-mean Gaussian with a two-point

correlator that's diagonal in neural indices, $\mathbb{E}\left[z^{(1)}_{i_1;\delta_1} z^{(1)}_{i_2;\delta_2}\right] = \delta_{i_1 i_2} G^{(1)}_{\delta_1\delta_2}$ (4.23), and used this to swap full expectations for Gaussian expectations and then performed the sums.

As we did before for the NTK variance (8.31) and the NTK-preactivation cross correlation (8.37), it is convenient to decompose this dNTK-preactivation cross correlation (11.29) into two tensors with sample indices only:

$$\mathbb{E}\left[\widehat{\mathrm{d}H}^{(2)}_{i_0 i_1 i_2;\delta_0\delta_1\delta_2} z^{(2)}_{i_3;\delta_3}\right] \equiv \frac{1}{n_1}\left[\delta_{i_0 i_3}\delta_{i_1 i_2} P^{(2)}_{\delta_0\delta_1\delta_2\delta_3} + \delta_{i_0 i_1}\delta_{i_2 i_3} Q^{(2)}_{\delta_0\delta_1\delta_2\delta_3} + \delta_{i_0 i_2}\delta_{i_1 i_3} Q^{(2)}_{\delta_0\delta_2\delta_1\delta_3}\right] . \tag{11.30}$$

Comparing with our explicit formula for the second-layer cross correlation (11.29), we see that these tensors have the following definitions,

$$P^{(2)}_{\delta_0\delta_1\delta_2\delta_3} \equiv \left(C^{(2)}_W\right)^2 H^{(1)}_{\delta_0\delta_1} H^{(1)}_{\delta_0\delta_2} \left\langle \sigma''_{\delta_0}\sigma'_{\delta_1}\sigma'_{\delta_2}\sigma_{\delta_3}\right\rangle_{G^{(1)}} , \tag{11.31}$$

$$Q^{(2)}_{\delta_0\delta_1\delta_2\delta_3} \equiv \left(C^{(2)}_W\right)^2 H^{(1)}_{\delta_0\delta_1} H^{(1)}_{\delta_0\delta_2} \left\langle \sigma''_{\delta_0}\sigma'_{\delta_1}\sigma'_{\delta_2}\sigma_{\delta_3}\right\rangle_{G^{(1)}} + \lambda^{(2)}_W C^{(2)}_W H^{(1)}_{\delta_0\delta_2} \left\langle \sigma'_{\delta_0}\sigma_{\delta_1}\sigma'_{\delta_2}\sigma_{\delta_3}\right\rangle_{G^{(1)}} , \tag{11.32}$$

and that they are manifestly of order one.[3] Overall, the dNTK-preactivation cross correlation (11.30) is of order $1/n_1$, vanishing in the strict infinite-width limit $n_1 \to \infty$.

These tensors (11.31) and (11.32) – and their deeper-layer siblings – control all the leading finite-width correlation between the preactivations and the dNTK, and in fact encapsulate the entire effect of the dNTK at order $1/n$. As we'll show next, any other dNTK-preactivation cross correlators, e.g. $\mathbb{E}\left[\widehat{\mathrm{d}H}^{(\ell)}_{i_0 i_1 i_2;\delta_0\delta_1\delta_2} z^{(\ell)}_{j_3;\delta_3} z^{(\ell)}_{j_4;\delta_4} z^{(\ell)}_{j_5;\delta_5}\right]$, can always be expressed in terms of a combination of $P^{(\ell)}$ and $Q^{(\ell)}$ at this order.

11.2.3 Deeper Layers: Growing dNTK

As before with §4.1 ‖ §8.1 ‖ §11.2.1 and §4.2 ‖ §8.2 ‖ §11.2.2, this section parallels our other sections analyzing the RG flow in deeper layers (§4.3 ‖ §8.3).

To proceed forward, we'll first need to evaluate an interlayer formula with three weight insertions (extending our work in §8.3.0), and then we'll immediately put it to use in order to obtain recursions for the dNTK-preactivation cross correlation.

*Inter*lude 2: *Inter*layer Correlations Reloaded

Since the forward equation for the dNTK (11.24) has terms with one or three $(\ell+1)$-th-layer weight matrices, we'll need interlayer formulae with one or three weight insertions.

[3] The cross-correlation tensor $P^{(2)}_{\delta_0\delta_1\delta_2\delta_3}$ (11.31) – and more generally $P^{(\ell)}_{\delta_0\delta_1\delta_2\delta_3}$ in deeper layers – is only symmetric under the exchange of its middle samples indices $\delta_1 \leftrightarrow \delta_2$. Meanwhile, it's manifestly clear from (11.32) that the other cross-correlation tensor $Q^{(2)}_{\delta_0\delta_1\delta_2\delta_3}$ has no symmetry whatsoever.

Let's start by recalling our generating function for interlayer correlations (8.53):

$$\mathbb{E}\left[\mathcal{O}\Big(z^{(\ell+1)}\Big)\, e^{\sum_{i,j}\mathcal{J}_{ij}W_{ij}^{(\ell+1)}}\mathcal{Q}\Big(z^{(\ell)},\widehat{H}^{(\ell)},\widehat{\mathrm{d}H}^{(\ell)}\Big)\right] \tag{11.33}$$
$$=\exp\left(\frac{C_W^{(\ell+1)}}{2n_\ell}\sum_{i,j}\mathcal{J}_{ij}^2\right)\mathbb{E}\left[\left\langle\!\!\left\langle \mathcal{O}\Big(z_{i;\delta}^{(\ell+1)}+\frac{C_W^{(\ell+1)}}{n_\ell}\sum_{j=1}^{n_\ell}\mathcal{J}_{ij}\sigma_{j;\delta}^{(\ell)}\Big)\right\rangle\!\!\right\rangle_{\widehat{G}^{(\ell+1)}}\mathcal{Q}\Big(z^{(\ell)},\widehat{H}^{(\ell)},\widehat{\mathrm{d}H}^{(\ell)}\Big)\right].$$

In this formula, $\mathcal{O}$ is a generic function of $(\ell+1)$-th-layer preactivations only, and $\mathcal{Q}$ is a function of any of our ℓ-th-layer objects; in particular, since the original derivation of this formula didn't depend on specifics of $\mathcal{Q}$, we've also included the ℓ-th-layer dNTK as part of its argument.

With that in mind, we first wrote down an interlayer formula with one insertion as (10.10) when analyzing (the lack of) representation learning in the infinite-width limit. To save you the need to flip back and refresh your memory, we'll reprint it here after making some minor notational adjustments:

$$\mathbb{E}\left[\mathcal{O}\Big(z^{(\ell+1)}\Big)\, W_{ij}^{(\ell+1)}\mathcal{Q}\Big(z^{(\ell)},\widehat{H}^{(\ell)},\widehat{\mathrm{d}H}^{(\ell)}\Big)\right] \tag{11.34}$$
$$=\frac{C_W^{(\ell+1)}}{n_\ell}\sum_{\delta\in\mathcal{D}}\mathbb{E}\left[\left\langle\!\!\left\langle\frac{\partial\mathcal{O}}{\partial z_{i;\delta}^{(\ell+1)}}\right\rangle\!\!\right\rangle_{\widehat{G}^{(\ell+1)}}\sigma_{j;\delta}^{(\ell)}\,\mathcal{Q}\Big(z^{(\ell)},\widehat{H}^{(\ell)},\widehat{\mathrm{d}H}^{(\ell)}\Big)\right].$$

This can be readily rederived by differentiating the generating function (11.33) with respect to the source as $\frac{d}{d\mathcal{J}_{ij}}$ once and then setting the source to zero.

In contrast, three-weight insertion formula will be new. Thrice-differentiating the generating function (11.33) with respect to the source as $\frac{d}{d\mathcal{J}_{i_0j_0}}\frac{d}{d\mathcal{J}_{i_1j_1}}\frac{d}{d\mathcal{J}_{i_2j_2}}$ and then setting the source to zero $\mathcal{J}=0$, we find

$$\mathbb{E}\left[\mathcal{O}\Big(z^{(\ell+1)}\Big)\, W_{i_0j_0}^{(\ell+1)}W_{i_1j_1}^{(\ell+1)}W_{i_2j_2}^{(\ell+1)}\mathcal{Q}\Big(z^{(\ell)},\widehat{H}^{(\ell)},\widehat{\mathrm{d}H}^{(\ell)}\Big)\right] \tag{11.35}$$
$$\begin{aligned}
&=\left(\frac{C_W^{(\ell+1)}}{n_\ell}\right)^2\delta_{i_0i_1}\delta_{j_0j_1}\sum_{\delta\in\mathcal{D}}\mathbb{E}\left[\left\langle\!\!\left\langle\frac{\partial\mathcal{O}}{\partial z_{i_2;\delta}^{(\ell+1)}}\right\rangle\!\!\right\rangle_{\widehat{G}^{(\ell+1)}}\sigma_{j_2;\delta}^{(\ell)}\,\mathcal{Q}\Big(z^{(\ell)},\widehat{H}^{(\ell)},\widehat{\mathrm{d}H}^{(\ell)}\Big)\right]\\
&\quad+\left(\frac{C_W^{(\ell+1)}}{n_\ell}\right)^2\delta_{i_0i_2}\delta_{j_0j_2}\sum_{\delta\in\mathcal{D}}\mathbb{E}\left[\left\langle\!\!\left\langle\frac{\partial\mathcal{O}}{\partial z_{i_1;\delta}^{(\ell+1)}}\right\rangle\!\!\right\rangle_{\widehat{G}^{(\ell+1)}}\sigma_{j_1;\delta}^{(\ell)}\,\mathcal{Q}\Big(z^{(\ell)},\widehat{H}^{(\ell)},\widehat{\mathrm{d}H}^{(\ell)}\Big)\right]\\
&\quad+\left(\frac{C_W^{(\ell+1)}}{n_\ell}\right)^2\delta_{i_1i_2}\delta_{j_1j_2}\sum_{\delta\in\mathcal{D}}\mathbb{E}\left[\left\langle\!\!\left\langle\frac{\partial\mathcal{O}}{\partial z_{i_0;\delta}^{(\ell+1)}}\right\rangle\!\!\right\rangle_{\widehat{G}^{(\ell+1)}}\sigma_{j_0;\delta}^{(\ell)}\,\mathcal{Q}\Big(z^{(\ell)},\widehat{H}^{(\ell)},\widehat{\mathrm{d}H}^{(\ell)}\Big)\right]\\
&\quad+\left(\frac{C_W^{(\ell+1)}}{n_\ell}\right)^3\sum_{\delta_0,\delta_1,\delta_2\in\mathcal{D}}\mathbb{E}\left[\left\langle\!\!\left\langle\frac{\partial^3\mathcal{O}}{\partial z_{i_0;\delta_0}^{(\ell+1)}\partial z_{i_1;\delta_1}^{(\ell+1)}\partial z_{i_2;\delta_2}^{(\ell+1)}}\right\rangle\!\!\right\rangle_{\widehat{G}^{(\ell+1)}}\sigma_{j_0;\delta_0}^{(\ell)}\sigma_{j_1;\delta_1}^{(\ell)}\sigma_{j_2;\delta_2}^{(\ell)}\mathcal{Q}\Big(z^{(\ell)},\widehat{H}^{(\ell)},\widehat{\mathrm{d}H}^{(\ell)}\Big)\right].
\end{aligned}$$

Intuitively, we can understand this formula as follows: each of the first three terms comes from forming one Wick contraction with two of the weight insertions and then

forming another contraction between the remaining weight and a weight hidden inside the $z^{(\ell+1)}$ in $\mathcal{O}$, while the final term comes from all three weight insertions each forming a contraction with other weights inside the observable $\mathcal{O}$.

dNTK-Preactivation Cross Correlations

Let's first use the interlayer formulae derived above to show that all of the dNTK's contributions to the statistics of the joint distribution $p\Big(z^{(\ell)}, \widehat{H}^{(\ell)}, \widehat{\mathrm{d}H}^{(\ell)}\Big|\mathcal{D}\Big)$ at order $1/n$ are captured by the cross correlation of the dNTK with a single preactivation, $\mathbb{E}\left[\widehat{\mathrm{d}H}^{(\ell)}_{i_0 i_1 i_2;\delta_0\delta_1\delta_2} z^{(\ell+1)}_{i_3;\delta_3}\right]$. To that end, we'll examine a dNTK-preactivation cross correlator of a very general form:

$$
\begin{aligned}
&\mathbb{E}\left[\mathcal{O}\Big(z^{(\ell+1)}\Big)\ \widehat{\mathrm{d}H}^{(\ell+1)}_{i_0 i_1 i_2;\delta_0\delta_1\delta_2}\right] \qquad (11.36)\\
=&\delta_{i_0 i_1}\frac{\lambda_W^{(\ell+1)}}{n_\ell}\sum_{k_0,k_2}\mathbb{E}\left[\mathcal{O}\Big(z^{(\ell+1)}\Big)\, W^{(\ell+1)}_{i_2 k_2}\sigma'^{(\ell)}_{k_0;\delta_0}\sigma^{(\ell)}_{k_0;\delta_1}\sigma'^{(\ell)}_{k_2;\delta_2}\widehat{H}^{(\ell)}_{k_0 k_2;\delta_0\delta_2}\right]\\
&+\delta_{i_0 i_2}\frac{\lambda_W^{(\ell+1)}}{n_\ell}\sum_{k_0,k_1}\mathbb{E}\left[\mathcal{O}\Big(z^{(\ell+1)}\Big)\, W^{(\ell+1)}_{i_1 k_1}\sigma'^{(\ell)}_{k_0;\delta_0}\sigma'^{(\ell)}_{k_1;\delta_1}\sigma^{(\ell)}_{k_0;\delta_2}\widehat{H}^{(\ell)}_{k_0 k_1;\delta_0\delta_1}\right]\\
&+\sum_{k_0,k_1,k_2}\mathbb{E}\left[\mathcal{O}\Big(z^{(\ell+1)}\Big)\, W^{(\ell+1)}_{i_0 k_0}W^{(\ell+1)}_{i_1 k_1}W^{(\ell+1)}_{i_2 k_2}\sigma''^{(\ell)}_{k_0;\delta_0}\sigma'^{(\ell)}_{k_1;\delta_1}\sigma'^{(\ell)}_{k_2;\delta_2}\widehat{H}^{(\ell)}_{k_0 k_1;\delta_0\delta_1}\widehat{H}^{(\ell)}_{k_0 k_2;\delta_0\delta_2}\right]\\
&+\sum_{k_0,k_1,k_2}\mathbb{E}\left[\mathcal{O}\Big(z^{(\ell+1)}\Big)\, W^{(\ell+1)}_{i_0 k_0}W^{(\ell+1)}_{i_1 k_1}W^{(\ell+1)}_{i_2 k_2}\sigma'^{(\ell)}_{k_0;\delta_0}\sigma'^{(\ell)}_{k_1;\delta_1}\sigma'^{(\ell)}_{k_2;\delta_2}\widehat{\mathrm{d}H}^{(\ell)}_{k_0 k_1 k_2;\delta_0\delta_1\delta_2}\right].
\end{aligned}
$$

Here, we took the expectation of the dNTK forward equation (11.24) multiplied by a generic observable $\mathcal{O}\Big(z^{(\ell+1)}\Big)$ of $(\ell+1)$-th-layer preactivations. We also ordered the four terms to reflect the order in which we will subsequently evaluate them.

First, let's simplify the first two terms. Using our interlayer formula with one weight insertion (11.34) on the first term in (11.36), we get

$$
\begin{aligned}
&\frac{\lambda_W^{(\ell+1)}}{n_\ell}\delta_{i_0 i_1}\sum_{k_0,k_2}\mathbb{E}\left[\mathcal{O}\Big(z^{(\ell+1)}\Big)\, W^{(\ell+1)}_{i_2 k_2}\sigma'^{(\ell)}_{k_0;\delta_0}\sigma^{(\ell)}_{k_0;\delta_1}\sigma'^{(\ell)}_{k_2;\delta_2}\widehat{H}^{(\ell)}_{k_0 k_2;\delta_0\delta_2}\right] \qquad (11.37)\\
=&\frac{\lambda_W^{(\ell+1)}C_W^{(\ell+1)}}{n_\ell^2}\sum_{k_0,k_2}\sum_{\delta\in\mathcal{D}}\delta_{i_0 i_1}\mathbb{E}\left[\left\langle\!\!\left\langle\frac{\partial\mathcal{O}}{\partial z^{(\ell+1)}_{i_2;\delta}}\right\rangle\!\!\right\rangle_{\widehat{G}^{(\ell+1)}}\sigma^{(\ell)}_{k_2;\delta}\sigma'^{(\ell)}_{k_0;\delta_0}\sigma^{(\ell)}_{k_0;\delta_1}\sigma'^{(\ell)}_{k_2;\delta_2}\widehat{H}^{(\ell)}_{k_0 k_2;\delta_0\delta_2}\right]\\
=&\frac{\lambda_W^{(\ell+1)}C_W^{(\ell+1)}}{n_\ell^2}\sum_{k,m}\sum_{\delta\in\mathcal{D}}\delta_{i_0 i_1}\left\langle\!\!\left\langle\frac{\partial\mathcal{O}}{\partial z^{(\ell+1)}_{i_2;\delta}}\right\rangle\!\!\right\rangle_{G^{(\ell+1)}}\mathbb{E}\left[\sigma^{(\ell)}_{k;\delta_1}\sigma^{(\ell)}_{m;\delta}\sigma'^{(\ell)}_{k;\delta_0}\sigma'^{(\ell)}_{m;\delta_2}\widehat{H}^{(\ell)}_{km;\delta_0\delta_2}\right]+O\Big(\frac{1}{n^2}\Big)\\
=&\frac{1}{n_\ell}\frac{\lambda_W^{(\ell+1)}}{C_W^{(\ell+1)}}\sum_{\delta\in\mathcal{D}}\delta_{i_0 i_1}\left\langle\!\!\left\langle\frac{\partial\mathcal{O}}{\partial z^{(\ell+1)}_{i_2;\delta}}\right\rangle\!\!\right\rangle_{G^{(\ell+1)}}F^{(\ell+1)}_{\delta_1\delta_0\delta_3\delta_2}+O\Big(\frac{1}{n^2}\Big)\,,
\end{aligned}
$$

where in the third line we used the Schwinger-Dyson formula (8.70) to expand the metric fluctuation around its mean $G^{(\ell+1)}$, noting that the second term with the fluctuating is subleading, and in the last line we identified the remaining expectation as the definition of the NTK-preactivation cross correlation tensor $F^{(\ell+1)}$ from (8.69) up to constants.[4]

Similarly, for the second term in (11.36) we get an identical contribution up to the swapping of neural-sample index pairs as $(i_1, \delta_1) \leftrightarrow (i_2, \delta_2)$.

Next, let's tackle the third term in (11.36). Applying our interlayer formula with three weight insertions (11.35), we get

$$\begin{aligned}
&\sum_{k_0,k_1,k_2} \mathbb{E}\left[\mathcal{O}\left(z^{(\ell+1)}\right) W^{(\ell+1)}_{i_0k_0} W^{(\ell+1)}_{i_1k_1} W^{(\ell+1)}_{i_2k_2} \sigma''^{(\ell)}_{k_0;\delta_0} \sigma'^{(\ell)}_{k_1;\delta_1} \sigma'^{(\ell)}_{k_2;\delta_2} \widehat{H}^{(\ell)}_{k_0k_1;\delta_0\delta_1} \widehat{H}^{(\ell)}_{k_0k_2;\delta_0\delta_2}\right] \\
=&\frac{\left(C_W^{(\ell+1)}\right)^2}{n_\ell} \delta_{i_0i_1} \sum_{\delta\in\mathcal{D}} \left\langle\!\!\left\langle \frac{\partial\mathcal{O}}{\partial z^{(\ell+1)}_{i_2;\delta}} \right\rangle\!\!\right\rangle_{G^{(\ell+1)}} \left\{ \frac{1}{n_\ell} \sum_{k,m} \mathbb{E}\left[\sigma^{(\ell)}_{m;\delta} \sigma''^{(\ell)}_{k;\delta_0} \sigma'^{(\ell)}_{k;\delta_1} \sigma'^{(\ell)}_{m;\delta_2} \widehat{H}^{(\ell)}_{kk;\delta_0\delta_1} \widehat{H}^{(\ell)}_{km;\delta_0\delta_2}\right] \right\} \\
&+ \frac{\left(C_W^{(\ell+1)}\right)^2}{n_\ell} \delta_{i_0i_2} \sum_{\delta\in\mathcal{D}} \left\langle\!\!\left\langle \frac{\partial\mathcal{O}}{\partial z^{(\ell+1)}_{i_1;\delta}} \right\rangle\!\!\right\rangle_{G^{(\ell+1)}} \left\{ \frac{1}{n_\ell} \sum_{k,m} \mathbb{E}\left[\sigma^{(\ell)}_{m;\delta} \sigma''^{(\ell)}_{k;\delta_0} \sigma'^{(\ell)}_{k;\delta_2} \sigma'^{(\ell)}_{m;\delta_1} \widehat{H}^{(\ell)}_{kk;\delta_0\delta_2} \widehat{H}^{(\ell)}_{km;\delta_0\delta_1}\right] \right\} \\
&+ \frac{\left(C_W^{(\ell+1)}\right)^2}{n_\ell} \delta_{i_1i_2} \sum_{\delta\in\mathcal{D}} \left\langle\!\!\left\langle \frac{\partial\mathcal{O}}{\partial z^{(\ell+1)}_{i_0;\delta}} \right\rangle\!\!\right\rangle_{G^{(\ell+1)}} \left\{ \frac{1}{n_\ell} \sum_{k,m} \mathbb{E}\left[\sigma^{(\ell)}_{m;\delta} \sigma''^{(\ell)}_{m;\delta_0} \sigma'^{(\ell)}_{k;\delta_1} \sigma'^{(\ell)}_{k;\delta_2} \widehat{H}^{(\ell)}_{mk;\delta_0\delta_1} \widehat{H}^{(\ell)}_{mk;\delta_0\delta_2}\right] \right\} \\
&+ O\left(\frac{1}{n^2}\right) ,
\end{aligned} \tag{11.38}$$

where for each term we again used the Schwinger-Dyson formula (8.70) to expand the metric fluctuation around its mean $G^{(\ell+1)}$, picking up the leading contribution from mean metric, and then took the Gaussian expectation outside the full ℓ-th-layer expectation. Note that the final would-be term proportional to $\partial^3\mathcal{O}$ is subleading:

$$\begin{aligned}
&\left(C_W^{(\ell+1)}\right)^3 \sum_{\delta,\delta_4,\delta_5\in\mathcal{D}} \left\langle\!\!\left\langle \frac{\partial^3\mathcal{O}}{\partial z^{(\ell+1)}_{i_0;\delta} \partial z^{(\ell+1)}_{i_1;\delta_4} \partial z^{(\ell+1)}_{i_2;\delta_5}} \right\rangle\!\!\right\rangle_{G^{(\ell+1)}} \\
&\quad\times \frac{1}{n_\ell^3} \sum_{k_0,k_1,k_2} \mathbb{E}\left[\sigma^{(\ell)}_{k_0;\delta} \sigma^{(\ell)}_{k_1;\delta_4} \sigma^{(\ell)}_{k_2;\delta_5} \sigma''^{(\ell)}_{k_0;\delta_0} \sigma'^{(\ell)}_{k_1;\delta_1} \sigma'^{(\ell)}_{k_2;\delta_2} \widehat{H}^{(\ell)}_{k_0k_1;\delta_0\delta_1} \widehat{H}^{(\ell)}_{k_0k_2;\delta_0\delta_2}\right] = O\left(\frac{1}{n^2}\right) .
\end{aligned} \tag{11.39}$$

To see why, decompose each stochastic NTK into a mean and fluctuation as $\widehat{H}^{(\ell)}_{i_1i_2;\delta_1\delta_2} = \delta_{i_1i_2} H^{(\ell)}_{\delta_1\delta_2} + \widehat{\Delta H}^{(\ell)}_{i_1i_2;\delta_1\delta_2}$ (8.58) and evaluate resulting four terms. In each case, you'll find the terms are $O(1/n^2)$ suppressed due to the Kronecker deltas constraining the triple sum and/or the additional $1/n$ suppression coming from the fluctuations.

Lastly, let's process the fourth and final term in our general cross correlation (11.36). Again applying our interlayer formula with three weight insertions (11.35) and taking

[4] Note that this is an $(\ell+1)$-th-layer quantity, rather than an ℓ-th-layer quantity as we typically have on the right-hand side of recursions. If you prefer, you can use the F-recursion (8.79) to re-express it in terms of a more complicated collection of ℓ-th-layer quantities.

only the leading-order pieces, we get

$$
\begin{aligned}
&\sum_{k_0,k_1,k_2} \mathbb{E}\left[\mathcal{O}\left(z^{(\ell+1)}\right) W^{(\ell+1)}_{i_0k_0} W^{(\ell+1)}_{i_1k_1} W^{(\ell+1)}_{i_2k_2} \sigma'^{(\ell)}_{k_0;\delta_0} \sigma'^{(\ell)}_{k_1;\delta_1} \sigma'^{(\ell)}_{k_2;\delta_2} \widehat{\mathrm{d}H}^{(\ell)}_{k_0k_1k_2;\delta_0\delta_1\delta_2}\right] \\
=&\frac{\left(C_W^{(\ell+1)}\right)^2}{n_\ell} \sum_{\delta\in\mathcal{D}} \delta_{i_0i_1} \left\langle\!\!\left\langle \frac{\partial\mathcal{O}}{\partial z^{(\ell+1)}_{i_2;\delta}} \right\rangle\!\!\right\rangle_{G^{(\ell+1)}} \left\{ \frac{1}{n_\ell} \sum_{k,m} \mathbb{E}\left[\sigma'^{(\ell)}_{k;\delta_0} \sigma'^{(\ell)}_{k;\delta_1} \sigma'^{(\ell)}_{m;\delta_2} \sigma^{(\ell)}_{m;\delta} \widehat{\mathrm{d}H}^{(\ell)}_{kkm;\delta_0\delta_1\delta_2}\right] \right\} \\
&+ \frac{\left(C_W^{(\ell+1)}\right)^2}{n_\ell} \sum_{\delta\in\mathcal{D}} \delta_{i_0i_2} \left\langle\!\!\left\langle \frac{\partial\mathcal{O}}{\partial z^{(\ell+1)}_{i_1;\delta}} \right\rangle\!\!\right\rangle_{G^{(\ell+1)}} \left\{ \frac{1}{n_\ell} \sum_{k,m} \mathbb{E}\left[\sigma'^{(\ell)}_{k;\delta_0} \sigma'^{(\ell)}_{m;\delta_1} \sigma'^{(\ell)}_{k;\delta_2} \sigma^{(\ell)}_{m;\delta} \widehat{\mathrm{d}H}^{(\ell)}_{kmk;\delta_0\delta_1\delta_2}\right] \right\} \\
&+ \frac{\left(C_W^{(\ell+1)}\right)^2}{n_\ell} \sum_{\delta\in\mathcal{D}} \delta_{i_1i_2} \left\langle\!\!\left\langle \frac{\partial\mathcal{O}}{\partial z^{(\ell+1)}_{i_0;\delta}} \right\rangle\!\!\right\rangle_{G^{(\ell+1)}} \left\{ \frac{1}{n_\ell} \sum_{k,m} \mathbb{E}\left[\sigma'^{(\ell)}_{m;\delta_0} \sigma'^{(\ell)}_{k;\delta_1} \sigma'^{(\ell)}_{k;\delta_2} \sigma^{(\ell)}_{m;\delta} \widehat{\mathrm{d}H}^{(\ell)}_{mkk;\delta_0\delta_1\delta_2}\right] \right\} \\
&+ O\left(\frac{1}{n^2}\right) .
\end{aligned}
\tag{11.40}
$$

Note that the factors in the curly brackets are actually of order one since – as we saw in the second layer and will recursively show next for general layers ℓ – the leading dNTK-preactivation cross correlation is of order $1/n$. Meanwhile, again the final would-be term proportional to $\partial^3\mathcal{O}$ is subleading:

$$
\begin{aligned}
&\left(C_W^{(\ell+1)}\right)^3 \sum_{\delta_3,\delta_4,\delta_5\in\mathcal{D}} \left\langle\!\!\left\langle \frac{\partial^3\mathcal{O}}{\partial z^{(\ell+1)}_{i_0;\delta_3} \partial z^{(\ell+1)}_{i_1;\delta_4} \partial z^{(\ell+1)}_{i_2;\delta_5}} \right\rangle\!\!\right\rangle_{K^{(\ell+1)}} \\
&\quad\times \frac{1}{n_\ell^3} \sum_{k_0,k_1,k_2} \mathbb{E}\left[\left(\sigma'^{(\ell)}_{k_0;\delta_0} \sigma^{(\ell)}_{k_0;\delta_3}\right) \left(\sigma'^{(\ell)}_{k_1;\delta_1} \sigma^{(\ell)}_{k_1;\delta_4}\right) \left(\sigma'^{(\ell)}_{k_2;\delta_2} \sigma^{(\ell)}_{k_2;\delta_5}\right) \widehat{\mathrm{d}H}^{(\ell)}_{k_0k_1k_2;\delta_0\delta_1\delta_2}\right] = O\left(\frac{1}{n^2}\right) .
\end{aligned}
\tag{11.41}
$$

To see why, note that the expectation is another dNTK-preactivation cross correlator and thus is at most of order $1/n$. Further, we only get such an order-$1/n$ contribution when two out of three neural indices k_0, k_1, k_2 coincide: this should be clear from the pattern of Kronecker deltas that arise when we evaluate such dNTK-preactivation cross correlations in terms of our *P-Q* decomposition; cf. (11.30) for the second layer, or look ahead a paragraph to (11.42) for deeper layers. This means that the sum over all three neural indices will be restricted to be only two independent sums and, taking the prefactor of $1/n^3$ into account, the overall contribution of this term will thus go as $\sim (1/n^3)(n^2)(1/n) \sim 1/n^2$.

Substituting all our evaluated contributions (11.37) (twice), (11.38), and (11.40) into

our expression for a general dNTK-preactivation cross correlator (11.36), we get

$$\begin{aligned}
&\mathbb{E}\left[\mathcal{O}\Big(z^{(\ell+1)}\Big)\,\widehat{\mathrm{d}H}^{(\ell+1)}_{i_0 i_1 i_2;\delta_0\delta_1\delta_2}\right] \qquad (11.42)\\
=&\frac{1}{n_\ell}\sum_{\delta\in\mathcal{D}}\left[\delta_{i_1 i_2}\left\langle\!\!\left\langle \frac{\partial\mathcal{O}}{\partial z^{(\ell+1)}_{i_0;\delta}}\right\rangle\!\!\right\rangle_{G^{(\ell+1)}} P^{(\ell+1)}_{\delta_0\delta_1\delta_2\delta}\right.\\
&\qquad\left.+\,\delta_{i_0 i_1}\left\langle\!\!\left\langle \frac{\partial\mathcal{O}}{\partial z^{(\ell+1)}_{i_2;\delta}}\right\rangle\!\!\right\rangle_{G^{(\ell+1)}} Q^{(\ell+1)}_{\delta_0\delta_1\delta_2\delta}+\delta_{i_0 i_2}\left\langle\!\!\left\langle \frac{\partial\mathcal{O}}{\partial z^{(\ell+1)}_{i_1;\delta}}\right\rangle\!\!\right\rangle_{G^{(\ell+1)}} Q^{(\ell+1)}_{\delta_0\delta_2\delta_1\delta}\right]+O\Big(\frac{1}{n^2}\Big),
\end{aligned}$$

where we've introduced the $(\ell+1)$-th-layer generalizations of the second-layer tensors $P^{(2)}$ (11.31) and $Q^{(2)}$ (11.32):

$$\begin{aligned}
P^{(\ell+1)}_{\delta_0\delta_1\delta_2\delta_3} \equiv&\Big(C_W^{(\ell+1)}\Big)^2\frac{1}{n_\ell}\sum_{k,m=1}^{n_\ell}\mathbb{E}\left[\sigma''^{(\ell)}_{m;\delta_0}\sigma'^{(\ell)}_{k;\delta_1}\sigma'^{(\ell)}_{k;\delta_2}\sigma^{(\ell)}_{m;\delta_3}\widehat{H}^{(\ell)}_{mk;\delta_0\delta_1}\widehat{H}^{(\ell)}_{mk;\delta_0\delta_2}\right] \qquad (11.43)\\
&+\Big(C_W^{(\ell+1)}\Big)^2\frac{1}{n_\ell}\sum_{k,m=1}^{n_\ell}\mathbb{E}\left[\sigma'^{(\ell)}_{m;\delta_0}\sigma'^{(\ell)}_{k;\delta_1}\sigma'^{(\ell)}_{k;\delta_2}\sigma^{(\ell)}_{m;\delta_3}\widehat{\mathrm{d}H}^{(\ell)}_{mkk;\delta_0\delta_1\delta_2}\right]+O\Big(\frac{1}{n}\Big),\\
Q^{(\ell+1)}_{\delta_0\delta_1\delta_2\delta_3} \equiv&\Big(C_W^{(\ell+1)}\Big)^2\frac{1}{n_\ell}\sum_{k,m=1}^{n_\ell}\mathbb{E}\left[\sigma''^{(\ell)}_{k;\delta_0}\sigma'^{(\ell)}_{k;\delta_1}\sigma'^{(\ell)}_{m;\delta_2}\sigma^{(\ell)}_{m;\delta_3}\widehat{H}^{(\ell)}_{kk;\delta_0\delta_1}\widehat{H}^{(\ell)}_{km;\delta_0\delta_2}\right]+\frac{\lambda_W^{(\ell+1)}}{C_W^{(\ell+1)}}F^{(\ell+1)}_{\delta_1\delta_0\delta_3\delta_2} \qquad (11.44)\\
&+\Big(C_W^{(\ell+1)}\Big)^2\frac{1}{n_\ell}\sum_{k,m=1}^{n_\ell}\mathbb{E}\left[\sigma'^{(\ell)}_{k;\delta_0}\sigma'^{(\ell)}_{k;\delta_1}\sigma'^{(\ell)}_{m;\delta_2}\sigma^{(\ell)}_{m;\delta_3}\widehat{\mathrm{d}H}^{(\ell)}_{kkm;\delta_0\delta_1\delta_2}\right]+O\Big(\frac{1}{n}\Big).
\end{aligned}$$

To see how these general expressions reduce to the ones we had for $P^{(2)}_{\delta_0\delta_1\delta_2\delta_3}$ and $Q^{(2)}_{\delta_0\delta_1\delta_2\delta_3}$ in the second layer, (11.31) and (11.32), recall that the first-layer NTK is deterministic and diagonal in neural indices $\widehat{H}^{(1)}_{i_1 i_2;\delta_1\delta_2}=\delta_{i_1 i_2}H^{(1)}_{\delta_1\delta_2}$, that the first-layer dNTK vanishes $\widehat{\mathrm{d}H}^{(1)}_{i_0 i_1 i_2;\delta_0\delta_1\delta_2}=0$, and that in the second layer we had for the NTK-preactivation cross correlation tensor $F^{(2)}_{\delta_1\delta_0\delta_3\delta_2}=\Big(C_W^{(2)}\Big)^2 H^{(1)}_{\delta_0\delta_2}\left\langle\sigma_{\delta_1}\sigma_{\delta_3}\sigma'_{\delta_0}\sigma'_{\delta_2}\right\rangle_{G^{(1)}}$ (8.39).

Finally, note that by setting our observable to $\mathcal{O}=z^{(\ell+1)}_{i_3;\delta_3}$, we get

$$\mathbb{E}\left[\widehat{\mathrm{d}H}^{(\ell+1)}_{i_0 i_1 i_2;\delta_0\delta_1\delta_2}z^{(\ell+1)}_{i_3;\delta_3}\right]=\frac{1}{n_\ell}\left[\delta_{i_0 i_3}\delta_{i_1 i_2}P^{(\ell+1)}_{\delta_0\delta_1\delta_2\delta_3}+\delta_{i_0 i_1}\delta_{i_2 i_3}Q^{(\ell+1)}_{\delta_0\delta_1\delta_2\delta_3}+\delta_{i_0 i_2}\delta_{i_1 i_3}Q^{(\ell+1)}_{\delta_0\delta_2\delta_1\delta_3}\right]. \qquad (11.45)$$

Importantly, this means that at leading non-vanishing order in $1/n$, the tensors in the decomposition of our elementary dNTK-preactivation cross correlator with a single preactivation (11.45) completely fix the general dNTK-preactivation cross correlation with more complicated observables (11.42).[5] Thus, to completely incorporate the leading

[5] In other words, at our order in $1/n$ we can always replace the ℓ-th-layer dNTK inside any expectations

effects of the dNTK in our analysis we only need to evaluate recursions for $P^{(\ell)}$ and $Q^{(\ell)}$.[6]

P-recursion

To find a recursion for $P^{(\ell)}$, we need to evaluate the two expectations in (11.43).

For the first expectation, making a decomposition of the NTK into a mean and fluctuation as $\widehat{H}^{(\ell)}_{i_1 i_2;\delta_1\delta_2} = \delta_{i_1 i_2} H^{(\ell)}_{\delta_1\delta_2} + \widehat{\Delta H}^{(\ell)}_{i_1 i_2;\delta_1\delta_2}$, we get

$$\begin{aligned}
&\frac{1}{n_\ell^2}\sum_{k,m}\mathbb{E}\left[\sigma''^{(\ell)}_{m;\delta_0}\sigma'^{(\ell)}_{k;\delta_1}\sigma'^{(\ell)}_{k;\delta_2}\sigma^{(\ell)}_{m;\delta_3}\widehat{H}^{(\ell)}_{mk;\delta_0\delta_1}\widehat{H}^{(\ell)}_{mk;\delta_0\delta_2}\right] \qquad (11.47)\\
=&\frac{1}{n_\ell}\left\langle\sigma''_{\delta_0}\sigma'_{\delta_1}\sigma'_{\delta_2}\sigma_{\delta_3}\right\rangle_{G^{(\ell)}} H^{(\ell)}_{\delta_0\delta_1}H^{(\ell)}_{\delta_0\delta_2} + \frac{1}{n_{\ell-1}}\left\langle\sigma''_{\delta_0}\sigma_{\delta_3}\right\rangle_{G^{(\ell)}}\left\langle\sigma'_{\delta_1}\sigma'_{\delta_2}\right\rangle_{G^{(\ell)}} B^{(\ell)}_{\delta_0\delta_0\delta_1\delta_2} + O\left(\frac{1}{n^2}\right).
\end{aligned}$$

In particular, the cross terms consisting of an NTK mean and an NTK fluctuation dropped out because the Kronecker delta from the mean constrained the double sum and the fluctuation gave an additional $1/n$ suppression. If this explanation was a little too fast, you should review our slower derivation of the B-recursion from (8.83) to (8.89), which is identical in form to (11.47) above up to where the sample indices and the derivatives go.

For the second expectation in (11.43), we can just apply our general cross-correlation formula (11.42) for layer ℓ, letting us simplify it as

$$\begin{aligned}
&\frac{1}{n_\ell}\sum_{k,m=1}^{n_\ell}\mathbb{E}\left[\sigma'^{(\ell)}_{m;\delta_0}\sigma'^{(\ell)}_{k;\delta_1}\sigma'^{(\ell)}_{k;\delta_2}\sigma^{(\ell)}_{m;\delta_3}\widehat{\mathrm{d}H}^{(\ell)}_{mkk;\delta_0\delta_1\delta_2}\right] \qquad (11.48)\\
=&\frac{1}{n_\ell n_{\ell-1}}\sum_{k,m=1}^{n_\ell}\sum_{\delta_4\in\mathcal{D}}\left[\left\langle\!\!\left\langle\frac{\partial}{\partial z^{(\ell)}_{m;\delta_4}}\left(\sigma'^{(\ell)}_{m;\delta_0}\sigma'^{(\ell)}_{k;\delta_1}\sigma'^{(\ell)}_{k;\delta_2}\sigma^{(\ell)}_{m;\delta_3}\right)\right\rangle\!\!\right\rangle_{G^{(\ell)}} P^{(\ell)}_{\delta_0\delta_1\delta_2\delta_4} + \delta_{mk}\times O(1)\right] + O\left(\frac{1}{n}\right)\\
=&\left(\frac{n_\ell}{n_{\ell-1}}\right)\left[\left\langle\sigma''_{\delta_0}\sigma_{\delta_3}\right\rangle_{G^{(\ell)}}\left\langle\sigma'_{\delta_1}\sigma'_{\delta_2}\right\rangle_{G^{(\ell)}} P^{(\ell)}_{\delta_0\delta_1\delta_2\delta_0} + \left\langle\sigma'_{\delta_0}\sigma'_{\delta_3}\right\rangle_{K^{(\ell)}}\left\langle\sigma'_{\delta_1}\sigma'_{\delta_2}\right\rangle_{K^{(\ell)}} P^{(\ell)}_{\delta_0\delta_1\delta_2\delta_3}\right] + O\left(\frac{1}{n}\right).
\end{aligned}$$

Here in the second line, we've simply written the $Q^{(\ell)}$ terms proportional to δ_{mk} as $O(1)$; the details of these terms do not matter because when we perform the double sum with

---

by the following differential operator

$$\widehat{\mathrm{d}H}^{(\ell)}_{i_0 i_1 i_2;\delta_0\delta_1\delta_2} \to \frac{1}{n_{\ell-1}}\sum_{\delta\in\mathcal{D}}\left[\delta_{i_1 i_2}P^{(\ell)}_{\delta_0\delta_1\delta_2\delta}\frac{\partial}{\partial z^{(\ell)}_{i_0;\delta}} + \delta_{i_0 i_1}Q^{(\ell)}_{\delta_0\delta_1\delta_2\delta}\frac{\partial}{\partial z^{(\ell)}_{i_2;\delta}} + \delta_{i_0 i_2}Q^{(\ell)}_{\delta_0\delta_2\delta_1\delta}\frac{\partial}{\partial z^{(\ell)}_{i_1;\delta}}\right], \quad (11.46)$$

with non-fluctuating coefficients $P^{(\ell)}$ and $Q^{(\ell)}$. When we use this replacement, we must remember that the derivatives act on all of the ℓ-th-layer preactivations multiplying the dNTK; i.e. move the dNTK all the way to the left side of the expectation before making such a replacement.

[6] Any preactivation-NTK-dNTK cross correlators such as $\mathbb{E}\left[\widehat{\Delta H}^{(\ell)}_{i_1 i_2;\delta_1\delta_2}\widehat{\mathrm{d}H}^{(\ell)}_{i_3 i_4 i_5;\delta_3\delta_4\delta_5}z^{(\ell)}_{i_6;\delta_6}\right]$ and any higher-order dNTK correlators such as $\mathbb{E}\left[\widehat{\mathrm{d}H}^{(\ell)}_{i_0 i_1 i_2;\delta_0\delta_1\delta_2}\widehat{\mathrm{d}H}^{(\ell)}_{i_3 i_4 i_5;\delta_3\delta_4\delta_5}\right]$ are subleading. We won't show this explicitly, but you may find additional tranquility in working it out for yourself; both examples are relatively simple to work out for the second layer, $L = 2$.

the restriction $m = k$, they will be subleading. As for the term proportional to $P^{(\ell)}$, the diagonal contribution with $k = m$ is similarly subleading; the leading contribution comes from the $(n_\ell^2 - n_\ell)$ off-diagonal pieces with $k \neq m$, for which the derivative acts only on two activations out of the four, and then we can further use Gaussian factorization to write each term as a product of single-neuron Gaussian expectations.

Plugging these two simplified expectations back into our expression for $P^{(\ell+1)}$ (11.43), we get a recursion:

$$\begin{aligned} P^{(\ell+1)}_{\delta_0\delta_1\delta_2\delta_3} &= \left(C_W^{(\ell+1)}\right)^2 \left\langle \sigma''_{\delta_0}\sigma'_{\delta_1}\sigma'_{\delta_2}\sigma_{\delta_3}\right\rangle_{G^{(\ell)}} H^{(\ell)}_{\delta_0\delta_1} H^{(\ell)}_{\delta_0\delta_2} \\ &\quad + \left(\frac{n_\ell}{n_{\ell-1}}\right)\left(C_W^{(\ell+1)}\right)^2 \left\langle \sigma'_{\delta_1}\sigma'_{\delta_2}\right\rangle_{G^{(\ell)}} \\ &\quad\quad \times\left[\left\langle \sigma''_{\delta_0}\sigma_{\delta_3}\right\rangle_{G^{(\ell)}} P^{(\ell)}_{\delta_0\delta_1\delta_2\delta_0} + \left\langle \sigma'_{\delta_0}\sigma'_{\delta_3}\right\rangle_{G^{(\ell)}} P^{(\ell)}_{\delta_0\delta_1\delta_2\delta_3} + \left\langle \sigma''_{\delta_0}\sigma_{\delta_3}\right\rangle_{G^{(\ell)}} B^{(\ell)}_{\delta_0\delta_0\delta_1\delta_2}\right] + O\left(\frac{1}{n}\right) . \end{aligned} \tag{11.49}$$

Interestingly, we see that this dNTK tensor $P^{(\ell)}$ mixes with the NTK mean $H^{(\ell)}$ as well as the NTK-variance tensor $B^{(\ell)}$. Since $H^{(\ell)}$ and $B^{(\ell)}$ are of order one, and since $P^{(1)} = 0$, this recursion shows that $P^{(\ell)}$ will recursively stay of order one for all layers ℓ.

Q-recursion

To find a recursion for $Q^{(\ell)}$, we need to work out the two expectations in (11.44).

For the first expectation, again making a decomposition of the NTK into a mean and fluctuation as $\widehat{H}^{(\ell)}_{i_1 i_2;\delta_1\delta_2} = \delta_{i_1 i_2} H^{(\ell)}_{\delta_1\delta_2} + \widehat{\Delta H}^{(\ell)}_{i_1 i_2;\delta_1\delta_2}$, we get

$$\begin{aligned} &\frac{1}{n_\ell^2}\sum_{k,m}\mathbb{E}\left[\sigma''^{(\ell)}_{k;\delta_0}\sigma'^{(\ell)}_{k;\delta_1}\sigma'^{(\ell)}_{m;\delta_2}\sigma^{(\ell)}_{m;\delta_3}\widehat{H}^{(\ell)}_{kk;\delta_0\delta_1}\widehat{H}^{(\ell)}_{km;\delta_0\delta_2}\right] \\ =&\frac{1}{n_\ell^2}\sum_{k}\mathbb{E}\left[\sigma''^{(\ell)}_{k;\delta_0}\sigma'^{(\ell)}_{k;\delta_1}\sigma'^{(\ell)}_{k;\delta_2}\sigma^{(\ell)}_{k;\delta_3}\right] H^{(\ell)}_{\delta_0\delta_1} H^{(\ell)}_{\delta_0\delta_2} + \frac{1}{n_\ell^2}\sum_{k}\mathbb{E}\left[\sigma^{(\ell)}_{k;\delta_3}\sigma''^{(\ell)}_{k;\delta_0}\sigma'^{(\ell)}_{k;\delta_1}\sigma'^{(\ell)}_{k;\delta_2}\widehat{\Delta H}^{(\ell)}_{kk;\delta_0\delta_1}\right] H^{(\ell)}_{\delta_0\delta_2} \\ &+ \frac{1}{n_\ell^2}\sum_{k,m}\mathbb{E}\left[\sigma''^{(\ell)}_{k;\delta_0}\sigma'^{(\ell)}_{k;\delta_1}\sigma'^{(\ell)}_{m;\delta_2}\sigma^{(\ell)}_{m;\delta_3}\widehat{\Delta H}^{(\ell)}_{km;\delta_0\delta_2}\right] H^{(\ell)}_{\delta_0\delta_1} \\ &+ \frac{1}{n_\ell^2}\sum_{k,m}\mathbb{E}\left[\sigma''^{(\ell)}_{k;\delta_0}\sigma'^{(\ell)}_{k;\delta_1}\sigma'^{(\ell)}_{m;\delta_2}\sigma^{(\ell)}_{m;\delta_3}\widehat{\Delta H}^{(\ell)}_{kk;\delta_0\delta_1}\widehat{\Delta H}^{(\ell)}_{km;\delta_0\delta_2}\right] \\ =&\frac{1}{n_\ell}\left\langle \sigma''_{\delta_0}\sigma'_{\delta_1}\sigma'_{\delta_2}\sigma_{\delta_3}\right\rangle_{G^{(\ell)}} H^{(\ell)}_{\delta_0\delta_1} H^{(\ell)}_{\delta_0\delta_2} \\ &+ \frac{1}{n_{\ell-1}}\sum_{\delta_4,\delta_5,\delta_6,\delta_7} H^{(\ell)}_{\delta_0\delta_1}\left\langle z_{\delta_4}\sigma''_{\delta_0}\sigma'_{\delta_1}\right\rangle_{G^{(\ell)}}\left\langle z_{\delta_5}\sigma'_{\delta_2}\sigma_{\delta_3}\right\rangle_{G^{(\ell)}} G^{\delta_4\delta_6}_{(\ell)} G^{\delta_5\delta_7}_{(\ell)} F^{(\ell)}_{\delta_6\delta_0\delta_7\delta_2} + O\left(\frac{1}{n^2}\right) . \end{aligned} \tag{11.50}$$

Here, to go from the second expression to the third expression, we had to evaluate four expectations: the first expectation gives a single-neuron Gaussian expectation at leading order; the second expectation is subleading, cf. (8.71); the third expectation can also be evaluated with that same general NTK-preactivation cross-correlation formula (8.71),

but in this case gives a leading term proportional to $F^{(\ell)}$; and the final expectation vanishes due to the unpaired m neural index, cf. similar manipulations in (8.87) and then the decomposition (8.82).

To simplify the second expectation in (11.44), we can again apply our general cross-correlation formula (11.42) for layer ℓ:

$$\begin{aligned}
&\frac{1}{n_\ell}\sum_{k,m=1}^{n_\ell}\mathbb{E}\left[\sigma'^{(\ell)}_{k;\delta_0}\sigma'^{(\ell)}_{k;\delta_1}\sigma'^{(\ell)}_{m;\delta_2}\sigma^{(\ell)}_{m;\delta_3}\widehat{\mathrm{d}H}^{(\ell)}_{kkm;\delta_0\delta_1\delta_2}\right] \qquad (11.51)\\
=&\frac{1}{n_\ell n_{\ell-1}}\sum_{k,m=1}^{n_\ell}\sum_{\delta_4\in\mathcal{D}}\left[\left\langle\!\!\left\langle\frac{\partial}{\partial z^{(\ell)}_{m;\delta_4}}\left(\sigma'^{(\ell)}_{k;\delta_0}\sigma'^{(\ell)}_{k;\delta_1}\sigma'^{(\ell)}_{m;\delta_2}\sigma^{(\ell)}_{m;\delta_3}\right)\right\rangle\!\!\right\rangle_{G^{(\ell)}}Q^{(\ell)}_{\delta_0\delta_1\delta_2\delta_4}\right]+O\left(\frac{1}{n}\right)\\
=&\left(\frac{n_\ell}{n_{\ell-1}}\right)\left[\left\langle\sigma''_{\delta_2}\sigma_{\delta_3}\right\rangle_{K^{(\ell)}}\left\langle\sigma'_{\delta_0}\sigma'_{\delta_1}\right\rangle_{G^{(\ell)}}Q^{(\ell)}_{\delta_0\delta_1\delta_2\delta_2}+\left\langle\sigma'_{\delta_2}\sigma'_{\delta_3}\right\rangle_{G^{(\ell)}}\left\langle\sigma'_{\delta_1}\sigma'_{\delta_2}\right\rangle_{G^{(\ell)}}Q^{(\ell)}_{\delta_0\delta_1\delta_2\delta_3}\right]+O\left(\frac{1}{n}\right).
\end{aligned}$$

Here in the second line, this time we didn't even write the terms proportional to δ_{mk}; just as we saw before when working out the P-recursion, the restriction $k=m$ for the double sum will make such terms subleading. In the final equality – again similarly to the P-recursion – we kept the off-diagonal terms with $k\neq m$, took the derivative, used Gaussian factorization, and performed the double sum.

Plugging these two simplified expectations back into our expression for $Q^{(\ell+1)}$ (11.44), we get our final recursion of the book:

$$\begin{aligned}
Q^{(\ell+1)}_{\delta_0\delta_1\delta_2\delta_3}=&\left(C_W^{(\ell+1)}\right)^2\left\langle\sigma''_{\delta_0}\sigma'_{\delta_1}\sigma'_{\delta_2}\sigma_{\delta_3}\right\rangle_{G^{(\ell)}}H^{(\ell)}_{\delta_0\delta_1}H^{(\ell)}_{\delta_0\delta_2}+\frac{\lambda_W^{(\ell+1)}}{C_W^{(\ell+1)}}F^{(\ell+1)}_{\delta_1\delta_0\delta_3\delta_2}\\
&+\left(\frac{n_\ell}{n_{\ell-1}}\right)\left(C_W^{(\ell+1)}\right)^2\left\langle\sigma'_{\delta_0}\sigma'_{\delta_1}\right\rangle_{G^{(\ell)}}\left[\left\langle\sigma''_{\delta_2}\sigma_{\delta_3}\right\rangle_{G^{(\ell)}}Q^{(\ell)}_{\delta_0\delta_1\delta_2\delta_2}+\left\langle\sigma'_{\delta_2}\sigma'_{\delta_3}\right\rangle_{G^{(\ell)}}Q^{(\ell)}_{\delta_0\delta_1\delta_2\delta_3}\right]\\
&+\left(\frac{n_\ell}{n_{\ell-1}}\right)\left(C_W^{(\ell+1)}\right)^2H^{(\ell)}_{\delta_0\delta_1}\sum_{\delta_4,\ldots,\delta_7\in\mathcal{D}}\left\langle z_{\delta_4}\sigma''_{\delta_0}\sigma'_{\delta_1}\right\rangle_{G^{(\ell)}}\left\langle z_{\delta_5}\sigma'_{\delta_2}\sigma_{\delta_3}\right\rangle_{G^{(\ell)}}G^{\delta_4\delta_6}_{(\ell)}G^{\delta_5\delta_7}_{(\ell)}F^{(\ell)}_{\delta_6\delta_0\delta_7\delta_2}\\
&+O\left(\frac{1}{n}\right). \qquad (11.52)
\end{aligned}$$

Interestingly, in this case we see that this dNTK tensor $Q^{(\ell)}$ mixes with the NTK mean $H^{(\ell)}$ as well as the NTK-preactivation cross-correlation tensor $F^{(\ell)}$.[7] Again, since $H^{(\ell)}$ and $F^{(\ell)}$ are both of order one, and since $Q^{(1)}=0$, this recursion shows that $Q^{(\ell)}$ will also recursively stay of order one for all layers ℓ.

[7] Also of possible interest, while the recursions for $F^{(\ell)}$ (8.79) and $B^{(\ell)}$ (8.89) were sourced by the NTK mean $H^{(\ell)}$, but otherwise didn't mix with any finite-width tensors, the recursion for NTK-preactivation cross-correlation tensor $D^{(\ell)}$ (8.77) mixed with the four-point vertex $V^{(\ell)}$, and the recursion for NTK-variance tensor $A^{(\ell)}$ (8.97) mixed with both $V^{(\ell)}$ and $D^{(\ell)}$. Thus, at least at this order, it seems like the correlations and fluctuations of the type $F^{(\ell)}$ and $B^{(\ell)}$ are potentially useful for the representation learning that the dNTK induces, while $V^{(\ell)}$, $D^{(\ell)}$ and $A^{(\ell)}$ may be more associated with instantiation-to-instantiation fluctuations.

In conclusion, together with the general dNTK-preactivation cross-correlation formula (11.42), the P-Q recursions, (11.49) and (11.52), show that the leading dNTK-preactivation cross correlation is $1/n$-suppressed. In other words, the effects of the dNTK are visible *at finite width only*.

11.3 Effective Theory of the dNTK at Initialization

This section parallels our previous effective theory work on preactivation statistics and NTK-preactivation joint statistics (§5 ‖ §9). In particular, since we already know so many different reasons why criticality is essential, cf. §5, §6, §9, and §10, we'll spend less time on the disastrous consequences of not picking critical initialization hyperparameters and more time on finding asymptotic solutions to the P- and Q-recursions at criticality.

As we did in our discussion of preactivation criticality in §5 and NTK criticality in §9, throughout this section we'll set the bias variance $C_b^{(\ell)}$ and the rescaled weight variance $C_W^{(\ell)}$ to be uniform across layers

$$C_b^{(\ell)} = C_b\,, \qquad C_W^{(\ell)} = C_W\,. \tag{11.53}$$

Further mirroring §5.4 and §9.1–§9.3, we'll consider MLPs with uniform hidden layer widths

$$n_1 = \ldots = n_{L-1} \equiv n\,. \tag{11.54}$$

Finally, analogous to §9, we're only going to focus on single-input statistics, leaving the evaluation of the multi-input recursions as an adventure for thrill seekers.[8]

With these choices made, let's write down the leading single-input recursions for $P^{(\ell)}$ and $Q^{(\ell)}$. Dropping the sample indices and contributions that are subleading in $1/n$, in particular replacing the mean metric by the kernel $G^{(\ell)} \to K^{(\ell)}$ and the NTK mean by the frozen NTK $H^{(\ell)} \to \Theta^{(\ell)}$, the recursions (11.49) and (11.52) reduce to

$$\begin{aligned} P^{(\ell+1)} =& C_W^2 \left\langle \sigma''\sigma'\sigma'\sigma \right\rangle_{K^{(\ell)}} \left(\Theta^{(\ell)}\right)^2 + C_W \chi_\perp^{(\ell)} \left\langle \sigma''\sigma \right\rangle_{K^{(\ell)}} B^{(\ell)} \\ &+ \left[C_W \chi_\perp^{(\ell)} \left\langle \sigma''\sigma \right\rangle_{K^{(\ell)}} + \left(\chi_\perp^{(\ell)}\right)^2 \right] P^{(\ell)}\,, \end{aligned} \tag{11.55}$$

$$\begin{aligned} Q^{(\ell+1)} =& C_W^2 \left\langle \sigma''\sigma'\sigma'\sigma \right\rangle_{K^{(\ell)}} \left(\Theta^{(\ell)}\right)^2 + \frac{\lambda_W^{(\ell+1)}}{C_W} F^{(\ell+1)} + 2h^{(\ell)} \chi_\parallel^{(\ell)} \Theta^{(\ell)} F^{(\ell)} \\ &+ \left[C_W \chi_\perp^{(\ell)} \left\langle \sigma''\sigma \right\rangle_{K^{(\ell)}} + \left(\chi_\perp^{(\ell)}\right)^2 \right] Q^{(\ell)}\,, \end{aligned} \tag{11.56}$$

with the initial conditions (cf. §11.2.1)

$$P^{(1)} = Q^{(1)} = 0\,. \tag{11.57}$$

[8] You'll have to generalize the $\gamma^{[a]}$ into a tensor product $\gamma^{[a]} \otimes \gamma^{[b]}$ and then further decompose such a basis according to the symmetries of the finite-width tensors you'll want to expand.

To simplify these expressions, we have recalled our two susceptibilities, the parallel susceptibility (5.50) and the perpendicular susceptibility (5.51), which are given by

$$\chi_{\parallel}^{(\ell)} \equiv \frac{C_W}{K^{(\ell)}} \left\langle \sigma' \sigma z \right\rangle_{K^{(\ell)}} , \tag{11.58}$$

$$\chi_{\perp}^{(\ell)} \equiv C_W \left\langle \sigma' \sigma' \right\rangle_{K^{(\ell)}} , \tag{11.59}$$

and we have also recalled our least favorite helper function (5.52),

$$h^{(\ell)} \equiv \frac{1}{2} \frac{d}{dK^{(\ell)}} \chi_{\perp}^{(\ell)} = \frac{C_W}{2K^{(\ell)}} \left\langle \sigma'' \sigma' z \right\rangle_{K^{(\ell)}} , \tag{11.60}$$

though we've given a new expression for it on the right-hand side, obtained through integration by parts, cf. (5.53).

To solve these recursions at criticality, we need to remember our scaling ansatz for observables (5.93),

$$\mathcal{O}^{(\ell)} = \left(\frac{1}{\ell}\right)^{p_{\mathcal{O}}} \left[c_{0,0} + c_{1,1} \left(\frac{\log \ell}{\ell}\right) + c_{1,0} \left(\frac{1}{\ell}\right) + c_{2,2} \left(\frac{\log^2 \ell}{\ell^2}\right) + \ldots \right] . \tag{11.61}$$

Here, solving the dNTK recursions will yield new critical exponents p_P and p_Q that describe the asymptotic depth scaling of $P^{(\ell)}$ and $Q^{(\ell)}$, respectively.

Additionally, in order to understand the relative size of the dNTK-preactivation cross correlation, we will need to identify appropriate dimensionless quantities just as we did before in §5.4 and §9.1. In this case, it will turn out that we should normalize by two factors of the (frozen) NTK

$$\frac{P^{(\ell)}}{n \left(\Theta^{(\ell)}\right)^2} \sim \frac{1}{n} \left(\frac{1}{\ell}\right)^{p_P - 2p_\Theta} , \qquad \frac{Q^{(\ell)}}{n \left(\Theta^{(\ell)}\right)^2} \sim \frac{1}{n} \left(\frac{1}{\ell}\right)^{p_Q - 2p_\Theta} , \tag{11.62}$$

where on the right-hand side p_Θ is the critical exponent for the frozen NTK.

To see why these are the appropriate quantities to consider, recall our discussion of *dimensional analysis* in footnote 15 of §1.3 and that our notation of $[z]$ means "z is measured in units of $[z]$." Looking at our second-order update for preactivations in (11.9), and remembering that we can only add terms that have the same dimensions, we must have

$$[z] = [\eta] \, [\epsilon] \, [\widehat{H}] = [\eta]^2 \, [\epsilon]^2 \, [\widehat{\mathrm{d}H}] , \tag{11.63}$$

from which it's clear that $[\eta] \, [\epsilon] = [z] \, [\widehat{H}]^{-1}$, and subsequently we see that P and Q have dimensions of NTK squared:

$$[P] = [Q] \equiv [z] \, [\widehat{\mathrm{d}H}] = [\widehat{H}]^2 . \tag{11.64}$$

If this still seems a little counterintuitive, the utility of considering these particular ratios (11.62) will become even more apparent when we analyze the stochastic prediction of a fully-trained finite-width network in §∞.2.3.

After solving the P- and Q-recursions for both universality classes and looking at these dimensionless quantities (11.62), we'll again find *scaling laws* that transcend universality class:

$$p_P - 2p_\Theta = -1\,, \quad p_Q - 2p_\Theta = -1\,. \tag{11.65}$$

Specifically, we'll see that these laws hold for both the scale-invariant and $K^\star = 0$ universality classes.[9] Thus, we'll be able to conclude that all the leading finite-width effects of the preactivation-NTK-dNTK joint distribution are *relevant* – in the sense of RG flow – controlled by the same ℓ/n perturbative cutoff.

Now that we're fully prepared for what we're going to see, let's actually solve the dNTK recursions for our two important universality classes.

11.3.1 Scale-Invariant Universality Class

First, let's recall some previous results that we'll need in order to evaluate the recursions (11.55) and (11.56). For the scale-invariant universality class, we know from (5.62), (9.33), and (9.37) that

$$\chi_\perp^{(\ell)} = C_W A_2 \equiv \chi\,, \qquad h^{(\ell)} = 0\,, \qquad \left\langle \sigma'\sigma'\sigma\sigma \right\rangle_{K^{(\ell)}} = A_4 K^{(\ell)}\,, \tag{11.66}$$

where as a reminder $A_2 \equiv (a_+^2 + a_-^2)/2$ and $A_4 \equiv (a_+^4 + a_-^4)/2$, and the $a_\pm$ are the respective slopes of the positive/negative linear pieces of the activation function, cf. (5.59).

Next, we see that all the new Gaussian expectations in the recursions (11.55) and (11.56) involve the second derivative of the activation. For these, we need to be somewhat careful with nonlinear scale-invariant activation functions, since they have an undifferentiable kink at the origin. (For linear activation functions, $\sigma''(z) = 0$, and there's no subtlety.) For the first of these new expectations, note that we can integrate it by parts as

$$\left\langle \sigma''\sigma \right\rangle_K = \frac{1}{\sqrt{2\pi K}} \int_{-\infty}^{\infty} dz e^{-\frac{z^2}{2K}} \left(\frac{d}{dz}\sigma'\right)\sigma = \frac{1}{K}\left\langle z\sigma'\sigma \right\rangle_K - \left\langle \sigma'\sigma' \right\rangle_K = 0\,, \tag{11.67}$$

where in the final equality we used

$$\frac{1}{K}\left\langle z\sigma'\sigma \right\rangle_K = \left\langle \sigma'\sigma' \right\rangle_K = A_2\,. \tag{11.68}$$

Similarly, integrating the other new Gaussian expectation by parts, we find

$$\left\langle \sigma''\sigma'\sigma'\sigma \right\rangle_K = \frac{1}{K}\left\langle z\sigma'\sigma'\sigma'\sigma \right\rangle_K - 2\left\langle \sigma''\sigma'\sigma'\sigma \right\rangle_K - \left\langle \sigma'\sigma'\sigma'\sigma' \right\rangle_K\,. \tag{11.69}$$

Rearranging this, we easily see that

$$\left\langle \sigma''\sigma'\sigma'\sigma \right\rangle_K = \frac{1}{3K}\left\langle z\sigma'\sigma'\sigma'\sigma \right\rangle_K - \frac{1}{3}\left\langle \sigma'\sigma'\sigma'\sigma' \right\rangle_K = 0\,, \tag{11.70}$$

[9] To be more specific, for the scale-invariant class, we'll find that $P^{(\ell)}$ identically vanishes and that $Q^{(\ell)}$ solely determines the leading finite-width dNTK-preactivation cross correlation.

where in the final equality we used

$$\frac{1}{K}\left\langle z\sigma'\sigma'\sigma'\sigma\right\rangle_K = \left\langle \sigma'\sigma'\sigma'\sigma'\right\rangle_K = A_4\,. \tag{11.71}$$

Thus, we can safely ignore both of these new expectations.

Substituting in all of (11.66) and ignoring ignorable expectations, our single-input recursions (11.55) and (11.56) are extremely simple:

$$P^{(\ell+1)} = \chi^2 P^{(\ell)}\,, \tag{11.72}$$

$$Q^{(\ell+1)} = \chi^2 Q^{(\ell)} + \frac{\lambda_W^{(\ell+1)}}{C_W} F^{(\ell+1)}\,. \tag{11.73}$$

In particular, since our initial condition (11.57) is $P^{(1)} = 0$, we see immediately that $P^{(\ell)}$ vanishes identically for all layers:

$$P^{(\ell)} = 0\,. \tag{11.74}$$

In contrast, for $Q^{(\ell)}$ we see that the susceptibility χ is going to generically lead to exponential behavior.

Now, let's tune to scale-invariant criticality by setting the initialization hyperparameters as $C_b = 0$ and $C_W = 1/A_2$. As a consequence, this fixes the susceptibility to unity, $\chi = 1$, and leaves the kernel fixed for all layers as $K^{(\ell)} = K^\star$. Additionally, let's pick our training hyperparameters according to the learning rate equivalence principle, for which we're instructed to choose layer-independent learning rates (9.94) as

$$\lambda_b^{(\ell)} = \frac{\widetilde{\lambda}_b}{L}\,, \qquad \lambda_W^{(\ell)} = \frac{\widetilde{\lambda}_W}{L}\,. \tag{11.75}$$

Finally, with these hyperparameter choices, the single-input solution for the frozen NTK (9.44) and the single-input solution for the NTK-preactivation cross correlation $F^{(\ell)}$ (9.50) are given by

$$\Theta^{(\ell)} = \left(\widetilde{\lambda}_b + \widetilde{\lambda}_W A_2 K^\star\right)\frac{\ell}{L}\,, \tag{11.76}$$

$$F^{(\ell)} = \frac{\ell(\ell-1)}{2L}\left[\frac{A_4}{A_2^2}\left(\widetilde{\lambda}_b + \widetilde{\lambda}_W A_2 K^\star\right)K^\star\right]. \tag{11.77}$$

Plugging in the critical initialization hyperparameters, the fixed kernel, the learning rates (11.75), and the expression for $F^{(\ell)}$ (11.77), the Q-recursion (11.73) becomes

$$Q^{(\ell+1)} = Q^{(\ell)} + \frac{\ell(\ell+1)}{2L^2}\left[\frac{A_4}{A_2}\left(\widetilde{\lambda}_b + \widetilde{\lambda}_W A_2 K^\star\right)\widetilde{\lambda}_W K^\star\right]. \tag{11.78}$$

This simple recursion is exactly solved by

$$Q^{(\ell)} = \frac{\ell(\ell^2-1)}{6L^2}\left[\frac{A_4}{A_2}\left(\widetilde{\lambda}_b + \widetilde{\lambda}_W A_2 K^\star\right)\widetilde{\lambda}_W K^\star\right], \tag{11.79}$$

which satisfies the initial condition $Q^{(1)} = 0$.

With this solution, we can identify the critical exponent associated with the large-ℓ behavior of $Q^{(\ell)}$: $p_Q = -3$. Further, we see that our dimensionless quantity (11.62) will satisfy the scaling law (11.65) as promised,

$$p_Q - 2p_\Theta = -1\,, \tag{11.80}$$

where we have also used $p_\Theta = -1$ from the scale-invariant frozen NTK solution reprinted above in (11.76). More specifically, substituting in the solution for $\Theta^{(\ell)}$ (11.76) and the solution for $Q^{(\ell)}$ (11.79) into the dimensionless ratio (11.62), we find

$$\frac{Q^{(\ell)}}{n\left(\Theta^{(\ell)}\right)^2} = \frac{A_4}{6A_2}\left[\frac{\widetilde{\lambda}_W K^\star}{\widetilde{\lambda}_b + \widetilde{\lambda}_W A_2 K^\star}\right]\frac{\ell}{n} + \ldots\,. \tag{11.81}$$

Thus, we have verified the ℓ/n scaling of the leading dNTK-preactivation cross correlation for scale-invariant activation functions.

11.3.2 $K^\star = 0$ Universality Class

For the $K^\star = 0$ universality class, we'll begin again by recalling some previous results. First, we know from (5.84), (5.85), (9.64), and (11.60) the following:

$$\chi_\parallel(K) = \left(C_W\sigma_1^2\right)\left[1 + 2a_1 K + O\left(K^2\right)\right]\,, \tag{11.82}$$

$$\chi_\perp(K) = \left(C_W\sigma_1^2\right)\left[1 + b_1 K + O\left(K^2\right)\right]\,, \tag{11.83}$$

$$C_W^2\left\langle\sigma'\sigma'\sigma\sigma\right\rangle_K = \left(C_W\sigma_1^2\right)^2\left[K + O\left(K^2\right)\right]\,, \tag{11.84}$$

$$h(K) = \frac{1}{2}\frac{d}{dK}\chi_\perp(K) = \left(C_W\sigma_1^2\right)\left[\frac{b_1}{2} + O\left(K^1\right)\right]\,. \tag{11.85}$$

To interpret these results, remember that we Taylor expanded the activation function as

$$\sigma(z) = \sum_{p=0}^{\infty}\frac{\sigma_p}{p!}z^p\,, \tag{11.86}$$

defined the following combination of Taylor coefficients for convenience

$$a_1 \equiv \left(\frac{\sigma_3}{\sigma_1}\right) + \frac{3}{4}\left(\frac{\sigma_2}{\sigma_1}\right)^2\,, \tag{11.87}$$

$$b_1 \equiv \left(\frac{\sigma_3}{\sigma_1}\right) + \left(\frac{\sigma_2}{\sigma_1}\right)^2\,, \tag{11.88}$$

and required that all activation functions in this class satisfy $\sigma_0 = 0$ and $\sigma_1 \neq 0$. Then, making analogous Taylor expansions and performing Gaussian integrations order

by order, we can evaluate the new Gaussian expectations in the recursions (11.55) and (11.56) as

$$C_W \left\langle \sigma'' \sigma \right\rangle_K = \left(C_W \sigma_1^2 \right) \left[(2a_1 - b_1) K + O\left(K^2\right) \right] , \tag{11.89}$$

$$C_W^2 \left\langle \sigma'' \sigma' \sigma' \sigma \right\rangle_K = \left(C_W \sigma_1^2 \right)^2 \left[(-6a_1 + 7b_1) K + O\left(K^2\right) \right] . \tag{11.90}$$

Now, let's jump right into $K^\star = 0$ criticality (5.90) by tuning the initialization hyperparameters as $C_b = 0$ and $C_W = 1/\sigma_1^2$. At the same time, let's also tune our training hyperparameters according to the learning rate equivalence principle, which for $K^\star = 0$ activation functions is given by (9.95),

$$\lambda_b^{(\ell)} = \widetilde{\lambda}_b \left(\frac{1}{\ell} \right)^{p_\perp} L^{p_\perp - 1} , \qquad \lambda_W^{(\ell)} = \widetilde{\lambda}_W \left(\frac{L}{\ell} \right)^{p_\perp - 1} , \tag{11.91}$$

where the critical exponent for perpendicular perturbations is defined as $p_\perp \equiv b_1/a_1$. With these hyperparameter settings, let's also record the other single-input solutions that we need for the recursions, the kernel $K^{(\ell)}$ (5.92), the frozen NTK $\Theta^{(\ell)}$ (9.71), the NTK-preactivation cross correlation $F^{(\ell)}$ (9.81), and the NTK variance $B^{(\ell)}$ (9.82):

$$K^{(\ell)} = \left[\frac{1}{(-a_1)} \right] \frac{1}{\ell} + \ldots , \tag{11.92}$$

$$\Theta^{(\ell)} = \left[\widetilde{\lambda}_b + \frac{\widetilde{\lambda}_W \sigma_1^2}{(-a_1)} \right] \left(\frac{L}{\ell} \right)^{p_\perp - 1} + \ldots , \tag{11.93}$$

$$F^{(\ell)} = \frac{1}{(5 - p_\perp)} \left[\frac{1}{(-a_1)} \right] \left[\widetilde{\lambda}_b + \frac{\widetilde{\lambda}_W \sigma_1^2}{(-a_1)} \right] \left(\frac{L}{\ell} \right)^{p_\perp - 1} + \ldots , \tag{11.94}$$

$$B^{(\ell)} = \frac{L^{2p_\perp - 2}}{3} \left[\widetilde{\lambda}_b + \frac{\widetilde{\lambda}_W \sigma_1^2}{(-a_1)} \right]^2 \left(\frac{1}{\ell} \right)^{2p_\perp - 3} + \ldots . \tag{11.95}$$

Finally, plugging all our collected results and tunings (11.82)–(11.85) and (11.89)–(11.95) into the P-recursion (11.55) and the Q-recursion (11.56), we get

$$P^{(\ell+1)} = \left[1 - \frac{(p_\perp + 2)}{\ell} + \ldots \right] P^{(\ell)} + \frac{(p_\perp - 2)}{3} \left[\widetilde{\lambda}_b + \frac{\widetilde{\lambda}_W \sigma_1^2}{(-a_1)} \right]^2 \left(\frac{L}{\ell} \right)^{2p_\perp - 2} + \ldots , \tag{11.96}$$

$$\begin{aligned} Q^{(\ell+1)} = & \left[1 - \frac{(p_\perp + 2)}{\ell} + \ldots \right] Q^{(\ell)} \\ & + \frac{1}{(5 - p_\perp)} \left[(1 - p_\perp) \frac{\widetilde{\lambda}_W \sigma_1^2}{(-a_1)} - p_\perp \widetilde{\lambda}_b \right] \left[\widetilde{\lambda}_b + \frac{\widetilde{\lambda}_W \sigma_1^2}{(-a_1)} \right] \left(\frac{L}{\ell} \right)^{2p_\perp - 2} + \ldots . \end{aligned} \tag{11.97}$$

Let's tackle the P-recursion first. Plugging our scaling ansatz (11.61) into the recursion (11.96) and matching the terms, we find an asymptotic solution

$$P^{(\ell)} = - \frac{L^{2p_\perp - 2}}{3} \left(\frac{2 - p_\perp}{5 - p_\perp} \right) \left[\widetilde{\lambda}_b + \frac{\widetilde{\lambda}_W \sigma_1^2}{(-a_1)} \right]^2 \left(\frac{1}{\ell} \right)^{2p_\perp - 3} + \ldots . \tag{11.98}$$

Thus, the critical exponent for this dNTK-preactivation cross correlation is $p_P = 2p_\perp - 3$, and we obtain our promised scaling law (11.65)

$$p_P - 2p_\Theta = -1\,, \tag{11.99}$$

after substituting in $p_\Theta = p_\perp - 1$ from the $K^\star = 0$ frozen NTK solution (11.93). More specifically, substituting in the solution for $\Theta^{(\ell)}$ (11.93) and the solution for $P^{(\ell)}$ (11.98) into the dimensionless ratio (11.62), we get

$$\frac{P^{(\ell)}}{n\left(\Theta^{(\ell)}\right)^2} = -\frac{1}{3}\left(\frac{2-p_\perp}{5-p_\perp}\right)\frac{\ell}{n} + \ldots\,, \tag{11.100}$$

which *(i)* scales as ℓ/n, *(ii)* is independent of the training hyperparameters, and *(iii)* is manifestly negative, given $p_\perp \leq 1$.[10]

Similarly for the Q-recursion (11.97), plugging our scaling ansatz (11.61) into the recursion (11.97) and matching the terms one final time, we find

$$Q^{(\ell)} = \frac{L^{2p_\perp - 2}}{(5-p_\perp)^2}\left[1 - p_\perp - \frac{\widetilde{\lambda}_b}{\widetilde{\lambda}_b + \widetilde{\lambda}_W\sigma_1^2/(-a_1)}\right]\left[\widetilde{\lambda}_b + \frac{\widetilde{\lambda}_W\sigma_1^2}{(-a_1)}\right]^2\left(\frac{1}{\ell}\right)^{2p_\perp - 3}\,. \tag{11.101}$$

This gives a critical exponent of $p_Q = 2p_\perp - 3$ and a dimensionless ratio of

$$\frac{Q^{(\ell)}}{n\left(\Theta^{(\ell)}\right)^2} = \frac{1}{(5-p_\perp)^2}\left[1 - p_\perp - \frac{\widetilde{\lambda}_b}{\widetilde{\lambda}_b + \widetilde{\lambda}_W\sigma_1^2/(-a_1)}\right]\frac{\ell}{n} + \ldots\,. \tag{11.102}$$

Thus, we've now fully verified our scaling law (11.65):

$$p_Q - 2p_\Theta = -1\,. \tag{11.103}$$

In conclusion, all the nonzero dimensionless dNTK-preactivation cross correlations will grow as ℓ/n at leading order in the finite-width expansion. As the dNTK leads to a dynamical NTK, this is sufficient to see that deep finite-width networks are representation learners.

In the next section, we're going to set aside our direct investigation of deep MLPs and instead focus on a pedagogical model of representation learning that directly incorporates the effect of a nonzero dNTK. In the following chapter, we'll return to our finite-width networks and complete our goal of computing the effective distribution over wide and deep finite-width MLPs after training. The simplified model presented in the next section will make it easier to understand the results of our later analysis of such fully-trained networks.

[10] To recall why $p_\perp \leq 1$, without flipping back to footnote 33 of §10, *(i)* remember that we must have $a_1 < 0$ in order for the kernel $K^{(\ell)}$ to stay positive when asymptotically approaching the $K^\star = 0$ fixed point, cf. (11.92), *(ii)* note to yourself that $b_1 \geq a_1$, cf. (11.87) and (11.88), and *(iii)* realize that $a_1 < 0$ and $b_1 \geq a_1$ together imply that $p_\perp \equiv b_1/a_1 \leq 1$.

11.4 Nonlinear Models and Nearly-Kernel Methods

In §10.4, we placed infinite-width networks into a broader context of machine learning models. There, we discussed how the infinite-width network can be understood as either a *linear model* with fixed random features or, dually, as a *kernel method* with the kernel given by the frozen NTK. Now we are ready to find a broader context for finite-width networks.

In this chapter, we've seen that the statistics needed to compute the dynamics of finite-width networks (§11.2 and §11.3) are far more complicated than for infinite-width networks, nontrivially incorporating the second derivative of the network function. This additional complexity is irreducibly encapsulated by a single object: the dNTK (§11.1). The dNTK enables the NTK to evolve during training, leading to nontrivial *representation learning* from the training data.

The goal of this section is to abstract this property away from the deep learning framework and distill it into a **minimal model of representation learning** that captures all of its important features. Such a model provides a framework for studying the type of representation learning exhibited by deep neural networks, but more succinctly and more broadly. In other words, we hope that this endeavor will extend the standard toolbox of machine learning.

First in §11.4.1, we'll extend linear models discussed in §10.4.1 to *nonlinear models*. Then in §11.4.2, we'll give a dual description of these nonlinear models, *nearly-kernel methods*, which will extend the standard kernel methods that we discussed in §10.4.2. Finally in §11.4.3, as we did analogously for infinite-width networks before in §10.4.3, we'll see how finite-width networks can be understood in terms of this new framework.

11.4.1 Nonlinear Models

As a reminder from §10.4.1, a *linear model* is given by (10.119)

$$z_i(x_\delta;\theta) = \sum_{j=0}^{n_f} W_{ij}\phi_j(x_\delta)\,, \tag{11.104}$$

where the model parameters are given by the weight matrix $\theta = W_{ij}$, the model's features are given by the *feature function* $\phi_j(x)$, and we again adopt the old-school convention for incorporating biases, including a constant feature $\phi_0(x) \equiv 1$ so that the bias vector is given by $W_{i0} \equiv b_i$. Note that since we're not discussing neural networks in particular, there's no neural indices or layer indices here. Instead, in this equation, δ is a *sample index*, running over the $N_\mathcal{D}$ samples in the dataset $\delta \in \mathcal{D}$; i is a *vectorial index*, running over the n_{out} vectorial component of the model output z_i; and j is a *feature index*, running over $(n_f + 1)$ different features. In this setup, a feature function $\phi_j(x)$ is computed on an input sample x, and the weight matrix W_{ij} determines the effect of the j-th feature on the i-th component of the output. Traditionally, feature functions $\phi_j(x)$ are often *designed* such that the linear model works well for the desired task after optimization or *linear regression*.

To go beyond this linear paradigm, let's slightly *deform* it to get a ***nonlinear* model**:

$$z_i(x_\delta;\theta) = \sum_{j=0}^{n_f} W_{ij}\phi_j(x_\delta) + \frac{\epsilon}{2}\sum_{j_1,j_2=0}^{n_f} W_{ij_1}W_{ij_2}\psi_{j_1j_2}(x_\delta) + \dots \,. \tag{11.105}$$

Here, $\epsilon \ll 1$ is small parameter that controls the size of the deformation, and more importantly, we've introduced another set of feature functions, $\psi_{j_1j_2}(x)$, with *two* feature indices, making the model nonlinear in its weights. By definition, $\psi_{j_1j_2}(x)$ is symmetric under the exchange of the feature indices $j_1 \leftrightarrow j_2$, and the factor of $1/2$ is there because of the double counting of the double sum. For reasons that will be made clear shortly, we will call each component of $\psi_{j_1j_2}(x)$ a **meta feature function**, and so there are $(n_f+1)(n_f+2)/2$ different meta feature functions.

To familiarize ourselves with the features of this model, let's see how the model outputs change given a small change in the model parameters $W_{ij} \to W_{ij} + đW_{ij}$:

$$\begin{aligned}
&z_i(x_\delta;\theta+đ\theta)\\
=&z_i(x_\delta;\theta) + \sum_{j=0}^{n_f} đW_{ij}\left[\phi_j(x_\delta) + \epsilon\sum_{j_1=0}^{n_f} W_{ij_1}\psi_{j_1j}(x_\delta)\right] + \frac{\epsilon}{2}\sum_{j_1,j_2=0}^{n_f} đW_{ij_1}đW_{ij_2}\psi_{j_1j_2}(x_\delta) + \dots\\
=&z_i(x_\delta;\theta) + \sum_{j=0}^{n_f} đW_{ij}\,\phi^{\mathrm{E}}_{ij}(x_\delta;\theta) + \frac{\epsilon}{2}\sum_{j_1,j_2=0}^{n_f} đW_{ij_1}đW_{ij_2}\psi_{j_1j_2}(x_\delta) + \dots\,,
\end{aligned} \tag{11.106}$$

where in the last line we summarized the quantity in the square bracket in terms of an **effective feature function**:

$$\phi^{\mathrm{E}}_{ij}(x_\delta;\theta) \equiv \frac{dz_i(x_\delta;\theta)}{dW_{ij}} = \phi_j(x_\delta) + \epsilon\sum_{k=0}^{n_f} W_{ik}\psi_{kj}(x_\delta)\,. \tag{11.107}$$

The utility of this is as follows: the linear response of the model to changing its parameters is as if it *effectively* has a feature function, $\phi^{\mathrm{E}}_{ij}(x_\delta;\theta)$, which itself depends on the value of the model parameters. Moreover, the change in the effective feature function given a small change in the model parameters $W_{ik} \to W_{ik} + đW_{ik}$ is given by

$$\phi^{\mathrm{E}}_{ij}(x_\delta;\theta+đ\theta) = \phi^{\mathrm{E}}_{ij}(x_\delta;\theta) + \epsilon\sum_{k=0}^{n_f} đW_{ik}\,\psi_{kj}(x_\delta)\,. \tag{11.108}$$

Thus, the effective features $\phi^{\mathrm{E}}_{ij}(x_\delta;\theta)$ are *learnable*, evolving as a linear model, and for these effective features, the meta features $\psi_{kj}(x_\delta)$ play the role usually played by the features.[11] In this way, we can think of our nonlinear model as having a hierarchical structure, where the features evolve as if they are described by a linear model according

[11] This role of the meta features $\psi_{kj}(x_\delta)$ in the linear model for the effective features explains our choice of name. Note also that in this setup, we assume that both the feature function $\phi_j(x)$ and the meta feature function $\psi_{j_1j_2}(x)$ are picked by the model designer, just as the feature function was before for the linear model.

to (11.108), while the model's output evolves in a more complicated nonlinear way according to (11.106).

Note that we could extend this hierarchical structure further, e.g. by further deforming our nonlinear model (11.105) with a term that's cubic in the parameters and includes a three-indexed *meta-meta feature function* that analogously allows the meta feature functions to learn.[12] However, as the quadratic term already suffices to provide a minimal model of representation learning – just as a nonzero dNTK sufficed to induce nontrivial representation learning in the MLP – from here on, we'll explicitly truncate the model (11.105) at the quadratic order, giving us a **quadratic model**.[13]

To understand how representation learning works in this model, we should find the optimal values for the weights $W^\star_{ij}$ given a training set $\mathcal{A}$. Mirroring what we did before for linear models, let's minimize the MSE loss, which for our quadratic model is given by

$$\begin{aligned}\mathcal{L}_{\mathcal{A}}(\theta) &= \frac{1}{2}\sum_{\tilde{\alpha}\in\mathcal{A}}\sum_{i=1}^{n_{\text{out}}}\Big[y_{i;\tilde{\alpha}} - z_i(x_{\tilde{\alpha}};\theta)\Big]^2, \qquad (11.109)\\ &= \frac{1}{2}\sum_{\tilde{\alpha}\in\mathcal{A}}\sum_{i=1}^{n_{\text{out}}}\left[y_{i;\tilde{\alpha}} - \sum_{j=0}^{n_f} W_{ij}\,\phi_j(x_{\tilde{\alpha}}) - \frac{\epsilon}{2}\sum_{j_1,j_2=0}^{n_f} W_{ij_1}W_{ij_2}\psi_{j_1j_2}(x_{\tilde{\alpha}})\right]^2 .\end{aligned}$$

Supervised learning with such a quadratic model (11.105) doesn't have a particular name, but if it did, we'd all probably agree that its name should be **quadratic regression**. However, unlike linear regression – where the MSE loss (10.120) was quadratic in the model parameters – quadratic regression – where the loss is now *quartic* in the parameters – does not in general yield analytic solutions. Nevertheless, we can perturbatively find a solution to the quadratic regression problem by expanding in our small parameter ϵ, for which the regression is *nearly linear*.

Nearly-Linear Quadratic Regression

Taking the derivative of the loss $\mathcal{L}_{\mathcal{A}}$ (11.109) with respect to the model parameter W_{ij_0} and setting it to zero, we find

$$0 = \sum_{\tilde{\alpha}} \phi^{\text{E}}_{ij_0}(x_{\tilde{\alpha}};\theta^\star)\left[z_i(x_{\tilde{\alpha}};\theta^\star) - y_{i;\tilde{\alpha}}\right], \qquad (11.110)$$

making clear that the effective features (11.107) give the linear response of the model. Next, substituting in for the quadratic model (11.104) and the effective feature function

[12] Such deformations are essentially equivalent to incorporating further terms in the Taylor expansion of an arbitrary general model function $z_i(x_\delta;\theta)$, where the linear model is the base model, and the nonlinear model (11.105) gives the first improvement from the expansion.

[13] To leading order in the $1/n$ expansion, finite-width networks cannot self-consistently be described by a truncated quadratic model; instead, they fall into a class of *cubic models*, with evolving meta feature functions. This is one of the ways in which such finite-width networks are *non-minimal* representation learners and is the main reason for discussing quadratic models first before moving on to the more complicated analysis of finite-width networks with their dynamical dNTKs.

(11.107) and rearranging we find

$$\sum_{\tilde{\alpha}\in\mathcal{A}}\left\{\sum_{j_1=0}^{n_f} W^\star_{ij_1}\phi_{j_1}(x_{\tilde{\alpha}})\phi_{j_0}(x_{\tilde{\alpha}}) + \epsilon\sum_{j_1,j_2=0}^{n_f} W^\star_{ij_1}W^\star_{ij_2}\left[\phi_{j_1}(x_{\tilde{\alpha}})\psi_{j_2j_0}(x_{\tilde{\alpha}}) + \frac{1}{2}\psi_{j_1j_2}(x_{\tilde{\alpha}})\phi_{j_0}(x_{\tilde{\alpha}})\right]\right\}$$
$$= \sum_{\tilde{\alpha}\in\mathcal{A}} y_{i;\tilde{\alpha}}\left[\phi_{j_0}(x_{\tilde{\alpha}}) + \epsilon\sum_{j_1=0}^{n_f} W^\star_{ij_1}\psi_{j_1j_0}(x_{\tilde{\alpha}})\right] + O\left(\epsilon^2\right) . \tag{11.111}$$

This expression contains the two terms we found for linear regression in (10.121) as well as three additional terms proportional to the meta features $\psi_{j_1j_2}(x)$. Additionally, we truncated the $O(\epsilon^2)$ term since ϵ is assumed to be parametrically small. Importantly, we see clearly that the equation is overall nonlinear, with the two terms quadratic in the weights. In the language of physics, the linear equation of linear regression is *free* and exactly solvable, while the nonlinear equation of quadratic regression is *interacting*. Since ϵ multiplies all the new terms associated with quadratic regression, the nonlinear terms are all small, and so (11.111) exhibits *weakly-interacting* dynamics. This means that we can systematically solve this nonlinear equation via perturbation theory.

With that in mind, let's decompose the optimal weight matrix into a free linear part and an interacting nonlinear part as

$$W^\star_{ij} \equiv W^{\mathrm{F}}_{ij} + W^{\mathrm{I}}_{ij} , \tag{11.112}$$

with the idea being that the free part W^{F}_{ij} will solve the free linear regression equation (10.121), while the interacting part W^{I}_{ij} will solve the remaining linearized equation after substituting back in the solution for W^{F}_{ij}. Given that the quadratic regression problem (11.111) becomes a linear regression problem (10.121) in the limit of $\epsilon \to 0$, we naturally expect that the interacting part of the optimal weights should be proportional to the small parameter $W^{\mathrm{I}}_{ij} = O(\epsilon)$.

Let's quickly review our **direct optimization** solution to linear regression from §10.4.1 in the current context: defining an $(n_f{+}1)$-by-$(n_f{+}1)$ symmetric matrix (10.122)

$$M_{j_1j_2} \equiv \sum_{\tilde{\alpha}\in\mathcal{A}} \phi_{j_1}(x_{\tilde{\alpha}})\,\phi_{j_2}(x_{\tilde{\alpha}}) , \tag{11.113}$$

the linear part of the quadratic regression problem (11.111) can be written as

$$\sum_{j_1=0}^{n_f} W^{\mathrm{F}}_{ij_1} M_{j_1j_0} = \sum_{\tilde{\alpha}\in\mathcal{A}} y_{i;\tilde{\alpha}}\phi_{j_0}(x_{\tilde{\alpha}}) , \tag{11.114}$$

which can be solved by the multiplication of the inverse $(M^{-1})_{j_0j}$,

$$W^{\mathrm{F}}_{ij} = \sum_{j_0=0}^{n_f}\sum_{\tilde{\alpha}\in\mathcal{A}} y_{i;\tilde{\alpha}}\phi_{j_0}(x_{\tilde{\alpha}})\left(M^{-1}\right)_{j_0j} . \tag{11.115}$$

Recall that the inverse $(M^{-1})_{j_0 j}$ will not uniquely exist if we're in the *overparameterized* regime with more features than training examples, $(n_f + 1) > N_{\mathcal{A}}$, but we can use our regularization trick (10.124) to pick out a particular inverse. Going forward, we will assume that we're in this overparameterized regime and that the inverse was picked in this way.

Next, plugging in our decomposition (11.112) into our equation (11.111) and collecting the terms of order ϵ, remembering also that $W^{\mathrm{I}}_{ij} = O(\epsilon)$, we find for our linearized interacting dynamics,

$$\begin{aligned}\sum_{j_1=0}^{n_f} W^{\mathrm{I}}_{ij_1} M_{j_1 j_0} =& \epsilon \sum_{j_1=0}^{n_f} W^{\mathrm{F}}_{ij_1} \sum_{\tilde{\alpha}\in\mathcal{A}} y_{i;\tilde{\alpha}} \psi_{j_1 j_0}(x_{\tilde{\alpha}}) - \epsilon \sum_{j_1,j_2=0}^{n_f} W^{\mathrm{F}}_{ij_1} W^{\mathrm{F}}_{ij_2} \sum_{\tilde{\alpha}\in\mathcal{A}} \phi_{j_1}(x_{\tilde{\alpha}}) \psi_{j_2 j_0}(x_{\tilde{\alpha}}) \\ & - \frac{\epsilon}{2} \sum_{j_1,j_2=0}^{n_f} W^{\mathrm{F}}_{ij_1} W^{\mathrm{F}}_{ij_2} \sum_{\tilde{\alpha}\in\mathcal{A}} \phi_{j_0}(x_{\tilde{\alpha}}) \psi_{j_1 j_2}(x_{\tilde{\alpha}}) + O\Big(\epsilon^2\Big) \,. \end{aligned} \tag{11.116}$$

Here on the right-hand side, the first two terms actually cancel each other, since the free solution satisfies

$$\sum_{j=0}^{n_f} W^{\mathrm{F}}_{ij} \phi_j(x_{\tilde{\alpha}}) = y_{i;\tilde{\alpha}} \,, \tag{11.117}$$

i.e. for overparameterized models, linear part of the optimized model can correctly predict all the training-set examples.[14] After making that cancellation, we can multiply by the inverse $(M^{-1})_{j_0 j}$ to find a solution:

$$W^{\mathrm{I}}_{ij} = -\frac{\epsilon}{2} \sum_{j_1,j_2,j_3=0}^{n_f} W^{\mathrm{F}}_{ij_1} W^{\mathrm{F}}_{ij_2} \sum_{\tilde{\alpha}\in\mathcal{A}} \left[\psi_{j_1 j_2}(x_{\tilde{\alpha}}) \phi_{j_3}(x_{\tilde{\alpha}})\right] \Big(M^{-1}\Big)_{j_3 j} + O\Big(\epsilon^2\Big) \,. \tag{11.118}$$

In particular, the free solution, (11.115), and the interacting solution, (11.118), together solve the nonlinear optimization problem (11.111) to order ϵ.[15]

Finally, having obtained the solution, we can throw away the training data and simply store the optimal parameters $W^{\star}_{ij} = W^{\mathrm{F}}_{ij} + W^{\mathrm{I}}_{ij}$, making predictions on novel test

[14] This is shown in detail in §10.4.2. Note that for *underparametrized* models, the solution can still be analyzed, but the details will be different. We're focusing on overparameterized models here since *(i)* deep learning models are typically overparameterized and *(ii)* we suspect that the sort of representation learning our model exhibits is most useful in that regime. We'll elaborate on this quite a bit more in Epilogue ε.

[15] When doing nearly-linear quadratic regression practically, it would probably make the most sense to first find the optimal linear parameters W^{F}_{ij} and then plug them back into (11.118) to find the additional nonlinear parameters W^{I}_{ij}.

inputs $x_{\dot\beta}$ as

$$\begin{aligned} z_i(x_{\dot\beta};\theta^\star) =& \sum_{j=0}^{n_f} W^\star_{ij}\phi_j(x_{\dot\beta}) + \frac{\epsilon}{2}\sum_{j_1,j_2=0}^{n_f} W^\star_{ij_1}W^\star_{ij_2}\psi_{j_1j_2}(x_{\dot\beta}) \\ =& \sum_{j=0}^{n_f} W^{\mathrm{F}}_{ij}\phi_j(x_{\dot\beta}) + \sum_{j=0}^{n_f} W^{\mathrm{I}}_{ij}\phi_j(x_{\dot\beta}) + \frac{\epsilon}{2}\sum_{j_1,j_2=0}^{n_f} W^{\mathrm{F}}_{ij_1}W^{\mathrm{F}}_{ij_2}\psi_{j_1j_2}(x_{\dot\beta}) + O\Big(\epsilon^2\Big) \\ =& \frac{1}{2}\sum_{j=0}^{n_f} W^\star_{ij}\left[\phi_j(x_{\dot\beta}) + \phi^{\mathrm{E}}_{ij}(x_{\dot\beta};\theta^\star)\right] + O\Big(\epsilon^2\Big)\,. \end{aligned} \tag{11.119}$$

Here, we've given two different ways to think about the optimal output. On the first and second lines, we simply have the prediction of the quadratic model (11.105) expressed in terms of the fixed features $\phi_j(x_{\dot\beta})$ and the fixed meta features $\psi_{j_1j_2}(x_{\dot\beta})$. This presentation makes the nonlinearity manifest. After regrouping the terms and using our definition (11.107), in the last line we instead wrote the model prediction in the form of a linear model, where we see that features in this interpretation are the mean of the fixed unlearned features and the learned effective features. From this perspective, representation learning is manifest: the effective features $\phi^{\mathrm{E}}_{ij}(x_{\dot\beta};\theta^\star)$ depend on the training data through the optimal parameters, $W^\star_{ij} = W^\star_{ij}(x_{\tilde\alpha}, y_{\tilde\alpha})$.

In summary, our nonlinear quadratic model (11.105) serves as a minimal model of representation learning. As we will see soon, this captures the mechanism of feature evolution for a nonzero but fixed dNTK.

Aside: Model Comparison of Linear Regression and Quadratic Regression

Before we move on to the dual sample-space description of the quadratic model, let's briefly perform a model comparison between linear regression and quadratic regression. In particular, let's think about the *model complexity* of these classes of models.[16]

For both linear and quadratic regression, the number of model parameters is given by the number of elements in the combined weight matrix W_{ij}:

$$P \equiv n_{\mathrm{out}} \times (n_f + 1)\,. \tag{11.120}$$

Since both models completely memorize the same training set for a fixed and equal number of parameters, we obviously cannot naively use the Occam's razor heuristic (§6.2.2) for model comparison. This makes our model comparison somewhat subtle.

On the one hand, there is a sense in which the quadratic model is more complicated, as it computes far more functions of the input per parameter. Specifically, on a per parameter basis, we need to specify a far greater number of underlying functions for the quadratic model than we do for the linear model: i.e. we need

$$(n_f+1) + \left[\frac{1}{2}(n_f+1)(n_f+2)\right] = O\Big(P^2\Big) \tag{11.121}$$

[16] For a further discussion of model complexity, with a direct focus on overparameterized deep learning models, see Epilogue ε.

numbers to specify $\phi_j(x)$ *and* $\psi_{j_1 j_2}(x)$, while we need *just*

$$(n_f + 1) = O(P) \tag{11.122}$$

numbers to specify $\phi_j(x)$. In particular, the counting of the model functions is dominated by the meta feature functions $\psi_{j_1 j_2}(x)$. As such, this type of complexity is not really captured by the counting of model parameters, P; instead, it is expressed in the structure of the model, with the addition of the meta feature functions $\psi_{j_1 j_2}(x)$.

On the other hand, we can interpret these additional meta feature functions as *constraining* the quadratic model according to an explicit inductive bias for representation learning.[17] In particular, this additional structure alters the linear model solution (11.115) with the addition of $O(\epsilon)$ tunings W^{I}_{ij}, constrained by the $O\big(P^2\big)$ meta features that are defined before any learning takes place. Assuming these meta feature functions are *useful*, we might expect that the quadratic model will overfit less and generalize better.[18] (In fact, that was the whole point of introducing them.)

This latter point is worth a little further discussion. One typical signature of overfitting is that the parameters are extremely **finely-tuned**; these tunings are in some sense *unnatural* as they can arise from the extreme flexibility afforded to overparameterized models, enabling models to pass through all the training points *exactly*, to the extreme detriment of the test predictions.[19] Adding a regularization term on the parameter norm – i.e. the one we just discussed in footnote 17 – combats such tuning: the additional constraints on the optimization problem drive the norm of the parameters towards zero, effectively promoting parameter sparsity. Here, we see that since the nonlinear contribution to the optimal weights, W^{I}_{ij}, is fixed to be small, $O(\epsilon)$, it's adding constraints that

[17] Similarly, we could naively think that the addition of a *regularization* term such as $\sum_{\mu=1}^{P} a_\mu \theta_\mu^2$ to the loss as making a model more complex with its extra structure, despite being a well-known remedy for overfitting. Instead, it's probably better to think of this regularization term as an inductive bias for *constraining* the norm squared of the optimized model parameters.

[18] Interestingly, for the quadratic model, the number of *effective feature functions* (11.107), is actually the same as the number of model parameters: $n_{\mathrm{out}} \times (n_f + 1) = P$. Since it's only through these effective features that the meta feature functions enter the model predictions, cf. (11.119), this further underscores that, despite the additional model structure, there aren't actually $O\big(P^2\big)$ independent degrees of freedom that can be applied towards fitting the training data.

[19] This is very commonly illustrated by using polynomial basis of feature functions for linear regression, which is sometimes called *polynomial regression*. In this case, consider a linear model of a scalar function $f(x)$ with a scalar input x:

$$z(x_\delta; \theta) = \sum_{j=0}^{n_f} w_j \phi_j(x_\delta) \equiv w_0 + w_1 x_\delta + w_2 x_\delta^2 + \cdots + w_{n_f} x_\delta^{n_f} \,. \tag{11.123}$$

If there's any noise at all in the data, when the model is overparameterized, $n_f + 1 > N_{\mathcal{A}}$, the plot of this one-dimensional function will make $\sim n_f$ wild turns to go through the $N_{\mathcal{A}}$ training points. (This is particularly evocative if the target function is a simple linear function with noise, i.e. $f(x) = ax + b + \varepsilon$, with ε a zero-mean Gaussian noise with small variance $\sigma_\varepsilon^2 \ll 1$.) In order to make these turns, the optimal coefficients, $w_j^\star$, computed by (11.115), will be finely-tuned to many significant figures. This kind of fine-tuning problem in model parameters is indicative of the model being unnatural or wrong; in fact, the analog of this problem in high-energy theoretical physics is called *naturalness* (see e.g. [66] for a non-technical discussion).

– if they're useful – can combat any fine tunings that may appear in the linear solution, W_{ij}^{F}, and lead to better generalization.

11.4.2 Nearly-Kernel Methods

Now that we have some more parameter-space intuition for the potential advantages of nonlinear models over linear models, let's now develop a *dual* sample-space description of quadratic regression where a quadratic-model analog of the dNTK appears naturally.

Starting with the expression in the second line of the prediction formula (11.119) and plugging in the free solution (11.115) and the interacting solution (11.118), we get

$$\begin{aligned} z_i(x_{\dot\beta};\theta^\star) = &\sum_{\tilde\alpha\in\mathcal{A}} y_{i;\tilde\alpha}\left[\sum_{j_1,j_2=0}^{n_f}\phi_{j_1}(x_{\tilde\alpha})\left(M^{-1}\right)_{j_1j_2}\phi_{j_2}(x_{\dot\beta})\right] \\ &+\frac{\epsilon}{2}\sum_{\tilde\alpha_1,\tilde\alpha_2\in\mathcal{A}} y_{i;\tilde\alpha_1}y_{i;\tilde\alpha_2}\sum_{j_1,j_2,j_3,j_4=0}^{n_f}\phi_{j_1}(x_{\tilde\alpha_1})\left(M^{-1}\right)_{j_1j_3}\phi_{j_2}(x_{\tilde\alpha_2})\left(M^{-1}\right)_{j_2j_4} \\ &\times\left[\psi_{j_3j_4}(x_{\dot\beta})-\sum_{\tilde\alpha\in\mathcal{A}}\psi_{j_3j_4}(x_{\tilde\alpha})\sum_{j_5,j_6=0}^{n_f}\phi_{j_5}(x_{\tilde\alpha})\left(M^{-1}\right)_{j_5j_6}\phi_{j_6}(x_{\dot\beta})\right]+O\left(\epsilon^2\right)\,. \end{aligned} \tag{11.124}$$

To simplify this expression, recall formula (10.131) that we derived when discussing *kernel methods*,

$$\sum_{j_1,j_2=0}^{n_f}\phi_{j_1}(x_{\tilde\alpha})\left(M^{-1}\right)_{j_1j_2}\phi_{j_2}(x_{\dot\beta}) = k_{\dot\beta\tilde\alpha_1}\widetilde{k}^{\tilde\alpha_1\tilde\alpha}\,, \tag{11.125}$$

where the *kernel* was defined in (10.127) as

$$k_{\delta_1\delta_2}\equiv k(x_{\delta_1},x_{\delta_2})\equiv\sum_{j=0}^{n_f}\phi_j(x_{\delta_1})\,\phi_j(x_{\delta_2})\,, \tag{11.126}$$

and provided a measure of similarity between two inputs $x_{i;\delta_1}$ and $x_{i;\delta_2}$ in feature space. Plugging this formula (11.125) back into our quadratic regression prediction formula (11.124), we get

$$\begin{aligned} z_i(x_{\dot\beta};\theta^\star) = &\sum_{\tilde\alpha_1,\tilde\alpha_2\in\mathcal{A}} k_{\dot\beta\tilde\alpha_1}\widetilde{k}^{\tilde\alpha_1\tilde\alpha_2}y_{i;\tilde\alpha_2} \\ &+\frac{\epsilon}{2}\sum_{\tilde\alpha_1,\tilde\alpha_2\in\mathcal{A}} y_{i;\tilde\alpha_1}y_{i;\tilde\alpha_2}\sum_{j_1,j_2,j_3,j_4=0}^{n_f}\phi_{j_1}(x_{\tilde\alpha_1})\left(M^{-1}\right)_{j_1j_3}\phi_{j_2}(x_{\tilde\alpha_2})\left(M^{-1}\right)_{j_2j_4} \\ &\times\left[\psi_{j_3j_4}(x_{\dot\beta})-\sum_{\tilde\alpha_1,\tilde\alpha_2\in\mathcal{A}}k_{\dot\beta\tilde\alpha_1}\widetilde{k}^{\tilde\alpha_1\tilde\alpha_2}\psi_{j_3j_4}(x_{\tilde\alpha_2})\right]+O\left(\epsilon^2\right)\,, \end{aligned} \tag{11.127}$$

which is already starting to look a little better.

To simplify this expression further, we need to understand an object of the following form:

$$\sum_{j_1,j_2,j_3,j_4=0}^{n_f} \epsilon\, \phi_{j_1}(x_{\tilde{\alpha}_1}) \left(M^{-1}\right)_{j_1 j_3} \phi_{j_2}(x_{\tilde{\alpha}_2}) \left(M^{-1}\right)_{j_2 j_4} \psi_{j_3 j_4}(x_\delta)\,. \tag{11.128}$$

Taking inspiration from the steps (10.130) that we took to derive our kernel-method formula (11.125), let's act on this object with two training-set kernels:

$$\begin{aligned}
&\sum_{\tilde{\alpha}_1,\tilde{\alpha}_2 \in \mathcal{A}} \left[\sum_{j_1,j_2,j_3,j_4=0}^{n_f} \epsilon\, \phi_{j_1}(x_{\tilde{\alpha}_1}) \left(M^{-1}\right)_{j_1 j_3} \phi_{j_2}(x_{\tilde{\alpha}_2}) \left(M^{-1}\right)_{j_2 j_4} \psi_{j_3 j_4}(x_\delta) \right] \widetilde{k}_{\tilde{\alpha}_1 \tilde{\alpha}_3} \widetilde{k}_{\tilde{\alpha}_2 \tilde{\alpha}_4} \\
&= \sum_{\tilde{\alpha}_1,\tilde{\alpha}_2 \in \mathcal{A}} \sum_{j_1,\ldots,j_6=0}^{n_f} \epsilon\, \phi_{j_1}(x_{\tilde{\alpha}_1}) \left(M^{-1}\right)_{j_1 j_3} \phi_{j_2}(x_{\tilde{\alpha}_2}) \left(M^{-1}\right)_{j_2 j_4} \\
&\qquad \times \psi_{j_3 j_4}(x_\delta) \phi_{j_5}(x_{\tilde{\alpha}_1}) \phi_{j_5}(x_{\tilde{\alpha}_3}) \phi_{j_6}(x_{\tilde{\alpha}_2}) \phi_{j_6}(x_{\tilde{\alpha}_4}) \\
&= \sum_{j_1,\ldots,j_6=0}^{n_f} \epsilon\, M_{j_1 j_5} \left(M^{-1}\right)_{j_1 j_3} M_{j_2 j_6} \left(M^{-1}\right)_{j_2 j_4} \psi_{j_3 j_4}(x_\delta) \phi_{j_5}(x_{\tilde{\alpha}_3}) \phi_{j_6}(x_{\tilde{\alpha}_4}) \\
&= \sum_{j_1,j_2=0}^{n_f} \epsilon\, \psi_{j_1 j_2}(x_\delta) \phi_{j_1}(x_{\tilde{\alpha}_3}) \phi_{j_2}(x_{\tilde{\alpha}_4})\,.
\end{aligned} \tag{11.129}$$

Here on the second line, we used the definition of the kernel (11.126) to swap both kernels for feature functions, on the third line we used the definition of the symmetric matrix $M_{j_1 j_2}$, (11.113), to replace two pairs of feature functions, and on the final line we simply canceled these matrices against their inverses.

This last expression suggests that an important object worth defining is

$$\mu_{\delta_0 \delta_1 \delta_2} \equiv \sum_{j_1,j_2=0}^{n_f} \epsilon\, \psi_{j_1 j_2}(x_{\delta_0}) \phi_{j_1}(x_{\delta_1}) \phi_{j_2}(x_{\delta_2})\,, \tag{11.130}$$

which we will call the **meta kernel**.[20] Analogous to the kernel methods' kernel (11.126), the meta kernel is a parameter-independent tensor, symmetric under an exchange of its final two sample indices $\delta_1 \leftrightarrow \delta_2$, and given entirely in terms of the fixed feature and meta feature functions that define the model. One way to think about (11.130) is that for a fixed particular input, x_{δ_0}, the meta kernel computes a different feature-space inner product between the two other inputs, x_{δ_1} and x_{δ_2}. Note also that due to the inclusion of the small parameter ϵ into the definition of the meta kernel (11.130), we should think of $\mu_{\delta_0 \delta_1 \delta_2}$ as being parametrically small too.

[20] An alternate name for this object is the *differential of the kernel*, which we would consider symbolizing as $\mu_{\delta_0 \delta_1 \delta_2} \to \mathrm{d}k_{\delta_0 \delta_1 \delta_2}$. This name-symbol pair highlights the connection we're about to make to finite-width networks, but is perhaps less general in the context of making a broader model of representation learning.

With this definition, the relation (11.129) can now be succinctly summarized as

$$\begin{aligned}&\sum_{j_1,j_2,j_3,j_4=0}^{n_f} \epsilon\, \phi_{j_1}(x_{\tilde{\alpha}_1}) \left(M^{-1}\right)_{j_1 j_3} \phi_{j_2}(x_{\tilde{\alpha}_2}) \left(M^{-1}\right)_{j_2 j_4} \psi_{j_3 j_4}(x_\delta) \\ &= \sum_{\tilde{\alpha}_3,\tilde{\alpha}_4 \in \mathcal{A}} \mu_{\delta \tilde{\alpha}_3 \tilde{\alpha}_4} \widetilde{k}^{\tilde{\alpha}_3 \tilde{\alpha}_1} \widetilde{k}^{\tilde{\alpha}_4 \tilde{\alpha}_2} \,.\end{aligned} \tag{11.131}$$

Finally, plugging this simple relation back into (11.127), we get

$$\begin{aligned}z_i(x_{\dot{\beta}};\theta^\star) = &\sum_{\tilde{\alpha}_1,\tilde{\alpha}_2 \in \mathcal{A}} k_{\dot{\beta}\tilde{\alpha}_1} \widetilde{k}^{\tilde{\alpha}_1 \tilde{\alpha}_2} y_{i;\tilde{\alpha}_2} \\ &+ \frac{1}{2} \sum_{\tilde{\alpha}_1,\ldots,\tilde{\alpha}_4 \in \mathcal{A}} \left[\mu_{\dot{\beta}\tilde{\alpha}_1\tilde{\alpha}_2} - \sum_{\tilde{\alpha}_5,\tilde{\alpha}_6 \in \mathcal{A}} k_{\dot{\beta}\tilde{\alpha}_5} \widetilde{k}^{\tilde{\alpha}_5\tilde{\alpha}_6} \mu_{\tilde{\alpha}_6\tilde{\alpha}_1\tilde{\alpha}_2} \right] \left(\widetilde{k}^{\tilde{\alpha}_1\tilde{\alpha}_3} y_{i;\tilde{\alpha}_3}\right)\left(\widetilde{k}^{\tilde{\alpha}_2\tilde{\alpha}_4} y_{i;\tilde{\alpha}_4}\right) \,.\end{aligned} \tag{11.132}$$

When the prediction of a quadratic model is computed in this way, we'll hereby make it known as a *nearly-kernel machine* or **nearly-kernel methods**.[21]

Analogous to linear models, we again have two ways of thinking about the solution of our nonlinear quadratic model's predictions: on the one hand, we can use the optimal

[21] Unlike kernel methods, this solution actually depends on the details of the learning algorithm. For instance, if we had optimized the quadratic-regression loss (11.109) by *gradient descent* rather than by *direct optimization* (11.110), then we would have found instead (for zero initialization $W_{ij} = 0$)

$$\begin{aligned}z_i(x_{\dot{\beta}};\theta^\star) = &\sum_{\tilde{\alpha}_1,\tilde{\alpha}_2 \in \mathcal{A}} k_{\dot{\beta}\tilde{\alpha}_1} \widetilde{k}^{\tilde{\alpha}_1 \tilde{\alpha}_2} y_{i;\tilde{\alpha}_2} \\ &+ \sum_{\tilde{\alpha}_1,\ldots,\tilde{\alpha}_4 \in \mathcal{A}} \left[\mu_{\tilde{\alpha}_1\dot{\beta}\tilde{\alpha}_2} - \sum_{\tilde{\alpha}_5,\tilde{\alpha}_6 \in \mathcal{A}} k_{\dot{\beta}\tilde{\alpha}_5} \widetilde{k}^{\tilde{\alpha}_5\tilde{\alpha}_6} \mu_{\tilde{\alpha}_1\tilde{\alpha}_6\tilde{\alpha}_2} \right] Z_{\text{A}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4} y_{i;\tilde{\alpha}_3} y_{i;\tilde{\alpha}_4} \\ &+ \sum_{\tilde{\alpha}_1,\ldots,\tilde{\alpha}_4 \in \mathcal{A}} \left[\mu_{\dot{\beta}\tilde{\alpha}_1\tilde{\alpha}_2} - \sum_{\tilde{\alpha}_5,\tilde{\alpha}_6 \in \mathcal{A}} k_{\dot{\beta}\tilde{\alpha}_5} \widetilde{k}^{\tilde{\alpha}_5\tilde{\alpha}_6} \mu_{\tilde{\alpha}_6\tilde{\alpha}_1\tilde{\alpha}_2} \right] Z_{\text{B}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4} y_{i;\tilde{\alpha}_3} y_{i;\tilde{\alpha}_4}\end{aligned} \tag{11.133}$$

for our nearly-kernel methods prediction formula, where the *algorithm projectors* are given by

$$Z_{\text{A}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4} \equiv \widetilde{k}^{\tilde{\alpha}_1\tilde{\alpha}_3}\widetilde{k}^{\tilde{\alpha}_2\tilde{\alpha}_4} - \sum_{\tilde{\alpha}_5} \widetilde{k}^{\tilde{\alpha}_2\tilde{\alpha}_5} X_{\text{II}}^{\tilde{\alpha}_1\tilde{\alpha}_5\tilde{\alpha}_3\tilde{\alpha}_4} \,, \tag{11.134}$$

$$Z_{\text{B}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4} \equiv \widetilde{k}^{\tilde{\alpha}_1\tilde{\alpha}_3}\widetilde{k}^{\tilde{\alpha}_2\tilde{\alpha}_4} - \sum_{\tilde{\alpha}_5} \widetilde{k}^{\tilde{\alpha}_2\tilde{\alpha}_5} X_{\text{II}}^{\tilde{\alpha}_1\tilde{\alpha}_5\tilde{\alpha}_3\tilde{\alpha}_4} + \frac{\eta}{2} X_{\text{II}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4} \,, \tag{11.135}$$

with the tensor $X_{\text{II}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4}$ implicitly satisfying

$$\sum_{\tilde{\alpha}_3,\tilde{\alpha}_4 \in \mathcal{A}} X_{\text{II}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4} \left(\widetilde{k}_{\tilde{\alpha}_3\tilde{\alpha}_5}\delta_{\tilde{\alpha}_4\tilde{\alpha}_6} + \delta_{\tilde{\alpha}_3\tilde{\alpha}_5}\widetilde{k}_{\tilde{\alpha}_4\tilde{\alpha}_6} - \eta \widetilde{k}_{\tilde{\alpha}_3\tilde{\alpha}_5}\widetilde{k}_{\tilde{\alpha}_4\tilde{\alpha}_6} \right) = \delta^{\tilde{\alpha}_1}_{\tilde{\alpha}_5}\delta^{\tilde{\alpha}_2}_{\tilde{\alpha}_6} \,, \tag{11.136}$$

and global learning rate η. The origin of this gradient-descent solution should be clear after you traverse through §∞.2.2. Such *algorithm dependence* is to be expected for a nonlinear overparameterized model and is an important characteristic of finite-width networks as well. However, for the rest of the section we will continue to analyze the direct optimization formula, (11.132), with $Z_{\text{A}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4} = 0$ and $Z_{\text{B}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4} = \widetilde{k}^{\tilde{\alpha}_1\tilde{\alpha}_3}\widetilde{k}^{\tilde{\alpha}_2\tilde{\alpha}_4}/2$.

parameters (11.115) and (11.118) to make predictions (11.119); on the other hand, we can make *nearly-kernel predictions* using the formula (11.132) in which the features, the meta features, and the model parameters do not appear. That is, we've successfully traded our feature-space quantities $\phi_j(x)$, $\psi_{j_1 j_2}(x)$, and $W^\star_{ij}$ to the sample-space quantities $k_{\delta\tilde{\alpha}}$, $\mu_{\delta_0\tilde{\alpha}_1\tilde{\alpha}_2}$, and $y_{i;\tilde{\alpha}}$. As was the case before for kernel methods, this works because all the feature indices are contracted in our prediction formula (11.119), and so only combinations of the form $k_{\delta\tilde{\alpha}}$ and $\mu_{\delta_0\tilde{\alpha}_1\tilde{\alpha}_2}$ ever show up in the result and not the value of the features or meta features themselves.[22] This duality between the microscopic feature-space description of the model and a macroscopic sample-space description is another realization of the effective theory approach discussed in §0.1, and we will return to comment more broadly on this duality in Epilogue ε after we discuss the dynamics of finite-width networks in §∞.

Finally, as we saw before for kernel methods, the nearly-kernel prediction is computed by direct comparison with previously-seen examples. In this case, it has the same piece linear in the true outputs proportional to $y_{i;\tilde{\alpha}_2}$, and also has a new piece that's quadratic in the true output across different training examples proportional to $y_{i;\tilde{\alpha}_1} y_{i;\tilde{\alpha}_2}$. In this way, nearly-kernel methods are also *memory-based* methods that involve memorizing the entire training set.

Trained-Kernel Prediction

Even though these nearly-kernel methods are *very-nearly* kernel methods, there's a real qualitative difference between them due to the presence of interactions between the parameters. In the feature-space picture described in §11.4.1, this difference manifested itself in terms of the nontrivial feature learning for the effective features $\phi^{\mathrm{E}}_{ij}(x,\theta)$, as expressed in the last line of the quadratic model prediction formula (11.119). To better understand this from the dual sample-space picture, let's analogously define an **effective kernel**

$$k^{\mathrm{E}}_{ii;\delta_1\delta_2}(\theta) \equiv \sum_{j=0}^{n_f} \phi^{\mathrm{E}}_{ij}(x_{\delta_1};\theta)\, \phi^{\mathrm{E}}_{ij}(x_{\delta_2};\theta)\,, \tag{11.137}$$

which measures a parameter-dependent similarity between two inputs x_{δ_1} and x_{δ_2} using our effective features (11.107). Interestingly, we see that the model actually gives a different effective kernel for each output component i.[23] Let's try to understand this a

[22] Just as we discussed for kernel methods in footnote 46 of §10, in some situations we expect that specifying and evaluating the meta kernel $\mu_{\delta_0\delta_1\delta_2}$ is much simpler than specifying and evaluating meta feature function $\psi_{j_1 j_2}(x)$. Although picking these out of the thin air seems difficult, perhaps there are other inspired ways of determining these functions that don't require an underlying description in terms of neural networks. It would be interesting to determine the necessary and sufficient conditions for a general three-input function, $\mu(x_{\delta_0}, x_{\delta_1}, x_{\delta_2}) \equiv \mu_{\delta_0\delta_1\delta_2}$, to be a meta kernel.

[23] Here, the use of two i's in the subscript of the effective kernel to represent the output-component is just our convention; we'll later require a version with off-diagonal components in the slightly-less minimal model (11.146).

little better by evaluating the effective kernel at the end of training:

$$\begin{aligned} k^{\mathrm{E}}_{ii;\delta_1\delta_2}(\theta^\star) \equiv& \sum_{j=0}^{n_f} \phi^{\mathrm{E}}_{ij}(x_{\delta_1};\theta^\star)\, \phi^{\mathrm{E}}_{ij}(x_{\delta_2};\theta^\star) \\ =& \sum_{j=0}^{n_f} \phi_j(x_{\delta_1})\, \phi_j(x_{\delta_2}) + \sum_{j_1,j_2=0}^{n_f} W^{\mathrm{F}}_{ij_1} \left[\psi_{j_1 j_2}(x_{\delta_1})\phi_{j_2}(x_{\delta_2}) + \psi_{j_1 j_2}(x_{\delta_2})\phi_{j_2}(x_{\delta_1})\right] + O\Big(\epsilon^2\Big) \\ =& k_{\delta_1\delta_2} + \sum_{\tilde{\alpha}_1,\tilde{\alpha}_2 \in \mathcal{A}} (\mu_{\delta_1\delta_2\tilde{\alpha}_1} + \mu_{\delta_2\delta_1\tilde{\alpha}_1}) \widetilde{k}^{\tilde{\alpha}_1\tilde{\alpha}_2} y_{i;\tilde{\alpha}_2} + O\Big(\epsilon^2\Big)\ . \end{aligned} \tag{11.138}$$

To get this last result on the final line we plugged in the free solution (11.115), and then secretly used the following relation

$$\phi_{j_0}(x_{\tilde{\alpha}}) \left(M^{-1}\right)_{j_0 j_1} \psi_{j_1 j_2}(x_{\delta_1})\phi_{j_2}(x_{\delta_2}) = \sum_{\tilde{\alpha}_1 \in \mathcal{A}} \mu_{\delta_1\delta_2\tilde{\alpha}_1} \widetilde{k}^{\tilde{\alpha}_1\tilde{\alpha}}\,, \tag{11.139}$$

which can be derived with manipulations analogous to those that we used in (10.130) and (11.129).[24] Here, in (11.138) we see that the effective kernel is shifted from the kernel and includes a contribution proportional to the meta kernel as well as the true training outputs $y_{i;\tilde{\alpha}}$; this is what gives the effective kernel its output-component dependence.

Finally, let's define one more kernel:

$$k^{\sharp}_{ii;\delta_1\delta_2} \equiv \frac{1}{2}\left[k_{\delta_1\delta_2} + k^{\mathrm{E}}_{ii;\delta_1\delta_2}(\theta^\star)\right]\ . \tag{11.140}$$

This **trained kernel** averages between the simple kernel methods' kernel from the linear model and the learned nearly-kernel methods' effective kernel. Defining the inverse of the trained-kernel submatrix evaluated on the training set in the usual way,

$$\sum_{\tilde{\alpha}_2 \in \mathcal{A}} \widetilde{k^{\sharp}}_{ii}^{\tilde{\alpha}_1\tilde{\alpha}_2} \widetilde{k^{\sharp}}_{ii;\tilde{\alpha}_1\tilde{\alpha}_2} = \delta^{\tilde{\alpha}_1}_{\tilde{\alpha}_3}\,, \tag{11.141}$$

the utility of this final formulation is that the nearly-kernel prediction formula (11.132) can now be compressed as

$$z_i(x_{\dot{\beta}};\theta^\star) = \sum_{\tilde{\alpha}_1,\tilde{\alpha}_2 \in \mathcal{A}} k^{\sharp}_{ii;\dot{\beta}\tilde{\alpha}_1} \widetilde{k^{\sharp}}_{ii}^{\tilde{\alpha}_1\tilde{\alpha}_2} y_{i;\tilde{\alpha}_2} + O\Big(\epsilon^2\Big)\ , \tag{11.142}$$

taking the form of a *kernel prediction*, but with the benefit of nontrivial feature evolution incorporated into the trained kernel.[25] This is how representation learning manifests itself in nearly-kernel methods.

[24] Note that if we had instead optimized the quadratic-regression loss, (11.109), using gradient descent, then the effective kernel at the end of training, $k^{\mathrm{E}}_{ii;\delta_1\delta_2}(\theta^\star)$, would have a different expression than the one above, (11.138), for direct optimization, cf. our discussion in footnote 21.

[25] To verify the formula, use the definition of the trained kernel, (11.140), then expand in the effective kernel using the Schwinger-Dyson equations (4.55) to evaluate the matrix inverse. The result should agree with the nearly-kernel prediction formula (11.132).

Finally, note that in our minimal model of representation learning, there's no *wiring* or mixing among the n_{out} different output components: while the prediction $z_i(x_{\dot{\beta}}; \theta^\star)$ is quadratic in the true output $y_{i;\tilde{\alpha}}$ – most easily seen in (11.132) – it still only involves the i-th component. From the perspective of the **trained-kernel prediction**, (11.142), each output component i has a *different* trained kernel associated with its prediction, but the i-th prediction never depends on other true output components $y_{i';\tilde{\alpha}}$ with $i' \neq i$.

However, this lack of wiring is by our design; this representation-learning model is really intended to be *minimal*. To enable mixing of the output components, we'll have to slightly generalize the quadratic model. This we'll do next when we explain how finite-width networks can be described in this nearly-kernel methods framework.

11.4.3 Finite-Width Networks as Nonlinear Models

While the discussion so far in this section has been somewhat disconnected from the deep learning framework, much of it should still feel pretty familiar to you. For instance, the formula for the effective kernel at the end of training, (11.138), seems like it could be related to the update to the NTK, (11.10), if we identify the meta kernel $\mu_{\delta_0\delta_1\delta_2}$ with the dNTK $\widehat{\mathrm{d}H}_{i_0 i_1 i_2;\delta_0\delta_1\delta_2}$ and also make the previous identifications that we made in §10.4.3 between the kernel methods' kernel and the NTK. Let's now make these connections between finite-width networks and nonlinear models more precise.

To start, for neural networks, let us define an analog of the effective feature function $\phi^{\text{E}}_{ij}(x_\delta; \theta)$ (11.107) by

$$\phi^{\text{E}}_{i,\mu}(x_\delta; \theta) \equiv \frac{dz^{(L)}_{i;\delta}}{d\theta_\mu} \,. \tag{11.143}$$

Note that for the linear model description of infinite-width networks, the derivative of the model output is a constant, and these features are completely *fixed* throughout training. In contrast, for quadratic models and finite-width networks, the derivative (11.143) is not constant, and so these effective features evolve throughout training as the model parameters move. As for the function approximator itself, after a small change in the parameters $\theta \to \theta + \text{đ}\theta$, the network output evolves as

$$\begin{aligned} z^{(L)}_{i;\delta}(\theta + \text{đ}\theta) &= z^{(L)}_{i;\delta}(\theta) + \sum_{\mu=1}^{P} \frac{dz^{(L)}_{i;\delta}}{d\theta_\mu} \text{đ}\theta_\mu + \frac{1}{2} \sum_{\mu,\nu=1}^{P} \frac{d^2 z^{(L)}_{i;\delta}}{d\theta_\mu d\theta_\nu} \text{đ}\theta_\mu \text{đ}\theta_\nu + \ldots \,, \\ &= z^{(L)}_{i;\delta}(\theta) + \sum_{\mu=1}^{P} \phi^{\text{E}}_{i,\mu}(x_\delta; \theta)\, \text{đ}\theta_\mu + \frac{\epsilon}{2} \sum_{\mu,\nu=1}^{P} \widehat{\psi}_{i,\mu\nu}(x_\delta)\, \text{đ}\theta_\mu \text{đ}\theta_\nu + \ldots \,, \end{aligned} \tag{11.144}$$

where we've additionally defined an analog of the meta feature function $\psi_{j_1 j_2}(x_\delta)$ for neural networks by

$$\epsilon \widehat{\psi}_{i,\mu\nu}(x_\delta) \equiv \frac{d^2 z^{(L)}_{i;\delta}}{d\theta_\mu d\theta_\nu} \,. \tag{11.145}$$

For this discussion, we truncated the "…" in (11.144) so that the update to the output is exactly quadratic in the small change in the parameters. With this truncation, the

update (11.144) for a finite-width neural network is identical to the update equation (11.106) that we found for our quadratic model after taking a small step.[26]

Let us further note that for the linear model description of infinite-width networks, the meta feature functions (11.145) vanish identically – as any linear function has a zero second derivative – and thus have no effect on the dynamics. For finite-width networks with a quadratic truncation, these meta features (11.145) are parametrically small but no longer zero; they are stochastically sampled at initialization and then fixed over the course of training, hence decorated with a hat. Therefore, at quadratic order we will call these meta feature functions, $\widehat{\psi}_{i,\mu\nu}(x)$, as **random meta features**, just as we called the feature functions as *random features* for infinite-width networks.

Having established a connection in the feature space, let us now establish a similar connection in the sample-space dual description. First, associated with the effective feature functions (11.143) is the analog of the effective kernel $k^{\mathrm{E}}_{ii;\delta_1\delta_2}(\theta)$ (11.137), defined by

$$k^{\mathrm{E}}_{i_1 i_2;\delta_1\delta_2}(\theta) = \sum_{\mu,\nu} \lambda_{\mu\nu}\, \phi^{\mathrm{E}}_{i_1,\mu}(x_{\delta_1};\theta)\phi^{\mathrm{E}}_{i_2,\nu}(x_{\delta_2};\theta) = \sum_{\mu,\nu} \lambda_{\mu\nu} \frac{dz^{(L)}_{i_1;\delta_1}}{d\theta_\mu}\frac{dz^{(L)}_{i_2;\delta_2}}{d\theta_\nu} \equiv H^{(L)}_{i_1 i_2;\delta_1\delta_2}(\theta)\,. \tag{11.146}$$

Here, we used our more general definition of the kernel (10.136) to include the learning-rate tensor, and since the effective features (11.143) have a parameter dependence, in the final equality we used most general definition of the NTK, (7.17), and gave it a θ argument, $H^{(L)}_{i_1 i_2;\delta_1\delta_2}(\theta)$, to indicate its parameter dependence. In particular, if we evaluated the effective kernel at initialization $\theta = \theta(t=0)$ in terms of the random features

$$\widehat{\phi}_{i,\mu}(x_\delta) \equiv \phi^{\mathrm{E}}_{i,\mu}\Big(x_\delta;\theta(t=0)\Big) = \left.\frac{dz^{(L)}_{i;\delta}}{d\theta_\mu}\right|_{\theta=\theta(t=0)}\,, \tag{11.147}$$

we'd just have the usual L-th-layer stochastic NTK at initialization (8.5):

$$\begin{aligned}\widehat{k}_{i_1 i_2;\delta_1\delta_2} \equiv k^{\mathrm{E}}_{i_1 i_2;\delta_1\delta_2}\Big(\theta(t=0)\Big) &= \sum_{\mu,\nu} \lambda_{\mu\nu}\, \widehat{\phi}_{i_1,\mu}(x_{\delta_1})\, \widehat{\phi}_{i_2,\nu}(x_{\delta_2}) \\ &= \sum_{\mu,\nu} \lambda_{\mu\nu} \left(\frac{dz^{(L)}_{i_1;\delta_1}}{d\theta_\mu}\frac{dz^{(L)}_{i_2;\delta_2}}{d\theta_\nu}\right)\Bigg|_{\theta=\theta(t=0)} \equiv \widehat{H}^{(L)}_{i_1 i_2;\delta_1\delta_2}\,.\end{aligned} \tag{11.148}$$

For infinite-width networks, this NTK doesn't evolve during training and is composed of *random features* at initialization (10.138). In contrast, as we saw in §11.1, for finite-width networks the effective kernel (11.146) *does* evolve during training, just as the analogous effective kernel (11.137) did for the quadratic model.

[26] Considering the definition of our quadratic model, (11.105), we have included the small parameter ϵ as part of our identification. For MLPs, this parameter will be set automatically by the architecture, and is given by the effective theory cutoff, the depth-to-width ratio of the network: $\epsilon \equiv L/n$. However, for such finite-width networks there are additional terms of order $\epsilon \equiv L/n$ that need to be incorporated in order to have a consistent description, as we will explain soon.

Finally, analogously to the meta kernel for the quadratic model (11.130), we can form a meta kernel for finite-width networks from the random features (11.147) and the random meta features (11.145) as

$$\widehat{\mu}_{i_0 i_1 i_2;\delta_0\delta_1\delta_2} \equiv \sum_{\substack{\mu_1,\nu_1,\\ \mu_2,\nu_2}} \epsilon \lambda_{\mu_1\nu_1}\lambda_{\mu_2\nu_2} \widehat{\psi}_{i_0,\mu_1\mu_2}(x_{\delta_0})\, \widehat{\phi}_{i_1,\nu_1}(x_{\delta_1})\, \widehat{\phi}_{i_2,\nu_2}(x_{\delta_2}) \tag{11.149}$$

$$= \sum_{\substack{\mu_1,\nu_1,\\ \mu_2,\nu_2}} \lambda_{\mu_1\nu_1}\lambda_{\mu_2\nu_2} \left(\frac{d^2 z^{(L)}_{i_0;\delta_0}}{d\theta_{\mu_1} d\theta_{\mu_2}} \frac{dz^{(L)}_{i_1;\delta_1}}{d\theta_{\nu_1}} \frac{dz^{(L)}_{i_2;\delta_2}}{d\theta_{\nu_2}} \right) \Bigg|_{\theta=\theta(t=0)} \equiv \widehat{\mathrm{d}H}^{(L)}_{i_0 i_1 i_2;\delta_0\delta_1\delta_2} \,,$$

where we slightly generalized our earlier definition of the meta kernel (11.130) with the inclusion of the learning-rate tensors.[27] Thus, we've now identified the random meta kernel (11.149) with the L-th-layer stochastic dNTK (11.8).

With all these connections established, there are three notable differences between our minimal quadratic model and finite-width neural networks.

First, as should be clear from the definitions of the random features and random meta features, (11.147) and (11.145), these functions are stochastic rather than designed: they are determined by the details of the neural network architecture and depend on the values of the randomly-sampled parameters at initialization. We might more generally call such a quadratic model of the form (11.105) with random functions $\widehat{\phi}_j(x)$ and $\widehat{\psi}_{j_1 j_2}(x)$ a **random meta feature model**, generalizing the notion of a *random feature model* that we discussed in conjunction with infinite-width networks and linear models in §10.4.3.

Second, as we discussed at the end of §11.4.2, the quadratic model (11.105) does not wire together different components of the true outputs from the training set when making nearly-kernel predictions (11.132) on test-set inputs. In contrast, we will show soon in §∞.2.3 that the finite-width network predictions do have this wiring property. This deficiency of the quadratic model was actually by design on our part in an effort to eliminate extra complications when working through our minimal model of representation learning. To include wiring in the quadratic model, we can generalize it slightly as

$$z_i(x_\delta;\theta) = \sum_{\mu=1}^{P} \theta_\mu \widehat{\phi}_{i,\mu}(x_\delta) + \frac{\epsilon}{2} \sum_{\mu,\nu=1}^{P} \theta_\mu \theta_\nu \widehat{\psi}_{i,\mu\nu}(x_\delta)\,. \tag{11.150}$$

This slightly-less-minimal model will now allow a parameter θ_μ to connect to various different output components, as the feature functions and meta feature functions now also carry vectorial indices specifying an output-component.[28]

Third, as we've mentioned throughout this chapter, the leading finite-width contributions to the update to the network output include $O(\eta^3)$ terms. To capture these

[27] Our slightly more general definition of the meta kernel here should be understood as analogous to the slightly more general definition of the kernel (10.136).

[28] Note that these feature functions may have constraints, cf. the explicit form of the random feature (10.139). These constraints end up causing the infinite-width model not to wire, while allowing to wire the predictions of any particular network at finite width. These constraints can be thought of as a type of *weight-tying*.

effects, we need to deform our quadratic model (11.105) into a **cubic model**:

$$z_i(x_\delta;\theta) = \sum_{\mu=1}^{P} \theta_\mu \widehat{\phi}_{i,\mu}(x_\delta) + \frac{1}{2}\sum_{\mu,\nu=1}^{P} \theta_\mu\theta_\nu \widehat{\psi}_{i,\mu\nu}(x_\delta) + \frac{1}{6}\sum_{\mu,\nu,\rho=1}^{P} \theta_\mu\theta_\nu\theta_\rho \widehat{\Psi}_{i,\mu\nu\rho}(x_\delta)\,. \quad (11.151)$$

Here, the random **meta-meta feature function**, are given by the third derivative of the network output,

$$\widehat{\Psi}_{i,\mu\nu\rho}(x_\delta) \equiv \frac{d^3 z^{(L)}_{i;\delta}}{d\theta_\mu d\theta_\nu d\theta_\rho}\,; \quad (11.152)$$

the addition of this cubic term will enable the meta features to effectively evolve as if they're described by a linear model, while in turn the features will effectively evolve as if they're described by a quadratic model.[29] In summary, for finite-width networks of depth $L > 1$, this less-minimal model, (11.151), is a consistent description, with random features (11.147), random meta features (11.145), and random meta-meta features (11.152).

Deep Learning: A Non-Minimal Model of Representation Learning

Representation learning is a big part of what makes deep learning exciting. What our minimal model of representation learning has shown us is that we can actually decouple the analysis of the *learning* from the analysis of the *deep*: the simple quadratic model (11.105) exhibits nontrivial representation learning for general choices of feature functions $\phi_j(x)$ and meta feature functions $\psi_{jk}(x)$, or dually, of a kernel $k_{\delta_1\delta_2}$ and a meta kernel $\mu_{\delta_0\delta_1\delta_2}$. In particular, the meta kernel is what made learning features from the training data possible, and we hope that this broader class of representation-learning models will be of both theoretical and practical interest in their own right.

Of course, *deep* learning is a non-minimal model of representation learning, and the structure of these kernels and meta kernels *do* matter. Specifically, for deep neural networks the statistics of these functions encoded in the joint preactivation-NTK-dNTK distribution $p\Big(z^{(L)},\, \widehat{H}^{(L)},\, \widehat{\mathrm{d}H}^{(L)}\Big|\mathcal{D}\Big)$ are controlled by the representation group flow recursions – cf. §4, §8, and §11.2 – the details of which are implicitly determined by the underlying architecture and hyperparameters. In particular, we can understand the importance of this RG flow by remembering there can be a vast improvement from selecting other architectures beyond MLPs when applying function approximation to specific do-

[29] To make this connection precise, we must give the small parameter ϵ not in the cubic model definition (11.151), but instead in the statistics of the joint distribution, $p(\widehat{\phi}_{i,\mu}, \widehat{\psi}_{i,\mu\nu}, \widehat{\Psi}_{i,\mu\nu\rho})$, that controls the random meta-meta feature model. Schematically, the nontrivial combinations are the following:

$$\mathbb{E}\left[\widehat{\phi}^2\right] = O(1)\,, \qquad \mathbb{E}\left[\widehat{\psi}\,\widehat{\phi}^2 z\right] = O(\epsilon)\,, \qquad \mathbb{E}\left[\widehat{\Psi}\,\widehat{\phi}^3\right] = O(\epsilon)\,, \qquad \mathbb{E}\left[\widehat{\psi}^2\,\widehat{\phi}^2\right] = O(\epsilon)\,. \quad (11.153)$$

In the next chapter, we'll identify these combinations with the NTK, the dNTK, and (soon-to-be-revealed) two ddNTKs, respectively. Importantly, since all of these combinations are the same order in $\epsilon = L/n$, to describe finite-width networks self-consistently, we need to think of them as cubic models.

mains or datasets: RG flow *is* the inductive bias of the deep learning architecture.[30] Thus, even in the set of models that exhibit nontrivial representation learning, these choices – the initial features, meta features, and so on – are still really important.[31]

The full power of deep learning is likely due to the deep – i.e. the *representation group flow* induced by interactions between neurons in deep models of many iterated layers – working in conjunction with the learning – i.e. the *representation learning* induced by the nonlinear dynamical interactions present at finite width. The principles of deep learning theory presented in this book are precisely those that will let you analyze both of these irreducible basic elements in full generality.

[30]Note that the formalism of nonlinear models and nearly-kernel methods that we outlined in this section should also describe these other deep learning architectures so long as they admit an expansion around an infinite-width (or infinite-channel or infinite-head) limit. In particular, everything we learned here about representation learning and the training dynamics can be carried over; the only difference is that we will have have different functions $\phi_{i,\mu}(x)$ and $\psi_{i,\mu\nu}(x)$, leading to different kernels and meta kernels, $k_{i_1 i_2;\delta_1\delta_2}$ and $\mu_{i_0 i_1 i_2;\delta_0\delta_1\delta_2}$, that can be built up from a different set of recursions than the ones that we studied in this book.

[31]In Appendix B, we'll explore an aspect of this question directly by studying *residual networks*: these networks let us introduce a parameter that in a single network has an interpretation of trading off more layers of representation group flow against more effective realizations from the ensemble.

Chapter ∞

The End of Training

The job of a scientist is to listen carefully to nature, not to tell nature how to behave.

Freeman Dyson, explaining Richard Feynman's approach [67].

In this chapter, we'll finally finish our leading-order effective-theory analysis of finite-width networks and solve their training dynamics under gradient descent. In contrast to the infinite-width limit, for which the solution is independent of the training algorithm, the dynamics of such deep networks have a rich phenomenology that captures the different ways in which useful features may develop over the course of training. The solution to these training dynamics gives first-principles description of the ensemble of fully-trained finite-width networks, realizing a main goal of the book.

Unfortunately, our job will be disrupted by two facts of nature: *(i)* in order to have a consistent description of training dynamics at order $1/n$, we'll need to incorporate two additional objects that arise in the Taylor expansion of the update to the network output to third order, the update to the NTK to second order, and the update to the dNTK to first order; and *(ii)* due to a lack of smoothness, we won't be able to describe the dynamics of `ReLU` networks nor networks consisting of any of the other nonlinear activation functions from the scale-invariant universality class.

As for the first point, while the analysis of representation learning in the context of the quadratic model was illuminating, we've already telegraphed that it was insufficient to capture the particular details of finite-width networks. In particular, to leading order in $1/n$, there are two more NTK differentials, which we'll refer to as *ddNTKs*. Although it's straightforward, working out the stochastic forward equations, recursions, and effective theory for these ddNTKs is somewhat tedious, and no longer has any pedagogical value. As such, we won't provide the details of our derivations – you've already seen these sets of manipulations three times before in §4–§5, §8–§9, and §11.2–§11.3, for the preactivations, NTK, and dNTK, respectively – instead we'll simply state the results, leaving the details for you as a kind of post-training test evaluation; after all, this is the end of your training as well.

As for the second point, throughout the book we've had to use special methods in

order to work out exceptional explanations for any non-smooth activation function such as the `ReLU`. In our minds, this extra work was justified by the `ReLU`'s current privileged status as one of the most popular activation functions in practice. However, we have finally run out of tricks and will have to give up: for a reason that is simple to explain, our Taylor expansion in the global learning rate η will break down when applied to the dynamics of networks built with non-smooth activation functions. Instead, we'll have to follow the direction of the community and begin thinking again about smoothed versions of the `ReLU` – though only the ones that permit a type of criticality – such as the `GELU` and the `SWISH`.

With both those disruptions to our work heard, in §∞.1 we'll present all the relevant results for the ddNTKs – we'll define them, we'll give their tensor decomposition, and we'll explain their scaling with width and depth – while hiding all the irrelevant details at the back of the chapter in §∞.3. If you've been paying attention, you'll not be shocked to hear that – when properly normalized – the ddNTKs scale as the effective theory cutoff: ℓ/n. This scaling indicates that we need to consider the joint statistics of the preactivation-NTK-dNTK-ddNTKs in order to understand the leading-order finite-width dynamics of deep MLPs. Importantly, these ddNTKs endow the dNTK with its own dynamics; from the parameter-space perspective of §11.4.1, this means that the *meta feature functions* of the model will now evolve.

With those results stated, in §∞.2 we'll return to our regularly scheduled pedagogy and, at long last, solve the training dynamics at finite width. After an initial false start following our infinite-width giant leap, first in §∞.2.1 we'll learn how to take a small step following an adjusted giant leap, giving us our first finite-width solution. Then in §∞.2.2, we'll analyze many many steps of vanilla gradient descent, giving us our second finite-width solution. The nonlinear dynamics at finite width ultimately lead to a dependence of the fully-trained solution on the training algorithm, and so the solutions derived in these two subsections actually exhibit meaningful differences.

In particular, the function approximation of a fully-trained finite-width network can be decomposed into a universal part, independent of the optimization details, and a set of *algorithm projectors*, whose functional form encodes the entire dependence of the solution on the training algorithm. These projectors provide a dual sample-space perspective on the learning algorithm, analogous to the relationship between the model parameters and the different kernels.

Accordingly, in §∞.2.3 we'll discuss how these projectors impact the solution, letting us understand the inductive bias of the *training dynamics* separately from the inductive bias of the *network architecture*. We'll also further analyze the predictions made by such fully-trained networks, considering the growing tradeoff between increased representation learning and increased instantiation-to-instantiation fluctuations with network depth.

While this is the final chapter of the main text, in a small epilogue following this chapter, Epilogue ε, we'll explore how to define model complexity for overparameterized networks from our effective theory's macroscopic perspective. Then in two appendices, we'll further touch on some topics that are outside the scope of our main line of inquiry. In Appendix A, we'll introduce the framework of information theory, which will give us

the tools we need in order to estimate the optimal aspect ratio that separates *effectively-deep* networks from *overly-deep* networks. In Appendix B, we'll apply our effective theory approach to learn about residual networks and see how they can be used to extend the range of effectively-deep networks to greater and greater depths.

∞.1 Two More Differentials

Who ordered that?

I. I. Rabi, quipping about the $O(1/n)$ ddNTKs.

One last time, let's expand the ℓ-th-layer preactivations after a parameter update, this time recording terms up to *third order*:

$$\begin{aligned} \text{đ}z_{i;\delta}^{(\ell)} \equiv & z_{i;\delta}^{(\ell)}(t=1) - z_{i;\delta}^{(\ell)}(t=0) \\ = & \sum_{\ell_1=1}^{\ell} \sum_{\mu} \frac{dz_{i;\delta}^{(\ell)}}{d\theta_{\mu}^{(\ell_1)}} \text{đ}\theta_{\mu}^{(\ell_1)} + \frac{1}{2} \sum_{\ell_1,\ell_2=1}^{\ell} \sum_{\mu_1,\mu_2} \frac{d^2 z_{i;\delta}^{(\ell)}}{d\theta_{\mu_1}^{(\ell_1)} d\theta_{\mu_2}^{(\ell_2)}} \text{đ}\theta_{\mu_1}^{(\ell_1)} \text{đ}\theta_{\mu_2}^{(\ell_2)} \\ & + \frac{1}{6} \sum_{\ell_1,\ell_2,\ell_3=1}^{\ell} \sum_{\mu_1,\mu_2,\mu_3} \frac{d^3 z_{i;\delta}^{(\ell)}}{d\theta_{\mu_1}^{(\ell_1)} d\theta_{\mu_2}^{(\ell_2)} d\theta_{\mu_3}^{(\ell_3)}} \text{đ}\theta_{\mu_1}^{(\ell_1)} \text{đ}\theta_{\mu_2}^{(\ell_2)} \text{đ}\theta_{\mu_3}^{(\ell_3)} + \ldots . \end{aligned} \tag{∞.1}$$

For gradient descent, also recall that after use of the chain rule (11.3), the change in the ℓ_a-th-layer parameters of any particular network is given by (11.4),

$$\text{đ}\theta_{\mu}^{(\ell_a)} = -\eta \sum_{\nu} \lambda_{\mu\nu}^{(\ell_a)} \left(\sum_{j,k,\tilde{\alpha}} \frac{\partial \mathcal{L}_{\mathcal{A}}}{\partial z_{k;\tilde{\alpha}}^{(L)}} \frac{dz_{k;\tilde{\alpha}}^{(L)}}{dz_{j;\tilde{\alpha}}^{(\ell)}} \frac{dz_{j;\tilde{\alpha}}^{(\ell)}}{d\theta_{\nu}^{(\ell_a)}} \right) = -\eta \sum_{\nu,j,\tilde{\alpha}} \lambda_{\mu\nu}^{(\ell_a)} \, \epsilon_{j;\tilde{\alpha}}^{(\ell)} \frac{dz_{j;\tilde{\alpha}}^{(\ell)}}{d\theta_{\nu}^{(\ell_a)}} , \tag{∞.2}$$

where we've used our convention from §11.1 of explicitly specifying which layer each parameter comes from. Please also recall from there that the learning-rate tensor $\lambda_{\mu\nu}^{(\ell)}$ only connects the parameters within a given layer ℓ. In the above expression, ℓ is an intermediate layer such that $\ell_a \leq \ell$, and we also used our *ℓ-th-layer error factor* (11.5):

$$\epsilon_{j;\tilde{\alpha}}^{(\ell)} \equiv \sum_{k=1}^{n_L} \frac{\partial \mathcal{L}_{\mathcal{A}}}{\partial z_{k;\tilde{\alpha}}^{(L)}} \frac{dz_{k;\tilde{\alpha}}^{(L)}}{dz_{j;\tilde{\alpha}}^{(\ell)}} = \frac{d\mathcal{L}_{\mathcal{A}}}{dz_{j;\tilde{\alpha}}^{(\ell)}} . \tag{∞.3}$$

After substituting the parameter update (∞.2) back into the preactivation update (∞.1), you should be able to write it in the form

$$\begin{aligned} \text{đ}z_{i;\delta}^{(\ell)} = & -\eta \sum_{j,\tilde{\alpha}} \widehat{H}_{ij;\delta\tilde{\alpha}}^{(\ell)} \epsilon_{j;\tilde{\alpha}}^{(\ell)} + \frac{\eta^2}{2} \sum_{j_1,j_2,\tilde{\alpha}_1,\tilde{\alpha}_2} \widehat{\text{d}H}_{ij_1j_2;\delta\tilde{\alpha}_1\tilde{\alpha}_2}^{(\ell)} \epsilon_{j_1;\tilde{\alpha}_1}^{(\ell)} \epsilon_{j_2;\tilde{\alpha}_2}^{(\ell)} \\ & - \frac{\eta^3}{6} \sum_{j_1,j_2,j_3,\tilde{\alpha}_1,\tilde{\alpha}_2,\tilde{\alpha}_3} \widehat{\text{dd}_{\text{I}}H}_{ij_1j_2j_3;\delta\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3}^{(\ell)} \epsilon_{j_1;\tilde{\alpha}_1}^{(\ell)} \epsilon_{j_2;\tilde{\alpha}_2}^{(\ell)} \epsilon_{j_3;\tilde{\alpha}_3}^{(\ell)} , \\ & + O\Big(\eta^4\Big) \end{aligned} \tag{∞.4}$$

where the first two terms we found in the last chapter (11.9), and the cubic term is new, with the *first* of the **ddNTKs** defined as

$$\widehat{\mathrm{dd_I}H}^{(\ell)}_{i_0 i_1 i_2 i_3;\delta_0\delta_1\delta_2\delta_3} \equiv \sum_{\ell_1,\ell_2,\ell_3=1}^{\ell} \sum_{\substack{\mu_1,\nu_1,\\ \mu_2,\nu_2,\\ \mu_3,\nu_3}} \lambda^{(\ell_1)}_{\mu_1\nu_1}\lambda^{(\ell_2)}_{\mu_2\nu_2}\lambda^{(\ell_3)}_{\mu_3\nu_3} \frac{d^3 z^{(\ell)}_{i_0;\delta_0}}{d\theta^{(\ell_1)}_{\mu_1} d\theta^{(\ell_2)}_{\mu_2} d\theta^{(\ell_3)}_{\mu_3}} \frac{dz^{(\ell)}_{i_1;\delta_1}}{d\theta^{(\ell_1)}_{\nu_1}} \frac{dz^{(\ell)}_{i_2;\delta_2}}{d\theta^{(\ell_2)}_{\nu_2}} \frac{dz^{(\ell)}_{i_3;\delta_3}}{d\theta^{(\ell_3)}_{\nu_3}} \,. \tag{∞.5}$$

As always, the hat on the ddNTK indicates that it's stochastic, depending on the particular realization of the model parameters at initialization. Also, similar to the dNTK, this ddNTK is totally symmetric in its second, third, and fourth paired set of indices $(i_1,\delta_1) \leftrightarrow (i_2,\delta_2) \leftrightarrow (i_3,\delta_3)$, while the first neural-sample index (i_0,δ_0) is distinguished from the other three.

By expanding to order η^3, the update, (∞.4), is now *cubic* in error factors. Expanding to this order is necessary because the first ddNTK has statistics at initialization that are $O(1/n)$, and so needs to be included in our analysis. However, any higher-order terms in the update are subleading, so we may replace $O(\eta^4) = O(1/n^2)$ in this expression.

Just as we had to expand the update to the NTK to order η when we expanded the update to the preactivations to order η^2, we will now have to expand the update to the NTK to order η^2 for the dynamics with our cubic update (∞.4) to be consistent:

$$\begin{aligned}
\text{đ}H^{(\ell)}_{i_1 i_2;\delta_1\delta_2} \equiv& H^{(\ell)}_{i_1 i_2;\delta_1\delta_2}(t=1) - H^{(\ell)}_{i_1 i_2;\delta_1\delta_2}(t=0) \qquad (\infty.6)\\
=& \sum_{\ell_1=1}^{\ell} \sum_{\mu_1} \frac{dH^{(\ell)}_{i_1 i_2;\delta_1\delta_2}}{d\theta^{(\ell_1)}_{\mu_1}} \text{đ}\theta^{(\ell_1)}_{\mu_1} + \sum_{\ell_1,\ell_2=1}^{\ell} \sum_{\mu_1,\mu_2} \frac{d^2 H^{(\ell)}_{i_1 i_2;\delta_1\delta_2}}{d\theta^{(\ell_1)}_{\mu_1} d\theta^{(\ell_2)}_{\mu_2}} \text{đ}\theta^{(\ell_1)}_{\mu_1} \text{đ}\theta^{(\ell_2)}_{\mu_2} + \ldots \\
=& -\eta \sum_{j,\tilde{\alpha}} \left(\widehat{\mathrm{d}H}^{(\ell)}_{i_1 i_2 j;\delta_1\delta_2\tilde{\alpha}} + \widehat{\mathrm{d}H}^{(\ell)}_{i_2 i_1 j;\delta_2\delta_1\tilde{\alpha}} \right) \epsilon^{(\ell)}_{j;\tilde{\alpha}} \\
&+ \frac{\eta^2}{2} \sum_{j_1,j_2,\tilde{\alpha}_1,\tilde{\alpha}_2} \left[\widehat{\mathrm{dd_I}H}^{(\ell)}_{i_1 i_2 j_1 j_2;\delta_1\delta_2\tilde{\alpha}_1\tilde{\alpha}_2} + \widehat{\mathrm{dd_I}H}^{(\ell)}_{i_2 i_1 j_1 j_2;\delta_2\delta_1\tilde{\alpha}_1\tilde{\alpha}_2} \right] \epsilon^{(\ell)}_{j_1;\tilde{\alpha}_1} \epsilon^{(\ell)}_{j_2;\tilde{\alpha}_2} \\
&+ \eta^2 \sum_{j_1,j_2,\tilde{\alpha}_1,\tilde{\alpha}_2} \widehat{\mathrm{dd_{II}}H}^{(\ell)}_{i_1 i_2 j_1 j_2;\delta_1\delta_2\tilde{\alpha}_1\tilde{\alpha}_2} \epsilon^{(\ell)}_{j_1;\tilde{\alpha}_1} \epsilon^{(\ell)}_{j_2;\tilde{\alpha}_2} + O\left(\eta^3\right) \,.
\end{aligned}$$

To go to the final equality, we substituted in our NTK definition (11.7) and our parameter update (∞.2), computed the derivatives, and then collected the terms. To do so, we identified the *second* of the **ddNTKs**, defined as

$$\widehat{\mathrm{dd_{II}}H}^{(\ell)}_{i_1 i_2 i_3 i_4;\delta_1\delta_2\delta_3\delta_4} \equiv \sum_{\ell_1,\ell_2,\ell_3=1}^{\ell} \sum_{\substack{\mu_1,\nu_1,\\ \mu_2,\nu_2,\\ \mu_3,\nu_3}} \lambda^{(\ell_1)}_{\mu_1\nu_1}\lambda^{(\ell_2)}_{\mu_2\nu_2}\lambda^{(\ell_3)}_{\mu_3\nu_3} \frac{d^2 z^{(\ell)}_{i_1;\delta_1}}{d\theta^{(\ell_1)}_{\mu_1} d\theta^{(\ell_3)}_{\mu_3}} \frac{d^2 z^{(\ell)}_{i_2;\delta_2}}{d\theta^{(\ell_2)}_{\mu_2} d\theta^{(\ell_3)}_{\nu_3}} \frac{dz^{(\ell)}_{i_3;\delta_3}}{d\theta^{(\ell_1)}_{\nu_1}} \frac{dz^{(\ell)}_{i_4;\delta_4}}{d\theta^{(\ell_2)}_{\nu_2}} \,. \tag{∞.7}$$

The hat on this ddNTK indicates that it's also stochastic at initialization, and we will soon detail that it also has $O(1/n)$ statistics at leading order. Finally, $\widehat{\mathrm{dd_{II}}H}_{i_1 i_2 i_3 i_4;\delta_1\delta_2\delta_3\delta_4}$

has a more constrained symmetry, only symmetric under a joint swap of the paired set of indices as $(i_1, \delta_1) \leftrightarrow (i_2, \delta_2)$ *and* $(i_3, \delta_4) \leftrightarrow (i_4, \delta_4)$. However, this means that we can also swap indices as

$$\sum_{j_1,j_2,\tilde{\alpha}_1,\tilde{\alpha}_2} \widehat{\mathrm{dd_{II}}H}^{(\ell)}_{i_1 i_2 j_1 j_2;\delta_1\delta_2\tilde{\alpha}_1\tilde{\alpha}_2} \epsilon^{(\ell)}_{j_1;\tilde{\alpha}_1} \epsilon^{(\ell)}_{j_2;\tilde{\alpha}_2} = \sum_{j_1,j_2,\tilde{\alpha}_1,\tilde{\alpha}_2} \widehat{\mathrm{dd_{II}}H}^{(\ell)}_{i_2 i_1 j_1 j_2;\delta_2\delta_1\tilde{\alpha}_1\tilde{\alpha}_2} \epsilon^{(\ell)}_{j_1;\tilde{\alpha}_1} \epsilon^{(\ell)}_{j_2;\tilde{\alpha}_2} \,, \tag{∞.8}$$

which we used to simplify the final line of the NTK update (∞.6).

Overall, this NTK update is now *quadratic* in error factors, making its dynamics coupled and nonlinear. Again, expanding to this order is necessary because both ddNTKs have statistics at initialization that are $O(1/n)$, and so both need to be included in our analysis. However, any higher-order terms in the NTK update are subleading, so in (∞.6) we may replace $O(\eta^3) = O(1/n^2)$.

Finally, consider the leading-order update to the dNTK:

$$\begin{aligned} đ\mathrm{d}H^{(\ell)}_{i_0 i_1 i_2;\delta_0\delta_1\delta_2} \equiv& \mathrm{d}H^{(\ell)}_{i_0 i_1 i_2;\delta_0\delta_1\delta_2}(t=1) - \mathrm{d}H^{(\ell)}_{i_0 i_1 i_2;\delta_0\delta_1\delta_2}(t=0) \\ =& \sum_{\ell_1=1}^{\ell} \sum_{\mu_1} \frac{d\mathrm{d}H^{(\ell)}_{i_0 i_1 i_2;\delta_0\delta_1\delta_2}}{d\theta^{(\ell_1)}_{\mu_1}} đ\theta^{(\ell_1)}_{\mu_1} + \ldots \\ =& -\eta \sum_{j,\tilde{\alpha}} \left(\widehat{\mathrm{dd_I}H}^{(\ell)}_{i_0 i_1 i_2 j;\delta_0\delta_1\delta_2\tilde{\alpha}} + \widehat{\mathrm{dd_{II}}H}^{(\ell)}_{i_0 i_1 i_2 j;\delta_0\delta_1\delta_2\tilde{\alpha}} + \widehat{\mathrm{dd_{II}}H}^{(\ell)}_{i_0 i_2 i_1 j;\delta_0\delta_2\delta_1\tilde{\alpha}} \right) \epsilon^{(\ell)}_{j;\tilde{\alpha}} \\ &+ O\Big(\eta^2\Big) \,. \end{aligned} \tag{∞.9}$$

To go to the final equality, we substituted our dNTK definition (11.8) and parameter update (∞.2), took the derivative, and then collected all the terms using our ddNTK definitions, (∞.5) and (∞.7). The dNTK update is *linear* in error factors and its dynamics will be the simplest. Finally, the higher-order terms in the dNTK update are subleading, so in (∞.9) we may replace $O(\eta^2) = O(1/n^2)$. We also would like to apologize for the $đ\mathrm{d}H$ notation, and we promise that we won't have to use it again.

The updates (∞.4), (∞.6), and (∞.9) comprise the complete set of finite-width updates at $O(1/n)$. Thus, to proceed further in our analysis, we'll need to work out the leading-order statistics of the ddNTKs.

ddNTK Statistics

To find the distribution of fully-trained finite-width networks, we need the joint distribution of the network output, the NTK, the dNTK, and the ddNTKs:

$$p\Big(z^{(L)},\, \widehat{H}^{(L)},\, \widehat{\mathrm{d}H}^{(L)},\, \widehat{\mathrm{dd_I}H}^{(L)},\, \widehat{\mathrm{dd_{II}}H}^{(L)} \Big| \mathcal{D}\Big) \,. \tag{∞.10}$$

Rather than working through the details here, we'll do it in our own private notebooks. You already have all the tools you need: you can follow §4, §8, and §11.2 for examples of how to use RG flow to work out the recursions; and you can follow §5, §9, and §11.3

for examples of how to work out the details of the effective theory at initialization after tuning to criticality. The full details of this distribution (∞.10) can be found at the end of the chapter in §∞.3. Here, we'll highlight the results that you need in order to understand our dynamical computations in the following section.

After working out the stochastic forward equations for both ddNTKs – feel free to flip forward to (∞.169) and (∞.170) if you're curious about them – we'll need to find recursions for its statistics. For these tensors, the leading-order statistics come from their means; their cross correlations and their variances are all subleading. When evaluating the mean of each of the ddNTKs, it will become convenient to decompose them into the following set of tensors with sample indices only:

$$\mathbb{E}\left[\widehat{\mathrm{dd_I}H}^{(\ell)}_{i_0 i_1 i_2 i_3;\delta_0\delta_1\delta_2\delta_3}\right] \equiv \frac{1}{n_{\ell-1}}\left[\delta_{i_0 i_1}\delta_{i_2 i_3}R^{(\ell)}_{\delta_0\delta_1\delta_2\delta_3} + \delta_{i_0 i_2}\delta_{i_3 i_1}R^{(\ell)}_{\delta_0\delta_2\delta_3\delta_1} + \delta_{i_0 i_3}\delta_{i_1 i_2}R^{(\ell)}_{\delta_0\delta_3\delta_1\delta_2}\right] , \tag{∞.11}$$

$$\mathbb{E}\left[\widehat{\mathrm{dd_{II}}H}^{(\ell)}_{i_1 i_2 i_3 i_4;\delta_1\delta_2\delta_3\delta_4}\right] \equiv \frac{1}{n_{\ell-1}}\left[\delta_{i_1 i_2}\delta_{i_3 i_4}S^{(\ell)}_{\delta_1\delta_2\delta_3\delta_4} + \delta_{i_1 i_3}\delta_{i_4 i_2}T^{(\ell)}_{\delta_1\delta_3\delta_4\delta_2} + \delta_{i_1 i_4}\delta_{i_2 i_3}U^{(\ell)}_{\delta_1\delta_4\delta_2\delta_3}\right] . \tag{∞.12}$$

Thus, there are four new objects whose recursions we need to determine and whose scaling with depth we'll need to compute.

The ddNTKs both vanish in the first layer, just like the dNTK. Proceeding then from the second layer to deeper layers, after a bunch of tedious algebra and plenty of flipping around in the book to make various substitutions, you can find recursions for $R^{(\ell)}$, $S^{(\ell)}$, $T^{(\ell)}$, and $U^{(\ell)}$: (∞.172), (∞.174), (∞.175), and (∞.176), respectively. You can flip forward to take a look at them, but you probably won't want to.... Importantly, these recursions in conjunction with the decompositions (∞.11) and (∞.12) altogether demonstrate the both ddNTKs have nontrivial order-$1/n$ statistics:

$$\mathbb{E}\left[\widehat{\mathrm{dd_I}H}^{(\ell)}_{i_0 i_1 i_2 i_3;\delta_0\delta_1\delta_2\delta_3}\right] = O\left(\frac{1}{n}\right) , \qquad \mathbb{E}\left[\widehat{\mathrm{dd_{II}}H}^{(\ell)}_{i_1 i_2 i_3 i_4;\delta_1\delta_2\delta_3\delta_4}\right] = O\left(\frac{1}{n}\right) . \tag{∞.13}$$

Thus, as we've been forecasting, we must include them in our finite-width analysis of training dynamics.

Focusing on the single-input statistics, we again make the layer-independent choices $C_b^{(\ell)} = C_b$, $C_W^{(\ell)} = C_W$, and $n_1 = \cdots = n_{L-1} \equiv n$. Ignoring contributions that are subleading in $1/n$, in particular replacing the mean metric by the kernel $G^{(\ell)} \to K^{(\ell)}$ and the NTK mean by the frozen NTK $H^{(\ell)} \to \Theta^{(\ell)}$, the four recursions ($\infty$.172),

(∞.174), (∞.175), and (∞.176) together reduce to a form that at least fits on a page:

$$\begin{aligned}
R^{(\ell+1)} &= \left(\chi_\perp^{(\ell)}\right)^2 R^{(\ell)} \\
&\quad + \lambda_W^{(\ell+1)} C_W \left\langle \sigma''\sigma'\sigma'\sigma \right\rangle_{K^{(\ell)}} \left(\Theta^{(\ell)}\right)^2 + C_W^2 \left\langle \sigma'''\sigma'\sigma'\sigma' \right\rangle_{K^{(\ell)}} \left(\Theta^{(\ell)}\right)^3 \\
&\quad + \chi_\perp^{(\ell)} \left(\lambda_W^{(\ell+1)} \left\langle \sigma''\sigma \right\rangle_{K^{(\ell)}} + C_W \Theta^{(\ell)} \left\langle \sigma'''\sigma' \right\rangle_{K^{(\ell)}}\right) \left(B^{(\ell)} + P^{(\ell)}\right) \\
&\quad + \chi_\perp^{(\ell)} \left(\lambda_W^{(\ell+1)} \left\langle \sigma'\sigma' \right\rangle_{K^{(\ell)}} + C_W \Theta^{(\ell)} \left\langle \sigma''\sigma'' \right\rangle_{K^{(\ell)}}\right) P^{(\ell)} \,, \qquad (\infty.14) \\
S^{(\ell+1)} &= \left(\chi_\perp^{(\ell)}\right)^2 S^{(\ell)} \\
&\quad + C_W \lambda_W^{(\ell+1)} \left\langle \sigma'\sigma'\sigma'\sigma' \right\rangle_{K^{(\ell)}} \left(\Theta^{(\ell)}\right)^2 + C_W^2 \left\langle \sigma''\sigma''\sigma'\sigma' \right\rangle_{K^{(\ell)}} \left(\Theta^{(\ell)}\right)^3 \\
&\quad + \chi_\perp^{(\ell)} \left[\lambda_W^{(\ell+1)} \left\langle \sigma'\sigma' \right\rangle_{K^{(\ell)}} + C_W \Theta^{(\ell)} \left\langle \sigma''\sigma'' \right\rangle_{K^{(\ell)}}\right] B^{(\ell)} \,, \qquad (\infty.15) \\
T^{(\ell+1)} &= \left(\chi_\perp^{(\ell)}\right)^2 T^{(\ell)} \\
&\quad + 2 C_W \lambda_W^{(\ell+1)} \left\langle \sigma''\sigma'\sigma'\sigma \right\rangle_{K^{(\ell)}} \left(\Theta^{(\ell)}\right)^2 + C_W^2 \left\langle \sigma''\sigma''\sigma'\sigma' \right\rangle_{K^{(\ell)}} \left(\Theta^{(\ell)}\right)^3 \\
&\quad + \left(\lambda_W^{(\ell+1)}\right)^2 \left\langle \sigma'\sigma'\sigma\sigma \right\rangle_{K^{(\ell)}} \Theta^{(\ell)} \\
&\quad + \left(\lambda_W^{(\ell+1)} \left\langle z\sigma'\sigma \right\rangle_{K^{(\ell)}} + C_W \Theta^{(\ell)} \left\langle z\sigma''\sigma' \right\rangle_{K^{(\ell)}}\right)^2 \frac{F^{(\ell)}}{\left(K^{(\ell)}\right)^2} \\
&\quad + 2\chi_\perp^{(\ell)} \left[\lambda_W^{(\ell+1)} \left(\left\langle \sigma''\sigma \right\rangle_{K^{(\ell)}} + \left\langle \sigma'\sigma' \right\rangle_{K^{(\ell)}}\right) + C_W \Theta^{(\ell)} \left(\left\langle \sigma'''\sigma' \right\rangle_{K^{(\ell)}} + \left\langle \sigma''\sigma'' \right\rangle_{K^{(\ell)}}\right)\right] Q^{(\ell)} \,, \qquad (\infty.16) \\
U^{(\ell+1)} &= \left(\chi_\perp^{(\ell)}\right)^2 U^{(\ell)} + C_W^2 \left\langle \sigma''\sigma''\sigma'\sigma' \right\rangle_{K^{(\ell)}} \left(\Theta^{(\ell)}\right)^3 . \qquad (\infty.17)
\end{aligned}$$

Here, we also simplified these expressions by recalling the two susceptibilities, (5.50) and (5.51):

$$\chi_\parallel^{(\ell)} \equiv \frac{C_W}{K^{(\ell)}} \left\langle \sigma'\sigma z \right\rangle_{K^{(\ell)}} , \qquad \chi_\perp^{(\ell)} \equiv C_W \left\langle \sigma'\sigma' \right\rangle_{K^{(\ell)}} . \qquad (\infty.18)$$

ddNTK Scalings

Now, let's tune to criticality, and use our scaling ansatz (5.93) to find the critical exponents p_R, p_S, p_T, and p_U that describe the asymptotic depth scaling of $R^{(\ell)}$, $S^{(\ell)}$, $T^{(\ell)}$, and $U^{(\ell)}$, respectively.

To understand the relative size of these tensors controlling the means of the ddNTKs, we will again need to identify appropriate dimensionless ratios. Following the dimensional analysis logic from our dNTK discussion, cf. (11.63), let's look at our third-order update for preactivations (∞.4). Remembering again that we can only add terms that have the same dimensions, we see that

$$[z] = [\eta]\,[\epsilon]\,[\widehat{H}] = [\eta]^2\,[\epsilon]^2\,[\widehat{\mathrm{d}H}] = [\eta]^3\,[\epsilon]^3\,[\widehat{\mathrm{dd}_{\mathrm{I}}H}] = [\eta]^3\,[\epsilon]^3\,[\widehat{\mathrm{dd}_{\mathrm{II}}H}] \,. \qquad (\infty.19)$$

From the first equality, we see as before that $[\eta]\,[\epsilon] = [z]\,[\widehat{H}]^{-1}$, and so R, S, T, and U each have dimensions of NTK cubed:

$$[R] \equiv [\widehat{\mathrm{dd}_{\mathrm{I}} H}] = [\widehat{H}]^3\,[z]^{-2}\,, \qquad [S] = [T] = [U] \equiv [\widehat{\mathrm{dd}_{\mathrm{II}} H}] = [\widehat{H}]^3\,[z]^{-2}\,. \tag{∞.20}$$

This means that the proper dimensionless combinations for the ddNTKs are

$$\frac{R^{(\ell)} K^{(\ell)}}{n\left(\Theta^{(\ell)}\right)^3} \sim \frac{1}{n}\left(\frac{1}{\ell}\right)^{p_R + p_0 - 3p_\Theta}\,, \qquad \frac{S^{(\ell)} K^{(\ell)}}{n\left(\Theta^{(\ell)}\right)^3} \sim \frac{1}{n}\left(\frac{1}{\ell}\right)^{p_S + p_0 - 3p_\Theta}\,, \tag{∞.21}$$

$$\frac{T^{(\ell)} K^{(\ell)}}{n\left(\Theta^{(\ell)}\right)^3} \sim \frac{1}{n}\left(\frac{1}{\ell}\right)^{p_T + p_0 - 3p_\Theta}\,, \qquad \frac{U^{(\ell)} K^{(\ell)}}{n\left(\Theta^{(\ell)}\right)^3} \sim \frac{1}{n}\left(\frac{1}{\ell}\right)^{p_U + p_0 - 3p_\Theta}\,, \tag{∞.22}$$

where to get a normalized ratio, we multiplied by the kernel $K^{(\ell)}$ to account for the $[z]^{-2}$ and then divided by the three factors of the frozen NTK $\Theta^{(\ell)}$. Here, p_0 is the critical exponent for the kernel, and p_Θ is the critical exponent for the NTK.

Finally, as a brief aside, let us comment on one aspect of dimensionality that we've been ignoring but will soon become important. In particular, since the network output z is set to the true output y, they should really have the same dimensions:

$$[z] = [y]\,. \tag{∞.23}$$

However, while the leading depth scaling of the preactivations is given by the kernel,

$$\mathbb{E}\left[z^{(\ell)} z^{(\ell)}\right] = K^{(\ell)} \sim \left(\frac{1}{\ell}\right)^{p_0}\,, \tag{∞.24}$$

the true output y is fixed and doesn't scale with depth. When comparing the performance of networks of different depths, this suggests that it might be helpful to rescale the final network outputs as

$$z_{i;\delta} \to z_{i;\delta}\left(\frac{1}{L}\right)^{-p_0/2}\,, \tag{∞.25}$$

effectively fixing the scaling of the overall network output as $p_0 = 0$, or equivalently rescale the training outputs as

$$y_{i;\tilde{\alpha}} \to y_{i;\tilde{\alpha}}\left(\frac{1}{L}\right)^{p_0/2}\,. \tag{∞.26}$$

We will further see how this can affect the predictions of a fully-trained network on its test set in §∞.2.3.[1]

[1]For regression tasks, where we want to learn a vector of real numbers, this rescaling is appropriate. For classification tasks, where we want to learn a discrete probability distribution, we should rescale either the network outputs (before any softmax layer) or the raw output targets $y_{i;\delta}$ (again before any softmax layer) as in (10.37).

$K^\star = 0$ Universality Class

For the $K^\star = 0$ universality class, remember that we Taylor expanded the activation function as

$$\sigma(z) = \sum_{p=0}^{\infty} \frac{\sigma_p}{p!} z^p \,, \tag{∞.27}$$

and defined the following Taylor coefficient for convenience

$$a_1 \equiv \left(\frac{\sigma_3}{\sigma_1}\right) + \frac{3}{4}\left(\frac{\sigma_2}{\sigma_1}\right)^2 \,, \tag{∞.28}$$

and required that all activation functions in this class satisfy $\sigma_0 = 0$ and $\sigma_1 \neq 0$.

To solve our single input ddNTK recursions, (∞.14)–(∞.17), you'll have to evaluate a few new Gaussian expectations, taking particular note of that some of them now depend on the third derivative of the activation function. Finally, to tune to $K^\star = 0$ criticality (5.90), we need to set the initialization hyperparameters as $C_b = 0$ and $C_W = 1/\sigma_1^2$; to implement the learning rate equivalence principle, we need to set our training hyperparameters as (9.95),

$$\lambda_b^{(\ell)} = \widetilde{\lambda}_b \left(\frac{1}{\ell}\right)^{p_\perp} L^{p_\perp - 1} \,, \qquad \lambda_W^{(\ell)} = \widetilde{\lambda}_W \left(\frac{L}{\ell}\right)^{p_\perp - 1} \,. \tag{∞.29}$$

For simplicity, let us also focus on odd activation functions, such as `tanh`, for which importantly $\sigma_2 = 0$ and $p_\perp = 1$.

Inspecting the single-input ddNTK recursions (∞.14)–(∞.17), we see that they depend on Gaussian expectations of preactivations as well as our previous solutions for all the other objects that we've considered: the NTK variance, the NTK-preactivation cross correlation, and the dNTK-preactivation cross correlation. Substituting in the solutions for all these quantities as needed – you'll have to flip around to find them, though most were reprinted in §11.3.2 for the dNTK analysis – we can find solutions to all the single-input ddNTK recursions:

$$R^{(\ell)} = -\frac{\ell^2}{48}\left[3\widetilde{\lambda}_b + 4\frac{\widetilde{\lambda}_W \sigma_1^2}{(-a_1)}\right]\left[\widetilde{\lambda}_b + \frac{\widetilde{\lambda}_W \sigma_1^2}{(-a_1)}\right]^2 (-a_1) + \ldots \,, \tag{∞.30}$$

$$S^{(\ell)} = \frac{\ell^2}{12}\left[\widetilde{\lambda}_b + \frac{\widetilde{\lambda}_W \sigma_1^2}{(-a_1)}\right]^2 \widetilde{\lambda}_W \sigma_1^2 + \ldots \,, \tag{∞.31}$$

$$T^{(\ell)} = \frac{\ell^2}{32}\left[\widetilde{\lambda}_b + \frac{\widetilde{\lambda}_W \sigma_1^2}{(-a_1)}\right](-a_1)\widetilde{\lambda}_b^2 + \ldots \,, \tag{∞.32}$$

$$U^{(\ell)} = \frac{1}{2}\left[\widetilde{\lambda}_b + \frac{\widetilde{\lambda}_W \sigma_1^2}{(-a_1)}\right]^3 (-a_1) + \ldots \,. \tag{∞.33}$$

From these, we can read off the critical exponents as $p_R = p_S = p_T = -2$, and $p_U = 0$,

and we see that the dimensionless ratios are given by

$$\frac{R^{(\ell)}K^{(\ell)}}{n\left(\Theta^{(\ell)}\right)^3} = -\frac{1}{48}\left[\frac{3\widetilde{\lambda}_b + 4\widetilde{\lambda}_W\sigma_1^2/(-a_1)}{\widetilde{\lambda}_b + \widetilde{\lambda}_W\sigma_1^2/(-a_1)}\right]\frac{\ell}{n} + \ldots\,, \tag{∞.34}$$

$$\frac{S^{(\ell)}K^{(\ell)}}{n\left(\Theta^{(\ell)}\right)^3} = \frac{1}{12}\left[\frac{\widetilde{\lambda}_W\sigma_1^2/(-a_1)}{\widetilde{\lambda}_b + \widetilde{\lambda}_W\sigma_1^2/(-a_1)}\right]\frac{\ell}{n} + \ldots\,, \tag{∞.35}$$

$$\frac{T^{(\ell)}K^{(\ell)}}{n\left(\Theta^{(\ell)}\right)^3} = \frac{1}{32}\left[\frac{\widetilde{\lambda}_b}{\widetilde{\lambda}_b + \widetilde{\lambda}_W\sigma_1^2/(-a_1)}\right]^2\frac{\ell}{n} + \ldots\,, \tag{∞.36}$$

$$\frac{U^{(\ell)}K^{(\ell)}}{n\left(\Theta^{(\ell)}\right)^3} = \frac{1}{2}\frac{1}{\ell n} + \ldots\,. \tag{∞.37}$$

This means that $R^{(\ell)}$, $S^{(\ell)}$, and $T^{(\ell)}$ all scale according to our leading effective theory cutoff as

$$p_R + p_0 - 3p_\Theta = -1 \qquad p_S + p_0 - 3p_\Theta = -1\,, \qquad p_T + p_0 - 3p_\Theta = -1\,. \tag{∞.38}$$

Thus, we see that the leading-order finite-width dynamics of $K^\star = 0$ activation functions have contributions from the first ddNTK, $\widehat{\mathrm{dd_I}H}$, via $R^{(\ell)}$, and from the second ddNTK, $\widehat{\mathrm{dd_{II}}H}$, via $S^{(\ell)}$ and $T^{(\ell)}$.[2]

Scale-Invariant Universality Class

Perhaps the most important fact to remember about nonlinear scale-invariant activation functions (2.13),

$$\sigma(z) = \begin{cases} a_+ z\,, & z \geq 0\,, \\ a_- z\,, & z < 0\,, \end{cases} \tag{∞.39}$$

with $a_+ \neq -a_-$, is that they are not smooth: their first derivative is a step function centered at the origin, and their second derivative is a Dirac delta function, (2.32),

$$\sigma'(z) = \begin{cases} a_+\,, & z \geq 0\,, \\ a_-\,, & z < 0\,, \end{cases}\,, \qquad \sigma''(z) = (a_+ - a_-)\delta(z)\,, \tag{∞.40}$$

and higher derivatives will involve derivatives of the Dirac delta function. Inspecting again the single-input ddNTK recursions (∞.14)–(∞.17), the kink in these activation functions and the presence of Gaussian expectations with up to three derivatives of $\sigma(z)$ should scare you, especially if you heeded our warning at the end of §5.5. In fact, if you formally try to evaluate some of these expectations, particularly $\langle\sigma''\sigma''\rangle_K$ and others related to it via integration by parts, you'll find that they want to blow up, even if you

[2] If we relaxed the restriction for the activation function to be odd, we'd find the same scalings for $R^{(\ell)}$, $S^{(\ell)}$, and $T^{(\ell)}$ – though with different coefficients – and we'd find that $U^{(\ell)}$ was up by a factor of ℓ, but still subleading overall.

use all the *magic tricks* from physics that you might have at your disposal for trying to make sense of divergent integrals.[3]

The divergence of these Gaussian correlators is actually telling us that there is something very wrong with our expansions for the updates to the preactivations (∞.4), the NTK (∞.6), and the dNTK (∞.9). In particular, the Taylor expansion in the global learning rate η breaks down for these non-smooth activation functions and doesn't accurately describe how a network is updated. As a result, our approach for solving the finite-width dynamics will not work for nonlinear scale-invariant activation functions.

To understand why, let's consider an extremely simple model of a network, a single neuron with a bias:

$$z(x) = \sigma(x + b) \,. \tag{∞.41}$$

Here, the input x and the output z are both scalars, and for the activation function $\sigma(z)$ we'll pick the `ReLU`, with $a_+ = 1$ and $a_- = 0$. Accordingly, for a particular input x such that $x + b > 0$, the activation fires, and the output is $z(x) = x + b$; for a particular input x' such that $x' + b < 0$, the activation doesn't fire, and the output is $z(x') = 0$.

Now, let's consider a gradient-descent update to the parameters with a training example $x > -b$ such that the activation fires. Then, the bias updates as

$$đb = -\eta \frac{dz}{db} \epsilon = -\eta\epsilon = O(\eta) \ , \tag{∞.42}$$

where ϵ is the error factor of the loss, depending on the true output y. Now, if $x+b+đb > 0$, then the change in the output is

$$đz = -\eta\epsilon = O(\eta) \ , \tag{∞.43}$$

and would be perfectly described by our Taylor expansion (∞.4). However, if the training error is large enough such that $x + b + đb = x + b - \eta\epsilon < 0$, then the activation turns off and

$$đz = -x - b = O(1) \ . \tag{∞.44}$$

Importantly, this update is independent of our expansion parameter η, and a Taylor expansion in η cannot detect this discontinuity at $\eta = (x + b)/\epsilon$.

Thus, for the `ReLU`, any time an activation crosses its firing threshold it can contribute to the gradient-descent update in an η-independent way. Empirically, if you try to measure the NTK update for a deep MLP consisting of `ReLU` activation functions, you don't find anything consistent with the expansion (∞.6). Instead, for the appropriately normalized quantity, you'll find an $\sim 1/\sqrt{n}$ scaling with width and a linear scaling with depth ℓ, in contrast to the ℓ/n scaling expected from our perturbative formalism.[4]

[3] We were somewhat lucky in §11.3.1 when analyzing the dNTK: all the higher-derivative Gaussian integrals that we needed simplified via integration by parts and gave finite – in fact, vanishing – answers, cf. (11.67) and (11.69).

[4] It's temping to think that this $\sim 1/\sqrt{n}$ scaling arises from accumulating the probabilities that one of the Ln total hidden-layer $n \gg 1$ activations experiences an η-independent $O(1)$ change after the gradient-descent step.

From this we reach the unfortunate conclusion that we'll have to give up on describing the finite-width dynamics of nonlinear scale-invariant activation functions using these methods.

Note that everything we have discussed for nonlinear scale-invariant activation functions with respect to criticality and the infinite-width training dynamics is perfectly fine and experimentally validated, and the presence of a nonzero dNTK at finite width is still indicative of a dynamical NTK and representation learning at finite width. The problem we just described only affects the analysis we're going to perform in the following section for the training dynamics of finite-width networks, and the breakdown of the Taylor expansion just means that we will be unable to give a quantitative picture of representation learning for these non-smooth activation functions.[5] So, if you do want to understand this, you'll probably need an entirely new approach.

This leaves us one activation function in the entire scale-invariant universality class: the `linear` activation function used for deep linear networks. Tuning to criticality, $C_b = 0$ and $C_W = 1$, which fixes $\chi = 1$, and choosing layer-independent learning rates (9.94) as

$$\lambda_b^{(\ell)} = \frac{\widetilde{\lambda}_b}{L}\,, \qquad \lambda_W^{(\ell)} = \frac{\widetilde{\lambda}_W}{L}\,, \tag{∞.45}$$

we can solve the single-input ddNTK recursions (∞.14)–(∞.17). Note that for every term in the U-recursion, (∞.17), and for every term but one in the R-recursion, (∞.14), the Gaussian expectations of activations involve second derivatives or third derivatives, which vanish for a `linear` function. For the one term in the R-recursion that does not vanish, it is also multiplied by $P^{(\ell)}$, which vanishes for all scale-invariant activations, cf. (11.74). This means that

$$R^{(\ell)} = 0\,, \qquad U^{(\ell)} = 0\,, \tag{∞.46}$$

and in particular that the first ddNTK, $\widehat{\mathrm{dd_I}H}$, does not contribute to the deep linear network's dynamics at $O(1/n)$ since it's entirely determined by $R^{(\ell)}$. However, for the S-recursion, (∞.15), and the T-recursion, (∞.16), we find something nontrivial: these recursions can be exactly solved as

$$S^{(\ell)} = \frac{\ell^2(\ell^2-1)}{12L^3}(\widetilde{\lambda}_b + \widetilde{\lambda}_W K^\star)^2 \widetilde{\lambda}_W\,, \tag{∞.47}$$

$$T^{(\ell)} = \frac{\ell^2(\ell^2-1)}{12L^3}(\widetilde{\lambda}_b + \widetilde{\lambda}_W K^\star)\widetilde{\lambda}_W^2 K^\star\,, \tag{∞.48}$$

from which we see that the critical exponents are $p_S = p_T = -4$. Finally, the dimen-

[5] However, our analysis applies to any of the smoothed versions of the `ReLU` that permit a type of criticality, cf. our criticality discussion of the `SWISH` and `GELU` in §5.3.4. In particular, the training dynamics we'll work out in the next section describe these networks. These dynamical solutions, in conjunction with the output-layer solutions to all their associated recursions, will accurately characterize such fully-trained `ReLU`-like networks in practice.

sionless ratios are

$$\frac{S^{(\ell)}K^{(\ell)}}{n\left(\Theta^{(\ell)}\right)^3} = \frac{1}{12}\left[\frac{\widetilde{\lambda}_W K^\star}{\widetilde{\lambda}_b + \widetilde{\lambda}_W K^\star}\right]\frac{\ell}{n} + \ldots\,, \tag{∞.49}$$

$$\frac{T^{(\ell)}K^{(\ell)}}{n\left(\Theta^{(\ell)}\right)^3} = \frac{1}{12}\left[\frac{\widetilde{\lambda}_W K^\star}{\widetilde{\lambda}_b + \widetilde{\lambda}_W K^\star}\right]^2\frac{\ell}{n} + \ldots\,, \tag{∞.50}$$

and we see that these scale according to our leading effective theory cutoff as

$$p_S + p_0 - 3p_\Theta = -1\,, \qquad p_T + p_0 - 3p_\Theta = -1\,. \tag{∞.51}$$

In conclusion, we see that the second ddNTK, $\widehat{\mathrm{dd}_{\mathrm{II}}H}$, contributes via $S^{(\ell)}$ and $T^{(\ell)}$ to the dynamics of deep linear networks at leading order.

∞.2 Training at Finite Width

Now that we understand the joint statistics of the preactivations, the NTK, the dNTK, and the ddNTKs, we have nearly all the tools we need in order to evaluate the distribution of *fully-trained* networks at finite width and nonzero depth. To see why, recall our finite-width expansion of the network output evolution (∞.4)

$$\begin{aligned} z^{(L)}_{i;\delta}(t=1) = z^{(L)}_{i;\delta} &- \eta\sum_{j,\tilde{\alpha}}\widehat{H}^{(L)}_{ij;\delta\tilde{\alpha}}\epsilon^{(L)}_{j;\tilde{\alpha}} + \frac{\eta^2}{2}\sum_{j_1,j_2,\tilde{\alpha}_1,\tilde{\alpha}_2}\widehat{\mathrm{d}H}^{(L)}_{ij_1j_2;\delta\tilde{\alpha}_1\tilde{\alpha}_2}\epsilon^{(L)}_{j_1;\tilde{\alpha}_1}\epsilon^{(L)}_{j_2;\tilde{\alpha}_2} \\ &- \frac{\eta^3}{6}\sum_{j_1,j_2,j_3,\tilde{\alpha}_1,\tilde{\alpha}_2,\tilde{\alpha}_3}\widehat{\mathrm{dd}_{\mathrm{I}}H}^{(L)}_{ij_1j_2j_3;\delta\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3}\epsilon^{(\ell)}_{j_1;\tilde{\alpha}_1}\epsilon^{(\ell)}_{j_2;\tilde{\alpha}_2}\epsilon^{(\ell)}_{j_3;\tilde{\alpha}_3} + O\left(\frac{1}{n^2}\right)\,. \end{aligned} \tag{∞.52}$$

Importantly, all the quantities on the right-hand side of the network update (∞.52) – $z^{(L)}_{i;\delta}$, $\widehat{H}^{(L)}_{ij;\delta\tilde{\alpha}}$, $\epsilon^{(L)}_{j;\tilde{\alpha}}$, $\widehat{\mathrm{d}H}^{(L)}_{ij_1j_2;\delta\tilde{\alpha}_1\tilde{\alpha}_2}$, and $\widehat{\mathrm{dd}_{\mathrm{I}}H}^{(\ell)}_{ij_1j_2j_3;\delta\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3}$ – are evaluated at initialization and thus are determined completely by the statistics of the joint preactivation-NTK-dNTK-ddNTKs distribution,

$$p\left(z^{(L)},\, \widehat{H}^{(L)},\, \widehat{\mathrm{d}H}^{(L)},\, \widehat{\mathrm{dd}_{\mathrm{I}}H}^{(L)},\, \widehat{\mathrm{dd}_{\mathrm{II}}H}^{(L)}\Big|\mathcal{D}\right)\,, \tag{∞.53}$$

that we spent the majority of this book evaluating in detail. Accordingly – just as we did before at infinite width (§10) – we could use the joint distribution (∞.53) to compute the statistics of the fully-trained network outputs after the update (∞.52), *if* we tuned a single step of gradient descent for each realization of a network so that we landed on the minimum of the training loss $z^{(L)}_{i;\tilde{\alpha}}(t=1) = y_{i;\tilde{\alpha}}$.

More generally, *if* we trained the network with T steps of gradient descent such that

$$z^{(L)}_{i;\tilde{\alpha}}(t=T) = y_{i;\tilde{\alpha}}\,, \qquad \text{for all } \tilde{\alpha}\in\mathcal{A}\,, \tag{∞.54}$$

and if we determined how to express the output of such a fully-trained network for general inputs $\delta \in \mathcal{D}$ as a functional of our statistical variables at initialization,

$$z^{(L)}_{i;\delta}(T) \equiv \left[z^{(L)}_{i;\delta}(t=T)\right]\left(z^{(L)},\, \widehat{H}^{(L)},\, \widehat{\mathrm{d}H}^{(L)},\, \widehat{\mathrm{dd_I}H}^{(L)},\, \widehat{\mathrm{dd_{II}}H}^{(L)}\right) , \tag{∞.55}$$

then we could give an analytical expression for the distribution of *fully-trained network outputs*:

$$p\Big(z^{(L)}(T)\Big) \ . \tag{∞.56}$$

The equation (∞.54) is our *fully-trained condition*, and the distribution (∞.56) completely describes the ensemble of finite-width networks at the end of training. The theoretical understanding of this distribution is exactly the goal we set for ourselves at the beginning of this book in §0. The only thing left for us to work out is the functional (∞.55); to do so, we first need to figure out what kind of steps to take in order to fully train these finite-width networks.

Now, recall from §10.2.2 that in the infinite-width limit the fully-trained network solution (∞.55) had an *algorithm independence*: the distribution at the end of training didn't depend on the details of the optimization algorithm, and thus we could perform our theoretical analysis with any algorithm we wanted. In contrast, at finite width the fully-trained solution (∞.55) will have an **algorithm dependence**: different fully-trained solutions will make different test-set predictions depending on the details of the particular optimization algorithm used to train the network, even when holding fixed the statistical variables at initialization and their associated initialization and training hyperparameters. Encouragingly, the form of the solutions will nonetheless take a universal form, with the non-universal details of the particular training algorithm captured by six projective tensors: cf. (∞.154). Thus, we will be able to very generally study the distribution of fully-trained networks at finite width by working out such solutions.

With that in mind, in this section we'll present fully-trained solutions for two different optimization algorithms. First, in §∞.2.1 we'll take *two* Newton-like steps in order to satisfy our fully-trained condition (∞.54). While practically infeasible for large training sets, this training algorithm is rich in pedagogical value, emphasizing the way in which a finite-width network needs to adapt its representation to the training data in order to minimize its training error. Then, in §∞.2.2 we'll analytically solve the dynamics of the vanilla gradient descent at order $1/n$ and obtain a slightly different ensemble of fully-trained finite-width networks. This algorithm is not only practically implementable, but also quite often used to optimize real neural networks, and our corresponding solution is an actual theoretical description of such fully-trained networks. Together, these solutions will help us understand the ways in which the details of the optimization algorithm can affect the corresponding fully-trained solution. Finally, in §∞.2.3 we'll be able to generally analyze the predictions of these different fully-trained networks on novel examples from the test set.

Throughout this section, we will declutter the notation a bit by dropping the layer indices, since to understand training we only need to focus on the network output at layer $\ell = L$.

An Infinite-Width Giant Leap at Finite Width

Before we begin, let's first review the giant leap that we took in §10.2 at infinite width. From the finer-grained perspective of finite width, we'll see that our leap actually missed the minimum, exhibiting training errors of order $1/n$. However, our new eyes on this leap will be instructive, as they will help us see how we can correct for these errors and reduce the finite-width training error even further.

Recall from §10.2 that, in order to fully train an infinite-width network in a single step, we needed to make a *second-order update* of the form

$$d\!\!\!{}^-\theta_\mu = -\sum_{\nu,\tilde{\alpha}_1,\tilde{\alpha}_2,i} \eta\lambda_{\mu\nu}\kappa^{\tilde{\alpha}_1\tilde{\alpha}_2}\frac{dz_{i;\tilde{\alpha}_1}}{d\theta_\nu}\left(z_{i;\tilde{\alpha}_2}-y_{i;\tilde{\alpha}_2}\right), \tag{∞.57}$$

which we interpreted either *(i)* as a *generalized training algorithm* (10.19) optimizing the standard MSE loss (10.5), or *(ii)* as a standard (tensorial) gradient-descent step (7.11) optimizing a *generalized MSE loss* (10.22). Here, $\kappa^{\tilde{\alpha}_1\tilde{\alpha}_2}$ was called the *Newton tensor*, and – in the first understanding of (∞.57) – could be interpreted as allowing us to take anisotropic steps in training sample space.

With this type of parameter update, a finite-width network output will evolves as

$$\begin{aligned}
&z_{i;\delta}(t=1) \qquad\qquad (\infty.58)\\
=&z_{i;\delta}-\eta\sum_{\tilde{\alpha}_1,\tilde{\alpha}_2}H_{\delta\tilde{\alpha}_1}\kappa^{\tilde{\alpha}_1\tilde{\alpha}_2}\left(z_{i;\tilde{\alpha}_2}-y_{i;\tilde{\alpha}_2}\right)-\eta\sum_{j,\tilde{\alpha}_1,\tilde{\alpha}_2}\widehat{\Delta H}_{ij;\delta\tilde{\alpha}_1}\kappa^{\tilde{\alpha}_1\tilde{\alpha}_2}\left(z_{j;\tilde{\alpha}_2}-y_{j;\tilde{\alpha}_2}\right)\\
&+\frac{\eta^2}{2}\sum_{j_1,j_2,\tilde{\alpha}_1,\tilde{\alpha}_2,\tilde{\alpha}_3,\tilde{\alpha}_4}\widehat{\mathrm{d}H}_{ij_1j_2;\delta\tilde{\alpha}_1\tilde{\alpha}_2}\kappa^{\tilde{\alpha}_1\tilde{\alpha}_3}\kappa^{\tilde{\alpha}_2\tilde{\alpha}_4}\left(z_{j_1;\tilde{\alpha}_3}-y_{j_1;\tilde{\alpha}_3}\right)\left(z_{j_2;\tilde{\alpha}_4}-y_{j_2;\tilde{\alpha}_4}\right)\\
&-\frac{\eta^3}{6}\sum_{j_1,j_2,j_3,\tilde{\alpha}_1,\ldots,\tilde{\alpha}_6}\widehat{\mathrm{dd}_{\mathrm{I}}H}_{ij_1j_2j_3;\delta\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3}\kappa^{\tilde{\alpha}_1\tilde{\alpha}_4}\kappa^{\tilde{\alpha}_2\tilde{\alpha}_5}\kappa^{\tilde{\alpha}_3\tilde{\alpha}_6}\\
&\qquad\times\left(z_{j_1;\tilde{\alpha}_4}-y_{j_1;\tilde{\alpha}_4}\right)\left(z_{j_2;\tilde{\alpha}_5}-y_{j_2;\tilde{\alpha}_5}\right)\left(z_{j_3;\tilde{\alpha}_6}-y_{j_3;\tilde{\alpha}_6}\right)+O\!\left(\frac{1}{n^2}\right),
\end{aligned}$$

where here we've made our usual decomposition of the NTK into a mean and fluctuation

$$\widehat{H}_{ij;\delta\tilde{\alpha}} \equiv \delta_{ij}H_{\delta\tilde{\alpha}}+\widehat{\Delta H}_{ij;\delta\tilde{\alpha}}\,. \tag{∞.59}$$

In terms of such an update, our fully-trained condition (∞.54) after a single step $T=1$,

$$z_{i;\tilde{\alpha}}(t=1)=y_{i;\tilde{\alpha}}\,, \qquad \text{for all } \tilde{\alpha}\in\mathcal{A}\,, \tag{∞.60}$$

can be written as

$$
\begin{aligned}
& z_{i;\tilde{\alpha}} - y_{i;\tilde{\alpha}} \qquad\qquad (\infty.61)\\
=& \eta \sum_{\tilde{\alpha}_1,\tilde{\alpha}_2} H_{\tilde{\alpha}\tilde{\alpha}_1} \kappa^{\tilde{\alpha}_1\tilde{\alpha}_2} \left(z_{i;\tilde{\alpha}_2} - y_{i;\tilde{\alpha}_2}\right) + \eta \sum_{j,\tilde{\alpha}_1,\tilde{\alpha}_2} \widehat{\Delta H}_{ij;\tilde{\alpha}\tilde{\alpha}_1} \kappa^{\tilde{\alpha}_1\tilde{\alpha}_2} \left(z_{j;\tilde{\alpha}_2} - y_{j;\tilde{\alpha}_2}\right) \\
& - \frac{\eta^2}{2} \sum_{j_1,j_2,\tilde{\alpha}_1,\tilde{\alpha}_2,\tilde{\alpha}_3,\tilde{\alpha}_4} \widehat{\mathrm{d}H}_{ij_1j_2;\tilde{\alpha}\tilde{\alpha}_1\tilde{\alpha}_2} \kappa^{\tilde{\alpha}_1\tilde{\alpha}_3} \kappa^{\tilde{\alpha}_2\tilde{\alpha}_4} \left(z_{j_1;\tilde{\alpha}_3} - y_{j_1;\tilde{\alpha}_3}\right) \left(z_{j_2;\tilde{\alpha}_4} - y_{j_2;\tilde{\alpha}_4}\right) \\
& + \frac{\eta^3}{6} \sum_{j_1,j_2,j_3,\tilde{\alpha}_1,\ldots,\tilde{\alpha}_6} \widehat{\mathrm{dd_I}H}_{ij_1j_2j_3;\tilde{\alpha}\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3} \kappa^{\tilde{\alpha}_1\tilde{\alpha}_4} \kappa^{\tilde{\alpha}_2\tilde{\alpha}_5} \kappa^{\tilde{\alpha}_3\tilde{\alpha}_6} \\
& \qquad\qquad \times \left(z_{j_1;\tilde{\alpha}_4} - y_{j_1;\tilde{\alpha}_4}\right) \left(z_{j_2;\tilde{\alpha}_5} - y_{j_2;\tilde{\alpha}_5}\right) \left(z_{j_3;\tilde{\alpha}_6} - y_{j_3;\tilde{\alpha}_6}\right) + O\left(\frac{1}{n^2}\right) .
\end{aligned}
$$

This new giant-leap condition (∞.61) perhaps seems a little daunting, and further it's not obvious that there's any particular choice of Newton tensor $\kappa^{\tilde{\alpha}_1\tilde{\alpha}_2}$ that can land us on the minimum. Nonetheless, we do expect that our infinite-width solution should be near the true finite-width solution, up to errors of order $O(1/n)$.[6]

With that in mind, as a first step let's try our infinite-width giant leap (10.30) and see where we land. This infinite-width giant leap had an interpretation as *Newton's method* and was given by the second-order update (∞.57), with a particular choice of the product of the global learning rate and the Newton tensor,

$$
\eta \kappa^{\tilde{\alpha}_1\tilde{\alpha}_2} = \widetilde{H}^{\tilde{\alpha}_1\tilde{\alpha}_2} , \qquad (\infty.62)
$$

where the *inverse* NTK mean submatrix $\widetilde{H}^{\tilde{\alpha}_1\tilde{\alpha}_2}$ was defined implicitly via

$$
\sum_{\tilde{\alpha}_2 \in \mathcal{A}} \widetilde{H}^{\tilde{\alpha}_1\tilde{\alpha}_2} \widetilde{H}_{\tilde{\alpha}_2\tilde{\alpha}_3} = \delta^{\tilde{\alpha}_1}_{\ \tilde{\alpha}_3} . \qquad (\infty.63)
$$

As always, the tilde on $\widetilde{H}_{\tilde{\alpha}_1\tilde{\alpha}_2}$ emphasizes that it's an $N_{\mathcal{A}} \times N_{\mathcal{A}}$-dimensional submatrix of the NTK mean evaluated on pairs of training inputs *only*. By now this distinction should be familiar enough that we will stop belaboring it.

More importantly, here we've used the inverse of the full NTK mean $\widetilde{H}_{\tilde{\alpha}_1\tilde{\alpha}_2}$ rather than the infinite-width frozen NTK $\widetilde{\Theta}_{\tilde{\alpha}_1\tilde{\alpha}_2}$. To explain why, let us recall from (9.3) that the NTK mean receives a series of corrections at each order in the $1/n$ expansion, of which the leading-order piece is the frozen NTK (9.4). Since we're now working at order $1/n$, we should in particular take into account the next-to-leading-order (NLO) $1/n$ correction to the NTK mean $H^{\{1\}}_{\tilde{\alpha}_1\tilde{\alpha}_2}$ by working with $\widetilde{H}_{\tilde{\alpha}_1\tilde{\alpha}_2}$ instead of $\widetilde{\Theta}_{\tilde{\alpha}_1\tilde{\alpha}_2}$.[7]

[6] For deep networks, we know that technically the corrections will be of order $O(L/n)$, corresponding to the cutoff scale of our effective theory description.

[7] While we won't show it explicitly, we expect that this *NLO NTK mean*, $H^{\{1\}(\ell)}_{\tilde{\alpha}_1\tilde{\alpha}_2}$, will have a solution that scales like $O(1/n)$ as compared to the frozen NTK; this would be analogous to what we found for the NLO metric $G^{\{1\}(\ell)}_{\tilde{\alpha}_1\tilde{\alpha}_2}$, cf. the discussion in §5.4 after (5.143). In particular, if we make a $1/n$ expansion for our training hyperparameters $\lambda_b^{(\ell)}$ and $\lambda_W^{(\ell)}$ as we did for our initialization hyperparameters in (5.138)

Substituting our infinite-width giant-leap Newton tensor (∞.62) into our giant-leap condition (∞.61) at finite width and rearranging, we get

$$\begin{aligned} 0 = &\sum_{j,\tilde{\alpha}_1,\tilde{\alpha}_2} \widehat{\Delta H}_{ij;\tilde{\alpha}\tilde{\alpha}_1} \widetilde{H}^{\tilde{\alpha}_1\tilde{\alpha}_2} (z_{j;\tilde{\alpha}_1} - y_{j;\tilde{\alpha}_1}) \\ &- \frac{1}{2} \sum_{\substack{j_1,j_2,\\ \tilde{\alpha}_1,\ldots,\tilde{\alpha}_4}} \widehat{\mathrm{d}H}_{ij_1j_2;\tilde{\alpha}\tilde{\alpha}_1\tilde{\alpha}_2} \widetilde{H}^{\tilde{\alpha}_1\tilde{\alpha}_3} \widetilde{H}^{\tilde{\alpha}_2\tilde{\alpha}_4} (z_{j_1;\tilde{\alpha}_3} - y_{j_1;\tilde{\alpha}_3})(z_{j_2;\tilde{\alpha}_4} - y_{j_2;\tilde{\alpha}_4}) \\ &+ \frac{1}{6} \sum_{\substack{j_1,j_2,j_3,\\ \tilde{\alpha}_1,\ldots,\tilde{\alpha}_6}} \widehat{\mathrm{dd_I}H}_{ij_1j_2j_3;\tilde{\alpha}\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3} \widetilde{H}^{\tilde{\alpha}_1\tilde{\alpha}_4} \widetilde{H}^{\tilde{\alpha}_2\tilde{\alpha}_5} \widetilde{H}^{\tilde{\alpha}_3\tilde{\alpha}_6} \\ &\qquad\qquad \times (z_{j_1;\tilde{\alpha}_4} - y_{j_1;\tilde{\alpha}_4}) (z_{j_2;\tilde{\alpha}_5} - y_{j_2;\tilde{\alpha}_5}) (z_{j_3;\tilde{\alpha}_6} - y_{j_3;\tilde{\alpha}_6}) + O\Big(\frac{1}{n^2}\Big) \,. \end{aligned} \tag{∞.64}$$

Thus, we actually missed the minimum: for the network to be fully trained, the right-hand side of (∞.64) should have vanished, while here it is clearly nonzero in general. Taking a step back, it's clear now that what we actually found at infinite width in §10.2 was

$$z_{i;\tilde{\alpha}}(t=1) - y_{i;\tilde{\alpha}} = O\Big(\frac{1}{n}\Big) \,. \tag{∞.65}$$

In other words, our networks were fully trained only at leading order, and (∞.64) gives an explicit expression for the $1/n$ correction.

Disentangling a little further, there are two such corrections at order $1/n$: the first correction – the first term on the right-hand side of (∞.64) – arises from the instantiation-to-instantiation fluctuations of the NTK across different realizations of the biases and weights at initialization; the second correction – comprised of the second and third terms on the right-hand side of (∞.64) – arises from nonlinear changes in the output as we take our step. In particular, this second correction is a *bona fide* manifestation of representation learning at finite width, accounting for the fact that the network's *effective features* are evolving. If we properly account for these two types of the $1/n$ corrections, we should be able to attain a training error of order $1/n^2$:

$$z_{i;\tilde{\alpha}}(T) - y_{i;\tilde{\alpha}} = O\Big(\frac{1}{n^2}\Big) \,. \tag{∞.66}$$

That is, we should be able to improve our effective theory of fully-trained networks, quantitatively by another multiplicative factor of $1/n$, and qualitatively by properly including representation learning into such an effective description.

---

and (5.139), then we will have extra freedom in the subleading hyperparameters $\lambda_b^{(\ell)\{1\}}$ and $\lambda_W^{(\ell)\{1\}}$ to eliminate the growing-in-ℓ contribution to $H_{\tilde{\alpha}_1\tilde{\alpha}_2}^{\{1\}(\ell)}$. Overall, this will make the NLO correction to the NTK mean scale as $O(1/n)$, subleading to the leading finite-width effects which scale as $O(L/n)$: in the language of RG flow, the NLO NTK mean is *marginal*. In practice, such a contribution is negligible and can thus be neglected for networks of any real depth.

Our first approach (§∞.2.1) to attain such effectively-zero training error, (∞.66), is to continue to engineer theoretical giant leaps so as to account for both the instantiation-to-instantiation fluctuations of the NTK and the effect of the dNTK and the first ddNTK.

Another approach (§∞.2.2) is to simply use the vanilla tensorial gradient descent algorithm as we do in practice; in that case, we will have to not only account for the dynamics of the network output, but also account for the dynamics of the NTK and dNTK. After doing so, we will see that we can iteratively decrease the training loss to zero after many many such steps.

∞.2.1 A Small Step Following a Giant Leap

Here we'll train our networks with *two* second-order updates. For the first update, a giant leap, we'll need to further generalize our theoretical optimization algorithm in order to properly account for the instantiation-to-instantiation fluctuations of the NTK. In the second update, a small step, we'll be able to account for the $1/n$ change in representation due to the nonzero dNTK and ddNTK, ultimately landing on the minimum of the loss as (∞.66). In particular, we can think of these updates as loosely corresponding to distinct phases of training that arise when implementing gradient-based training of neural networks in practice.

First Update: One Final Generalization of Gradient Decent

Please flip back to take a closer look at our unsatisfied condition for fully training our networks (∞.64). Right away, you should notice a serious problem in satisfying this constraint: the NTK fluctuation, $\widehat{\Delta H}_{ij;\tilde{\alpha}\tilde{\alpha}_1}$, the dNTK, $\widehat{\mathrm{d}H}_{ij_1j_2;\tilde{\alpha}\tilde{\alpha}_1\tilde{\alpha}_2}$, and the ddNTK, $\widehat{\mathrm{dd}_{\mathrm{I}}H}_{ij_1j_2j_3;\tilde{\alpha}\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3}$, all mix different output components together in an update; i.e. the j-th component of the prediction error $z_{j;\tilde{\alpha}} - y_{j;\tilde{\alpha}}$ at initialization affects the i-th component of the output $z_{i;\delta}$ after the update, even for $i \neq j$. This stands in contrast to what we found for an infinite-width update in §10.1.1, where there was no such mixing or *wiring* of output components. Meanwhile, we did see a similar wiring effect for Bayesian inference at finite width as we discussed in depth in §6.4.2.

While such wiring at finite width is fantastic from a practitioner's standpoint, it makes our theoretical work slightly more complicated. In particular, in order to satisfy the training constraint (∞.64), we will need to further generalize our second-order update (∞.57). Following in the footsteps of our previous two generalizations, (7.11) and (10.18), let's make one final generalization

$$\eta \to \eta\lambda_{\mu\nu} \to \eta\lambda_{\mu\nu}\kappa^{\tilde{\alpha}_1\tilde{\alpha}_2} \to \eta\lambda_{\mu\nu}\kappa_{ij}^{\tilde{\alpha}_1\tilde{\alpha}_2}\,, \tag{∞.67}$$

with the final form of our theoretical update given by

$$đ\theta_\mu = -\sum_{\nu,\tilde{\alpha}_1,\tilde{\alpha}_2,i,j} \eta\lambda_{\mu\nu}\kappa_{ij}^{\tilde{\alpha}_1\tilde{\alpha}_2}\frac{dz_{i;\tilde{\alpha}_1}}{d\theta_\nu}\epsilon_{j;\tilde{\alpha}_2}\,. \tag{∞.68}$$

Here, we also introduced a further *generalized* Newton tensor $\kappa_{ij}^{\tilde{\alpha}_1\tilde{\alpha}_2}$ with output-component indices. This flexibility will allow us to resolve the mixing of the output components in the residual training error from our infinite-width leap (∞.64).[8]

Note importantly that the μ, ν indices of $\lambda_{\mu\nu}$ are very different from the i, j indices of $\kappa_{ij}^{\tilde{\alpha}_1\tilde{\alpha}_2}$: the former each runs over all P parameters, while the latter each only runs over the n_L components of the network output. In particular, while learning-rate tensor $\lambda_{\mu\nu}$ lets us control how the gradient of the ν-th parameter affects the update to the μ-th parameter, the i, j indices of the generalized Newton tensor $\kappa_{ij}^{\tilde{\alpha}_1\tilde{\alpha}_2}$ instead control how the network's *features* (10.138) $dz_{i;\tilde{\alpha}_1}/d\theta_\nu$ are combined with the error factor $\epsilon_{j;\tilde{\alpha}_2}$ in order to make the update.[9] Allowing for a $\kappa_{ij}^{\tilde{\alpha}_1\tilde{\alpha}_2}$ with nonzero off-diagonal components in the i, j indices, we can precisely tune the *wiring* or mixing of output components that occurs in a particular update.

Finally, plugging our final-form second-order update (∞.68) back into the $1/n$ expansion of the network update (∞.4) and using the NTK, dNTK, and first ddNTK definitions, we can see how the network output changes after making an update with this new optimization algorithm:

$$
\begin{aligned}
& z_{i;\delta}(t=1) \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (\infty.70) \\
=& z_{i;\delta} - \eta \sum_{j,\tilde{\alpha}_1,\tilde{\alpha}_2} H_{\delta\tilde{\alpha}_1} \kappa_{ij}^{\tilde{\alpha}_1\tilde{\alpha}_2} \left(z_{j;\tilde{\alpha}_2} - y_{j;\tilde{\alpha}_2}\right) - \eta \sum_{j,k,\tilde{\alpha}_1,\tilde{\alpha}_2} \widehat{\Delta H}_{ij;\delta\tilde{\alpha}_1} \kappa_{jk}^{\tilde{\alpha}_1\tilde{\alpha}_2} \left(z_{k;\tilde{\alpha}_2} - y_{k;\tilde{\alpha}_2}\right) \\
& + \frac{\eta^2}{2} \sum_{\substack{j_1,j_2,k_1,k_2, \\ \tilde{\alpha}_1,\tilde{\alpha}_2,\tilde{\alpha}_3,\tilde{\alpha}_4}} \widehat{\mathrm{d}H}_{ij_1j_2;\delta\tilde{\alpha}_1\tilde{\alpha}_2} \kappa_{j_1k_1}^{\tilde{\alpha}_1\tilde{\alpha}_3} \kappa_{j_2k_2}^{\tilde{\alpha}_2\tilde{\alpha}_4} \left(z_{k_1;\tilde{\alpha}_3} - y_{k_1;\tilde{\alpha}_3}\right) \left(z_{k_2;\tilde{\alpha}_4} - y_{k_2;\tilde{\alpha}_4}\right) \\
& - \frac{\eta^3}{6} \sum_{\substack{j_1,j_2,j_3,k_1,k_2,k_3 \\ \tilde{\alpha}_1,\tilde{\alpha}_2,\tilde{\alpha}_3,\tilde{\alpha}_4,\tilde{\alpha}_5,\tilde{\alpha}_6}} \widehat{\mathrm{dd_I}H}_{ij_1j_2j_3;\delta\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3} \kappa_{j_1k_1}^{\tilde{\alpha}_1\tilde{\alpha}_4} \kappa_{j_2k_2}^{\tilde{\alpha}_2\tilde{\alpha}_5} \kappa_{j_3k_3}^{\tilde{\alpha}_3\tilde{\alpha}_6} \\
& \qquad\qquad \times \left(z_{k_1;\tilde{\alpha}_4} - y_{k_1;\tilde{\alpha}_4}\right) \left(z_{k_2;\tilde{\alpha}_5} - y_{k_2;\tilde{\alpha}_5}\right) \left(z_{k_3;\tilde{\alpha}_6} - y_{k_3;\tilde{\alpha}_6}\right) + O\left(\frac{1}{n^2}\right) .
\end{aligned}
$$

Here, we've again used the standard MSE loss, for which the error factor is given by

[8] Similarly to the generalized MSE loss (10.22) discussed in §10.2.1, we can alternatively think of this further-generalized second-order update (∞.68) optimizing the standard MSE loss as instead arising from a standard first-order (tensorial) gradient descent update (7.11) on a further-generalized MSE loss,

$$
\mathcal{L}_{\mathcal{A},\,\kappa}(\theta) \equiv \frac{1}{2} \sum_{i_1,i_2=1}^{n_L} \sum_{\tilde{\alpha}_1,\tilde{\alpha}_2 \in \mathcal{A}} \kappa_{i_1i_2}^{\tilde{\alpha}_1\tilde{\alpha}_2} \left(z_{i_1;\tilde{\alpha}_1} - y_{i_1;\tilde{\alpha}_1}\right) \left(z_{i_2;\tilde{\alpha}_2} - y_{i_2;\tilde{\alpha}_2}\right) , \qquad (\infty.69)
$$

as long as the Newton tensor $\kappa_{i_1i_2}^{\tilde{\alpha}_1\tilde{\alpha}_2}$ is *also* assumed to be symmetric under the exchange of paired indices $(i_1, \tilde{\alpha}_1) \leftrightarrow (i_2, \tilde{\alpha}_2)$. This symmetry will in fact be present for both our first update, the giant leap, and our second update, the small step. With this interpretation (∞.69), our generalized Newton tensor $\kappa_{ij}^{\tilde{\alpha}_1\tilde{\alpha}_2}$ acts as a metric on sample space, through its sample indices $\tilde{\alpha}_1, \tilde{\alpha}_2$, and on output-component space, through its L-th layer neural indices i, j.

[9] Note that when we developed our interpretation of an infinite-width network as a linear model in §10.4, we made a similar distinction between parameter indices μ and output-component indices i in (10.138) when defining the random feature functions $\widehat{\phi}_{i,\mu}(x)$. We also discussed how *wiring* can be incorporated into a non-minimal model of representation learning in §11.4.3.

the residual training error $\epsilon_{j;\tilde{\alpha}} = z_{j;\tilde{\alpha}} - y_{j;\tilde{\alpha}}$. Now, we'll need to pick our generalized Newton tensor $\kappa_{ij}^{\tilde{\alpha}_1\tilde{\alpha}_2}$ judiciously in order to make a first update that fully accounts for instantiation-to-instantiation fluctuations of particular networks in our ensemble.

Taking inspiration from our $1/n$-failed infinite-width Newton's step (∞.62), let's take a similar-looking first step according to

$$\begin{aligned}\eta\kappa_{ij}^{\tilde{\alpha}_1\tilde{\alpha}_2} =& \left(\widehat{H}^{-1}\right)_{ij}^{\tilde{\alpha}_1\tilde{\alpha}_2} \\ =&\delta_{ij}\widetilde{H}^{\tilde{\alpha}_1\tilde{\alpha}_2} - \sum_{\tilde{\alpha}_3,\tilde{\alpha}_4\in\mathcal{A}} \widetilde{H}^{\tilde{\alpha}_1\tilde{\alpha}_3}\widehat{\Delta H}_{ij;\tilde{\alpha}_3\tilde{\alpha}_4}\widetilde{H}^{\tilde{\alpha}_4\tilde{\alpha}_2} \\ &+ \sum_{k=1}^{n_L}\sum_{\tilde{\alpha}_3,\ldots,\tilde{\alpha}_6\in\mathcal{A}} \widetilde{H}^{\tilde{\alpha}_1\tilde{\alpha}_3}\widehat{\Delta H}_{ik;\tilde{\alpha}_3\tilde{\alpha}_4}\widetilde{H}^{\tilde{\alpha}_4\tilde{\alpha}_5}\widehat{\Delta H}_{kj;\tilde{\alpha}_5\tilde{\alpha}_6}\widetilde{H}^{\tilde{\alpha}_6\tilde{\alpha}_2} + O\Big(\Delta^3\Big)\,.\end{aligned} \tag{∞.71}$$

Here, we've introduced the complete inverse of the stochastic NTK sub-tensor evaluated on the training set, satisfying

$$\sum_{j,\tilde{\alpha}_2}\left(\widehat{H}^{-1}\right)_{ij}^{\tilde{\alpha}_1\tilde{\alpha}_2}\widehat{H}_{jk;\tilde{\alpha}_2\tilde{\alpha}_3} = \delta_{ik}\delta^{\tilde{\alpha}_1}_{\tilde{\alpha}_3}\,, \tag{∞.72}$$

and in the last equality of (∞.71) we've used the Schwinger-Dyson equations (4.55), which is a physicist's way of saying that we expanded the inverse of the stochastic NTK around the NTK mean.[10] The main difference between this new giant leap (∞.71) and our previous infinite-width giant leap (∞.62) is that we're now taking into account the instantiation-to-instantiation fluctuations of the NTK across different realizations of the model parameters; in other words, we're implementing a *different* Newton step for each *particular* network with its associated NTK $\widehat{H}_{i_1i_2;\tilde{\alpha}_1\tilde{\alpha}_2}$. Accordingly, this step can be thought of as loosely corresponding to the first phase of training for such a particular network.

Taking this step, i.e. plugging this generalized Newton tensor (∞.71) into our update to the network output (∞.70), we find the training error decreases to

$$\begin{aligned}&z_{i;\tilde{\alpha}}(t=1) - y_{i;\tilde{\alpha}} \\ =&\frac{1}{2}\sum_{\substack{j_1,j_2,\\ \tilde{\alpha}_1,\ldots,\tilde{\alpha}_4}} \widehat{\mathrm{d}H}_{ij_1j_2;\tilde{\alpha}\tilde{\alpha}_1\tilde{\alpha}_2}\widetilde{H}^{\tilde{\alpha}_1\tilde{\alpha}_3}\widetilde{H}^{\tilde{\alpha}_2\tilde{\alpha}_4}(z_{j_1;\tilde{\alpha}_3}-y_{j_1;\tilde{\alpha}_3})(z_{j_2;\tilde{\alpha}_4}-y_{j_2;\tilde{\alpha}_4}) \\ &-\frac{1}{6}\sum_{\substack{j_1,j_2,j_3,\\ \tilde{\alpha}_1,\ldots\tilde{\alpha}_6}} \widehat{\mathrm{dd_I}H}_{ij_1j_2j_3;\delta\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3}\widetilde{H}^{\tilde{\alpha}_1\tilde{\alpha}_4}\widetilde{H}^{\tilde{\alpha}_2\tilde{\alpha}_5}\widetilde{H}^{\tilde{\alpha}_3\tilde{\alpha}_6} \\ &\qquad\qquad\times\left(z_{j_1;\tilde{\alpha}_4}-y_{j_1;\tilde{\alpha}_4}\right)\left(z_{j_2;\tilde{\alpha}_5}-y_{j_2;\tilde{\alpha}_5}\right)\left(z_{j_3;\tilde{\alpha}_6}-y_{j_3;\tilde{\alpha}_6}\right) + O\left(\frac{1}{n^2}\right)\,.\end{aligned} \tag{∞.73}$$

[10] Despite saying that we wouldn't belabor this any further, this is one of those unfortunate situations where we had to decide between decorating the inverse $\left(\widehat{H}^{-1}\right)_{ij}^{\tilde{\alpha}_1\tilde{\alpha}_2}$ with either a hat or a tilde, and we went with the hat. Hopefully the tildes on the sample indices, e.g. $\tilde{\alpha}_1, \tilde{\alpha}_2$, will remind you that the inverse of this stochastic object is taken only with respect to the training set. Note that at no point will we ever need the inverse of the stochastic NTK evaluated on a general dataset $\mathcal{D}$.

Thus, we've correctly taken care of the first type of the $1/n$ corrections, and with this step we've reduced the residual prediction error on the training set from the order-one error of our initial prediction

$$z_{i;\tilde{\alpha}}(t=0) - y_{i;\tilde{\alpha}} = O(1) \ , \tag{∞.74}$$

to a much smaller error of

$$z_{i;\tilde{\alpha}}(t=1) - y_{i;\tilde{\alpha}} = O\left(\frac{1}{n}\right) \ , \tag{∞.75}$$

loosely corresponding to the empirically-large initial decrease of error when training networks in practice. Moreover, the rest of the error in (∞.73) is now entirely due to the additional finite-width corrections encoded by the dNTK and first ddNTK, a consequence of representation learning.[11]

Second Update: Representation Learning Strikes Back

To reduce the training error further, we'll need to update our network again to further account for the fact that its representation evolved with the first update. In other words, we'll need to make a second gradient-descent update. If you'd like, you can imagine that this update corresponds to a second phase of training – following a first phase where the network significantly decreased its training error with the NTK evaluated at initialization – and so now the model must refine its features in order to further improve its performance.

Accordingly, since our training error is already down to $\sim 1/n$, our second update is going to be a lot smaller than our first. In particular, the update itself will necessarily only be of order $1/n$ so as to precisely cancel the remaining $1/n$ training error; that is, it's actually more of a *small step* than another *giant leap*.[12]

To determine which step we need to take, let's write the network output after a second update as

$$\begin{aligned}
&z_{i;\delta}(t=2) &&(\infty.76)\\
=&z_{i;\delta}(t=1) - \sum_{j,k,\tilde{\alpha}_1,\tilde{\alpha}_2} H_{ij;\delta\tilde{\alpha}_1}(t=1)\,\eta\kappa^{\tilde{\alpha}_1\tilde{\alpha}_2}_{jk}(t=1)\left[z_{k;\tilde{\alpha}_2}(t=1) - y_{k;\tilde{\alpha}_2}\right] + O\left(\frac{1}{n^2}\right)\\
=&z_{i;\delta}(t=1) - \sum_{j,\tilde{\alpha}_1,\tilde{\alpha}_2} H_{\delta\tilde{\alpha}_1}\,\eta\kappa^{\tilde{\alpha}_1\tilde{\alpha}_2}_{ij}(t=1)\left[z_{j;\tilde{\alpha}_2}(t=1) - y_{j;\tilde{\alpha}_2}\right] + O\left(\frac{1}{n^2}\right) \ ,
\end{aligned}$$

[11]In particular, this first update (∞.71) would satisfy the training condition (∞.66) only if the NTK were constant under gradient descent. There's actually a name given to this type of phenomenon, *lazy training*, referring to situations when the network function behaves as if it is equal to a linearzation around the initial value of its parameters [68]. As we know from our discussion in §10.4, if the NTK is constant, then the network is a linear model.

[12]As we will see, the overall learning rate is essentially the same for both updates; the second update is only smaller because the gradient of the loss after the first update is itself much smaller when near a minimum.

where we left the second update's product of global learning rate and generalized Newton tensor $\eta\kappa_{ij}^{\tilde{\alpha}_1\tilde{\alpha}_2}(t=1)$ unspecified for now. Note here that in the first equality we dropped the d(d)NTK terms from the update: since after our first update (∞.73) the training error has already decreased to $O(1/n)$, the would-be dNTK term is very subleading $\sim \widehat{\mathrm{d}H} \times [\epsilon(t=1)]^2 = O(1/n^3)$, and the would-be ddNTK term is ridiculously subleading $\sim \widehat{\mathrm{dd_I}H} \times [\epsilon(t=1)]^3 = O(1/n^4)$. Similarly, on the third line we replaced the NTK after the first step $H_{ij;\delta\tilde{\alpha}_1}(t=1)$ by the NTK mean at initialization $\delta_{ij}H_{\delta\tilde{\alpha}_1}$. Given the $1/n$-suppressed training error after the first step, this substitution can be justified for these two reasons in conjunction: *(i)* the update to the NTK $\widehat{H}_{ij;\delta\tilde{\alpha}}(t=1) - \widehat{H}_{ij;\delta\tilde{\alpha}}(t=0)$ is itself suppressed by $1/n$, cf. (∞.6), so we may use the version from before the update, and *(ii)* the NTK fluctuation is also suppressed compared to its mean, so we may then swap the stochastic NTK at initialization for its mean. This means that if we make the following choice for our *small step* second update

$$\eta\kappa_{ij}^{\tilde{\alpha}_1\tilde{\alpha}_2}(t=1) = \delta_{ij}\widetilde{H}^{\tilde{\alpha}_1\tilde{\alpha}_2} + O\left(\frac{1}{n}\right) , \tag{∞.77}$$

then it's easy to see from (∞.76) that our network will now be fully trained as (∞.66):

$$z_{i;\tilde{\alpha}}(t=2) - y_{i;\tilde{\alpha}} = O\left(\frac{1}{n^2}\right) . \tag{∞.78}$$

Thus, with our second update we were able to reduce the residual training error by an additional factor of $1/n$ as compared to the error after the first update (∞.73).[13]

For general inputs $\delta \in \mathcal{D}$, plugging our choice of learning rate and Newton tensor (∞.77) back into our second update (∞.76) and further re-expressing the network output

[13] This suggests that additional updates could continue to reduce the training error by additional factors of $1/n$. However, these further refinements – determined in terms of the higher-order corrections to our effective theory description – will be qualitatively the same as our leading finite-width description at $O(L/n)$. In other words, improving an infinite-width description to finite-width description incorporates representation learning, while more precise finite-width descriptions just allow the model to make further refinements to its features. Practically speaking, we expect our leading finite-width description to be very accurate for networks with reasonable values of the aspect ratio L/n.

after the first step $z_{i;\delta}(t=1)$ by (∞.70) with (∞.71), we get

$$
\begin{aligned}
&z_{i;\delta}(t=2) \qquad\qquad (\infty.79)\\
=&z_{i;\delta} - \sum_{\tilde{\alpha}_1,\tilde{\alpha}_2 \in \mathcal{A}} H_{\delta\tilde{\alpha}_1} H^{\tilde{\alpha}_1\tilde{\alpha}_2} (z_{i;\tilde{\alpha}_2} - y_{i;\tilde{\alpha}_2})\\
&+ \sum_{j=1}^{n_L} \sum_{\tilde{\alpha}_1,\tilde{\alpha}_2 \in \mathcal{A}} \left[\widehat{\Delta H}_{ij;\delta\tilde{\alpha}_1} - \sum_{\tilde{\alpha}_3,\tilde{\alpha}_4 \in \mathcal{A}} H_{\delta\tilde{\alpha}_3} \widetilde{H}^{\tilde{\alpha}_3\tilde{\alpha}_4} \widehat{\Delta H}_{ij;\tilde{\alpha}_4\tilde{\alpha}_1} \right] \widetilde{H}^{\tilde{\alpha}_1\tilde{\alpha}_2} (z_{j;\tilde{\alpha}_2} - y_{j;\tilde{\alpha}_2})\\
&- \sum_{j,k=1}^{n_L} \sum_{\tilde{\alpha}_1,\ldots,\tilde{\alpha}_4 \in \mathcal{A}} \left[\widehat{\Delta H}_{ij;\delta\tilde{\alpha}_1} - \sum_{\tilde{\alpha}_5,\tilde{\alpha}_6 \in \mathcal{A}} H_{\delta\tilde{\alpha}_5} \widetilde{H}^{\tilde{\alpha}_5\tilde{\alpha}_6} \widehat{\Delta H}_{ij;\tilde{\alpha}_6\tilde{\alpha}_1} \right]\\
&\qquad\qquad \times \widetilde{H}^{\tilde{\alpha}_1\tilde{\alpha}_2} \widehat{\Delta H}_{jk;\tilde{\alpha}_2\tilde{\alpha}_3} \widetilde{H}^{\tilde{\alpha}_3\tilde{\alpha}_4} (z_{k;\tilde{\alpha}_4} - y_{k;\tilde{\alpha}_4})\\
&+ \frac{1}{2} \sum_{j_1,j_2=1}^{n_L} \sum_{\tilde{\alpha}_1,\ldots,\tilde{\alpha}_4 \in \mathcal{A}} \left[\widehat{\mathrm{d}H}_{ij_1j_2;\delta\tilde{\alpha}_1\tilde{\alpha}_2} - \sum_{\tilde{\alpha}_5,\tilde{\alpha}_6 \in \mathcal{A}} H_{\delta\tilde{\alpha}_5} \widetilde{H}^{\tilde{\alpha}_5\tilde{\alpha}_6} \widehat{\mathrm{d}H}_{ij_1j_2;\tilde{\alpha}_6\tilde{\alpha}_1\tilde{\alpha}_2} \right]\\
&\qquad\qquad \times \widetilde{H}^{\tilde{\alpha}_1\tilde{\alpha}_3} \widetilde{H}^{\tilde{\alpha}_2\tilde{\alpha}_4} (z_{j_1;\tilde{\alpha}_3} - y_{j_1;\tilde{\alpha}_3})(z_{j_2;\tilde{\alpha}_4} - y_{j_2;\tilde{\alpha}_4})\\
&- \frac{1}{6} \sum_{j_1,j_2,j_3=1}^{n_L} \sum_{\tilde{\alpha}_1,\ldots,\tilde{\alpha}_6 \in \mathcal{A}} \left[\widehat{\mathrm{dd_I}H}_{ij_1j_2j_3;\delta\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3} - \sum_{\tilde{\alpha}_7,\tilde{\alpha}_8 \in \mathcal{A}} H_{\delta\tilde{\alpha}_7} \widetilde{H}^{\tilde{\alpha}_7\tilde{\alpha}_8} \widehat{\mathrm{dd_I}H}_{ij_1j_2j_3;\tilde{\alpha}_8\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3} \right]\\
&\qquad\qquad \times \widetilde{H}^{\tilde{\alpha}_1\tilde{\alpha}_4} \widetilde{H}^{\tilde{\alpha}_2\tilde{\alpha}_5} \widetilde{H}^{\tilde{\alpha}_3\tilde{\alpha}_6} (z_{j_1;\tilde{\alpha}_4} - y_{j_1;\tilde{\alpha}_4}) (z_{j_2;\tilde{\alpha}_5} - y_{j_2;\tilde{\alpha}_5}) (z_{j_3;\tilde{\alpha}_6} - y_{j_3;\tilde{\alpha}_6})\\
&+ O\left(\frac{1}{n^2}\right) .
\end{aligned}
$$

Here, we see that the expressions in the square brackets vanish identically for all the training inputs $\delta = \tilde{\alpha} \in \mathcal{A}$: our network is thus fully trained. In particular, since all of the variables on the right-hand side of (∞.79) are at initialization, this solution realizes our goal (∞.55) of expressing the output of a fully-trained network as a functional of such variables at initialization. Accordingly, the statistics of the fully-trained distribution (∞.56) can now be worked out from the joint preactivation-NTK-dNTK-ddNTKs distribution (∞.53) at initialization.[14]

While this is all very exciting, let us also caution you that these generalized second-order updates are probably best thought of as giving a simple theoretical model of a training algorithm – designed to let us understand the training process analytically – and by no means are we suggesting that they provide a good or useful option for practical optimization. Instead, the most practical algorithm for optimization is vanilla first-order gradient descent. As we've already pointed out that the output of a fully-trained network *does* depend on the details of the algorithm used for optimization, now we really have no choice left other than to explicitly analyze many many steps of gradient descent.

∞.2.2 Many Many Steps of Gradient Descent

In this extended subsection, we're going to study tensorial gradient descent (7.11) and optimize finite-width neural networks according to the MSE loss with a constant global

[14] Alternatively, we could have reached this same solution in a *single update* if we had instead made a

learning rate η.[15] With this progenitor-of-all-other-gradient-based-learning-algorithm algorithm, the network output will evolve as (∞.4)

$$\begin{aligned} z_{i;\delta}(t+1) =& z_{i;\delta}(t) - \eta \sum_{j,\tilde{\alpha}} H_{ij;\delta\tilde{\alpha}}(t) \left[z_{j;\tilde{\alpha}}(t) - y_{j;\tilde{\alpha}} \right] \\ &+ \frac{\eta^2}{2} \sum_{j_1,j_2,\tilde{\alpha}_1,\tilde{\alpha}_2} \mathrm{d}H_{ij_1j_2;\delta\tilde{\alpha}_1\tilde{\alpha}_2}(t) \left[z_{j_1;\tilde{\alpha}_1}(t) - y_{j_1;\tilde{\alpha}_1} \right] \left[z_{j_2;\tilde{\alpha}_2}(t) - y_{j_2;\tilde{\alpha}_2} \right] \\ &- \frac{\eta^3}{6} \sum_{j_1,j_2,j_3,\tilde{\alpha}_1,\tilde{\alpha}_2,\tilde{\alpha}_3} \widehat{\mathrm{dd_I} H}_{ij_1j_2j_3;\delta\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3} \\ &\qquad \times \left[z_{j_1;\tilde{\alpha}_1}(t) - y_{j_1;\tilde{\alpha}_1} \right] \left[z_{j_2;\tilde{\alpha}_2}(t) - y_{j_2;\tilde{\alpha}_2} \right] \left[z_{j_3;\tilde{\alpha}_3}(t) - y_{j_3;\tilde{\alpha}_3} \right] , \end{aligned} \tag{∞.81}$$

---

finely tuned giant leap, picking the generalized Newton tensor as

$$\begin{aligned} \eta \kappa_{ij}^{\tilde{\alpha}_1\tilde{\alpha}_2} =& \delta_{ij} \widetilde{H}^{\tilde{\alpha}_1\tilde{\alpha}_2} - \sum_{\tilde{\alpha}_3,\tilde{\alpha}_4 \in \mathcal{A}} \widetilde{H}^{\tilde{\alpha}_1\tilde{\alpha}_3} \widehat{\Delta H}_{ij;\tilde{\alpha}_3\tilde{\alpha}_4} \widetilde{H}^{\tilde{\alpha}_4\tilde{\alpha}_2} \\ &+ \sum_{k=1}^{n_L} \sum_{\tilde{\alpha}_3,\ldots,\tilde{\alpha}_6 \in \mathcal{A}} \widetilde{H}^{\tilde{\alpha}_1\tilde{\alpha}_3} \widehat{\Delta H}_{ik;\tilde{\alpha}_3\tilde{\alpha}_4} \widetilde{H}^{\tilde{\alpha}_4\tilde{\alpha}_5} \widehat{\Delta H}_{kj;\tilde{\alpha}_5\tilde{\alpha}_6} \widetilde{H}^{\tilde{\alpha}_6\tilde{\alpha}_2} \\ &+ \frac{1}{2} \sum_{k=1}^{n_L} \sum_{\tilde{\alpha}_3,\ldots,\tilde{\alpha}_6 \in \mathcal{A}} \widetilde{H}^{\tilde{\alpha}_1\tilde{\alpha}_3} \widetilde{H}^{\tilde{\alpha}_2\tilde{\alpha}_4} \widetilde{H}^{\tilde{\alpha}_5\tilde{\alpha}_6} \widehat{\mathrm{d}H}_{ijk;\tilde{\alpha}_3\tilde{\alpha}_4\tilde{\alpha}_5} \left(z_{k;\tilde{\alpha}_6} - y_{k;\tilde{\alpha}_6} \right) \\ &- \frac{1}{6} \sum_{k_1,k_2=1}^{n_L} \sum_{\tilde{\alpha}_3,\ldots,\tilde{\alpha}_8 \in \mathcal{A}} \widetilde{H}^{\tilde{\alpha}_1\tilde{\alpha}_3} \widetilde{H}^{\tilde{\alpha}_2\tilde{\alpha}_4} \widetilde{H}^{\tilde{\alpha}_5\tilde{\alpha}_7} \widetilde{H}^{\tilde{\alpha}_6\tilde{\alpha}_8} \widehat{\mathrm{dd_I} H}_{ijk_1k_2;\tilde{\alpha}_3\tilde{\alpha}_4\tilde{\alpha}_5\tilde{\alpha}_6} \\ &\qquad \times \left(z_{k_1;\tilde{\alpha}_7} - y_{k_1;\tilde{\alpha}_7} \right) \left(z_{k_2;\tilde{\alpha}_8} - y_{k_2;\tilde{\alpha}_8} \right) . \end{aligned} \tag{∞.80}$$

In essence, the first three terms of (∞.80) come from inverting the stochastic NTK, corresponding to our first-update giant leap (∞.71) and accounting for the instantiation-to-instantiation fluctuations in the NTK. In contrast, the final two terms correspond to our second-update small step (∞.78) and accounts for the dNTK-ddNTK-induced representation learning. Note that this generalized Newton tensor (∞.80) is *asymmetric* under the exchange of paired indices $(i, \tilde{\alpha}_1) \leftrightarrow (j, \tilde{\alpha}_2)$ due to the last term, and thus this finely-tuned update doesn't admit an alternative interpretation of optimization with a further generalized loss (∞.69).

This single update is analogous to the *direct optimization* solution (11.132) for quadratic regression in §11.4. The very finely-tuned nature of this single-update algorithm (∞.80) suggests that it's easier to make the fine adjustments required to reach a solution by taking many simpler steps rather than fewer complicated steps. We will see the other side of this next in §∞.2.2 when we study training by many many steps of vanilla gradient descent.

[15] Don't let the word *tensorial* scare you here; this just means that we will allow for our training hyperparameters $\lambda_b^{(\ell)}$ and $\lambda_W^{(\ell)}$ as part of the definition of the NTK. We hope it's already clear why including these hyperparameters is a good idea – if not, please flip back to §9.4 and reread the paragraphs on the learning rate equivalence principle – and they are actually simple to include practically as part of any optimization algorithm.

Moreover, as they are just part of the definition of the NTK, they have absolutely no consequence on the dynamics presented here; i.e. our solution also covers *non-tensorial* gradient descent, though in such a case you'd have different asymptotic solutions for the statistics of the NTK, dNTK, and ddNTKs. This is also why we think of the training hyperparameters $\lambda_b^{(\ell)}$ and $\lambda_W^{(\ell)}$ as being independent of the details of the optimization algorithm itself.

in conjunction the NTK will evolve as (∞.6)

$$
\begin{aligned}
&H_{i_1 i_2;\delta_1\delta_2}(t+1) \qquad\qquad (\infty.82)\\
=&H_{i_1 i_2;\delta_1\delta_2}(t) - \eta \sum_{j,\tilde{\alpha}} \Big(\mathrm{d}H_{i_1 i_2 j;\delta_1\delta_2\tilde{\alpha}}(t) + \mathrm{d}H_{i_2 i_1 j;\delta_2\delta_1\tilde{\alpha}}(t) \Big) \left[z_{j;\tilde{\alpha}}(t) - y_{j;\tilde{\alpha}} \right] \\
&+ \frac{\eta^2}{2} \sum_{j_1,j_2,\tilde{\alpha}_1,\tilde{\alpha}_2} \Big(\widehat{\mathrm{dd_I} H}_{i_1 i_2 j_1 j_2;\delta_1\delta_2\tilde{\alpha}_1\tilde{\alpha}_2} + \widehat{\mathrm{dd_I} H}_{i_2 i_1 j_1 j_2;\delta_2\delta_1\tilde{\alpha}_1\tilde{\alpha}_2} + 2\widehat{\mathrm{dd_{II}} H}_{i_1 i_2 j_1 j_2;\delta_1\delta_2\tilde{\alpha}_1\tilde{\alpha}_2} \Big) \\
&\qquad\qquad \times \left[z_{j_1;\tilde{\alpha}_1}(t) - y_{j_1;\tilde{\alpha}_1} \right] \left[z_{j_2;\tilde{\alpha}_2}(t) - y_{j_2;\tilde{\alpha}_2} \right] ,
\end{aligned}
$$

and in further conjunction the dNTK will evolve as (∞.9)

$$
\begin{aligned}
&\mathrm{d}H_{i_0 i_1 i_2;\delta_0\delta_1\delta_2}(t+1) \qquad\qquad (\infty.83)\\
=&\mathrm{d}H_{i_0 i_1 i_2;\delta_0\delta_1\delta_2}(t) \\
&- \eta \sum_{j,\tilde{\alpha}} \Big(\widehat{\mathrm{dd_I} H}_{i_0 i_1 i_2 j;\delta_0\delta_1\delta_2\tilde{\alpha}} + \widehat{\mathrm{dd_{II}} H}_{i_0 i_1 i_2 j;\delta_0\delta_1\delta_2\tilde{\alpha}} + \widehat{\mathrm{dd_{II}} H}_{i_0 i_2 i_1 j;\delta_0\delta_2\delta_1\tilde{\alpha}} \Big) \left[z_{j;\tilde{\alpha}}(t) - y_{j;\tilde{\alpha}} \right] .
\end{aligned}
$$

In writing these dynamical equations, we've stopped explicitly denoting that our equations have $O(1/n^2)$ errors, and we've also expressed the fact that the ddNTKs are t-independent at order $1/n$, using their values at initialization, $\widehat{\mathrm{dd_I} H}_{i_1 i_2 i_3 i_4;\delta_1\delta_2\delta_3\delta_4}$ and $\widehat{\mathrm{dd_{II}} H}_{i_1 i_2 i_3 i_4;\delta_1\delta_2\delta_3\delta_4}$, with hats to remind us of their stochasticity. These joint updates (∞.81), (∞.82), and (∞.83) are coupled *difference equations*, and the equations for the network output and the dynamical NTK are both *nonlinear* in the output $z_{i;\delta}$.

Although this seems daunting, we are now going to solve these equations in a closed form. First, we'll analyze the dynamics while neglecting the finite-width effects of the dNTK and ddNTKs, in which the problem will reduce to a single *linear* difference equation. Then, we'll use perturbation theory to incorporate the effect of all the NTK differentials and allow for a dynamically evolving NTK and dNTK.

Free Theory: Step-Independent NTK

Let's begin by setting the dNTK and ddNTKs to zero. Since their leading statistics are $1/n$-suppressed, we'll be able to use perturbation theory to reincorporate all their effects later. In this limit the NTK update equation (∞.82) is trivial, solved by the *free* or *step-independent* NTK:

$$
H_{i_1 i_1;\delta_1\delta_2}(t) = H_{i_1 i_2;\delta_1\delta_2}(t=0) \equiv \widehat{H}_{i_1 i_2;\delta_1\delta_2} . \qquad (\infty.84)
$$

Unsurprisingly, this just means that when the dNTK and ddNTKs vanish, the NTK doesn't update from its initialization.

Plugging this solution into the preactivation update equation (∞.81) and turning off the dNTK and ddNTKs, the remaining dynamical equation simplifies to

$$
z_{i;\delta}(t+1) = z_{i;\delta}(t) - \eta \sum_{j,\tilde{\alpha}} \widehat{H}_{ij;\delta\tilde{\alpha}} [z_{j;\tilde{\alpha}}(t) - y_{j;\tilde{\alpha}}] . \qquad (\infty.85)
$$

Thus, the residual training error, $z_{\tilde{\alpha}}(t) - y_{\tilde{\alpha}}$, sources the updates to the network output $z_\delta(t)$ for general inputs $\delta \in \mathcal{D}$. Moreover, when restricted to the inputs from the training set $\tilde{\alpha} \in \mathcal{A}$, we can rewrite this difference equation (∞.85) as

$$z_{i;\tilde{\alpha}}(t+1) - y_{i;\tilde{\alpha}} = \sum_{j,\tilde{\alpha}_1} \left(I_{ij;\tilde{\alpha}\tilde{\alpha}_1} - \eta \widehat{H}_{ij;\tilde{\alpha}\tilde{\alpha}_1} \right) \left[z_{j;\tilde{\alpha}_1}(t) - y_{j;\tilde{\alpha}_1} \right] , \tag{∞.86}$$

where we've defined the *identity operator*

$$I_{i_1 i_2;\tilde{\alpha}_1\tilde{\alpha}_2} \equiv \delta_{i_1 i_2} \delta_{\tilde{\alpha}_1\tilde{\alpha}_2} \, . \tag{∞.87}$$

In this form, (∞.86) is a first-order homogeneous linear difference equation for the residual training error, $z_{\tilde{\alpha}}(t) - y_{\tilde{\alpha}}$, which is just a fancy way of saying that this is going to be a piece of cake.

In particular, the update to the prediction error is just a simple multiplication by a constant matrix, and the solution is given by an exponential:

$$z^{\text{F}}_{i;\tilde{\alpha}}(t) - y_{i;\tilde{\alpha}} = \sum_{j,\tilde{\alpha}_1} U_{ij;\tilde{\alpha}\tilde{\alpha}_1}(t) \left(z_{j;\tilde{\alpha}_1} - y_{j;\tilde{\alpha}_1} \right) \, . \tag{∞.88}$$

Here, on the left-hand side, we've labeled the solution with an "F" to indicate it's the *free solution*, with the nonlinear effects from the dNTK and ddNTKs turned off; on the right-hand side, we have the residual training error at initialization, and the **step-evolution operator** is defined as an iterative product of t steps:

$$\begin{aligned} U_{i_t i_0;\tilde{\alpha}_t\tilde{\alpha}_0}(t) &\equiv \left[\left(I - \eta \widehat{H} \right)^t \right]_{i_t i_0;\tilde{\alpha}_t\tilde{\alpha}_0} \qquad (\infty.89) \\ &\equiv \sum_{\substack{i_1,\ldots i_{t-1} \\ \tilde{\alpha}_1,\ldots,\tilde{\alpha}_{t-1}}} \left(I_{i_t i_{t-1};\tilde{\alpha}_t\tilde{\alpha}_{t-1}} - \eta \widehat{H}_{i_t i_{t-1};\tilde{\alpha}_t\tilde{\alpha}_{t-1}} \right) \cdots \left(I_{i_1 i_0;\tilde{\alpha}_1,\tilde{\alpha}_0} - \eta \widehat{H}_{i_1 i_0;\tilde{\alpha}_1\tilde{\alpha}_0} \right) \, . \end{aligned}$$

For any positive-definite NTK and sufficiently small global learning rate η, this operator will exponentially decay to zero, $U(t) \to 0$, as the number of steps becomes large, $t \to \infty$.[16] Thus, the residual training error (∞.88) will vanish exponentially quickly,

$$\lim_{t\to\infty} z^{\text{F}}_{i;\tilde{\alpha}}(t) = y_{i;\tilde{\alpha}} \, , \tag{∞.90}$$

with the step scale for the decay of the individual components set by the step-independent NTK.[17]

[16] In particular, for this limit to converge we just need $||I - \eta\widehat{H}||_\infty < 1$, i.e. the largest eigenvalue of the operator $I - \eta\widehat{H}$ must be less than one. With our attention to the principles of criticality and equivalence, our choices of initialization and training hyperparameters were made so that the NTK is always of order one, and thus it's very easy for our networks to satisfy this constraint.

[17] If we wanted to study the ODE or *continuum limit* of the dynamics, we could take the global learning rate to zero, $\eta \to 0$, while holding the product, $\tau \equiv \eta t$, fixed. In such a limit, the step-evolution operator becomes simply $U(t) \to \exp(-\widehat{H}\tau)$. While such a limit is mostly unnecessary for any theoretical purpose – it's just as easy to study the discrete dynamics that actually describe the

Having now solved the free dynamics on the training set, we can plug this solution (∞.88) back into the difference equation (∞.85) for general inputs $\delta \in \mathcal{D}$. With the source known explicitly, we can easily write down a solution that satisfies the initial condition $z^{\text{F}}_{i;\delta}(t=0) = z_{i;\delta}$:

$$\begin{aligned} z^{\text{F}}_{i;\delta}(t) =& z_{i;\delta} - \sum_{j,\tilde{\alpha}} \widehat{H}_{ij;\delta\tilde{\alpha}} \left\{ \eta \sum_{s=0}^{t-1} \left[z^{\text{F}}_{j;\tilde{\alpha}}(s) - y_{j;\tilde{\alpha}} \right] \right\} \\ =& z_{i;\delta} - \sum_{j,\tilde{\alpha}_1,\tilde{\alpha}_2} \widehat{H}_{ij;\delta\tilde{\alpha}_1} a_{j;\tilde{\alpha}}(t) \, . \end{aligned} \tag{∞.91}$$

Here we've defined a dynamical helper function, with an explicit representation given by

$$\begin{aligned} a_{j;\tilde{\alpha}}(t) \equiv& \eta \sum_{s=0}^{t-1} \left[z^{\text{F}}_{j;\tilde{\alpha}}(s) - y_{j;\tilde{\alpha}} \right] = \eta \sum_{s=0}^{t-1} \left[\sum_{k,\tilde{\alpha}_1} U_{jk;\tilde{\alpha}\tilde{\alpha}_1}(t) \left(z_{k;\tilde{\alpha}_1} - y_{k;\tilde{\alpha}_1} \right) \right] \\ =& \eta \sum_{k,\tilde{\alpha}_1} \left\{ \sum_{s=0}^{t-1} \left[\left(I - \eta \widehat{H} \right)^s \right]_{jk;\tilde{\alpha}\tilde{\alpha}_1} \right\} \left(z_{k;\tilde{\alpha}_1} - y_{k;\tilde{\alpha}_1} \right) \\ =& \eta \sum_{k,m,\tilde{\alpha}_1,\tilde{\alpha}_2} \left\{ \left[I - \left(I - \eta \widehat{H} \right) \right]^{-1} \right\}^{\tilde{\alpha}\tilde{\alpha}_2}_{jm} \left[I - \left(I - \eta \widehat{H} \right)^t \right]_{mk;\tilde{\alpha}_2\tilde{\alpha}_1} \left(z_{k;\tilde{\alpha}_1} - y_{k;\tilde{\alpha}_1} \right) \\ =& \sum_{m,\tilde{\alpha}_2} \left(\widehat{H}^{-1} \right)^{\tilde{\alpha}\tilde{\alpha}_2}_{jm} \left\{ z_{m;\tilde{\alpha}_2} - y_{m;\tilde{\alpha}_2} - \left[z^{\text{F}}_{m;\tilde{\alpha}_2}(t) - y_{m;\tilde{\alpha}_2} \right] \right\} \, . \end{aligned} \tag{∞.92}$$

In this expression, to get to the third line we used the standard formula for evaluating geometric sums, $1 + x + x^2 + \cdots + x^{t-1} = (1 - x^t)/(1 - x)$, and to get to the final line we evaluated the inverse and substituted in for the step-evolution operator (∞.89). Recall also that $\widehat{H}^{-1}$, defined by (∞.72), is the inverse of the stochastic NTK submatrix evaluated on the training set for a particular realization of the parameters.[18]

In particular, for sufficiently small η, as the number of steps becomes large, $t \to \infty$, the residual error $z^{\text{F}}_{m;\tilde{\alpha}_2}(t) - y_{m;\tilde{\alpha}_2}$ exponentially vanishes (∞.90). Thus, the prediction for a general input $\delta \in \mathcal{D}$ (∞.91) will exponentially converge to

$$z^{\text{F}}_{i;\delta}(t=\infty) = z_{i;\delta} - \sum_{j,k,\tilde{\alpha}_1,\tilde{\alpha}_2} \widehat{H}_{ij;\delta\tilde{\alpha}_1} \left(\widehat{H}^{-1} \right)^{\tilde{\alpha}_1\tilde{\alpha}_2}_{jk} \left(z_{k;\tilde{\alpha}_2} - y_{k;\tilde{\alpha}_2} \right) \, . \tag{∞.93}$$

practical optimization algorithm – it does provide more substance to objection in footnote 5 of §7.2 of the name *neural tangent kernel* for the stochastic operator $\widehat{H}$. In particular, this continuum limit makes clear that the NTK is really best thought of as a *Hamiltonian*, as it generates the evolution of observables, and the step-evolution operator $U(t)$ is like a unitary time-evolution operator, albeit in *i*maginary time. More precisely, in the limit where the dNTK and ddNTKs are set to zero, the NTK is akin to a *free* Hamiltonian, with exactly solvable dynamics; with a nonzero dNTK or ddNTKs, the Hamiltonian includes nontrivial *interactions* and can be analyzed via time-dependent perturbation theory.

[18] Unfortunately, in the absence of a Newton tensor, the raised/lowered sample indices in (∞.92) do not align well. (In fact, the problem really began in the update equation (∞.85) with the doubled-lowered $\tilde{\alpha}$ index.) If you'd like, you can fix this by judicious use of identity matrices such as $\delta_{\tilde{\alpha}_1\tilde{\alpha}_2}$ and $\delta^{\tilde{\alpha}_3\tilde{\alpha}_4}$. However, such usage over-clutters the presentation and so is probably not worth it, despite the additional clarity and *type safety* that such index alignment provides.

Although we took the limit of a large number of steps here, the exponential convergence means that for any number of steps T such that $T \gtrsim (\eta \widehat{H})^{-1}$, the prediction error will be exponentially close to its final $T \to \infty$ value.

In fact, this solution (∞.93) precisely matches the solution we would have gotten in §∞.2.1 after the first update (∞.68) with the Newton tensor (∞.71), so long as the dNTK and ddNTKs are turned off. This means that the *algorithm dependence* that emerges at finite width is solely due to the presence of the NTK differentials and the resulting nonlinear dynamics.

Interacting Theory: Dynamical NTK and dNTK

Now, let's incorporate the nonzero dNTK and ddNTKs into our analysis. To do so, we need to decompose both the dynamical network output and the dynamical NTK as

$$z_{i;\delta}(t) \equiv z^{\mathrm{F}}_{i;\delta}(t) + z^{\mathrm{I}}_{i;\delta}(t)\,, \tag{∞.94}$$

$$H_{ij;\delta\tilde{\alpha}}(t) \equiv \widehat{H}_{ij;\delta\tilde{\alpha}} + H^{\mathrm{I}}_{ij;\delta\tilde{\alpha}}(t)\,. \tag{∞.95}$$

Here, for the network output, $z^{\mathrm{F}}_{i;\delta}(t)$ is the *free* part, satisfying the linear difference equation (∞.85) with a solution given by (∞.91), and $z^{\mathrm{I}}_{i;\delta}(t)$ is the *interacting* part, encapsulating the corrections due to all the nonzero NTK differentials. Similarly, for the NTK, $\widehat{H}_{ij;\delta\tilde{\alpha}}$ is the step-independent or *free* part, fixed at initialization, and $H^{\mathrm{I}}_{ij;\delta\tilde{\alpha}}(t)$ is the dynamical step-dependent or *interaction* NTK. Since both $z^{\mathrm{I}}_{i;\delta}(t)$ and $H^{\mathrm{I}}_{ij;\delta\tilde{\alpha}}(t)$ are absent in the free limit, $\widehat{\mathrm{d}H}$, $\widehat{\mathrm{dd}_{\mathrm{I}}H}$, $\widehat{\mathrm{dd}_{\mathrm{II}}H} \to 0$, at leading order we expect them to be a linear combination of these objects, schematically

$$z^{\mathrm{I}}(t) = [\text{thing } 0](t)\ \widehat{\mathrm{d}H} + [\text{thing } 1](t)\ \widehat{\mathrm{dd}_{\mathrm{I}}H} + [\text{thing } 2](t)\ \widehat{\mathrm{dd}_{\mathrm{II}}H}\,, \tag{∞.96}$$

$$H^{\mathrm{I}}(t) = [\widetilde{\text{thing}}\ 0](t)\ \widehat{\mathrm{d}H} + [\widetilde{\text{thing}}\ 1](t)\ \widehat{\mathrm{dd}_{\mathrm{I}}H} + [\widetilde{\text{thing}}\ 2](t)\ \widehat{\mathrm{dd}_{\mathrm{II}}H}\,, \tag{∞.97}$$

where various tensorial things will have various time dependencies. This means in turn that any product of $z^{\mathrm{I}}_{i;\delta}(t)$ or $H^{\mathrm{I}}_{ij;\delta\tilde{\alpha}}(t)$ with one of $\widehat{\mathrm{d}H}, \widehat{\mathrm{dd}_{\mathrm{I}}H}, \widehat{\mathrm{dd}_{\mathrm{II}}H}$ can be neglected to leading order, which is the main reason why we'll be able to systematically solve these nonlinear dynamics by perturbation theory. Finally, the initial condition for the interacting parts of the network output and the NTK must satisfy

$$z^{\mathrm{I}}_{i;\delta}(t=0) = 0\,, \tag{∞.98}$$

$$H^{\mathrm{I}}_{ij;\delta\tilde{\alpha}}(t=0) = 0\,, \tag{∞.99}$$

since the free part of the network output already satisfies $z^{\mathrm{F}}_{i;\delta}(t=0) = z_{i;\delta}$ at initialization, and the free part of the NTK is step-independent and thus *is* the NTK at initialization. With all those in mind, our goal now is to find a solution for $z^{\mathrm{I}}_{i;\delta}(t)$ such that the full network output, $z_{i;\delta}(t)$, the interaction NTK, $H^{\mathrm{I}}_{ij;\delta\tilde{\alpha}}(t)$, and the dynamical dNTK, $\mathrm{d}H_{i_0 i_1 i_2;\delta_0\delta_1\delta_2}(t)$, all together satisfy the coupled nonlinear dynamics (∞.81), (∞.82), and (∞.83) to leading order in $1/n$.

First, let's work out the dynamics of the dNTK. From (∞.83) we see that the updates to the dNTK are sourced by the residual training error $z_{i;\tilde{\alpha}}(t) - y_{i;\tilde{\alpha}}$. Using our decomposition for the network output, (∞.94) and the initial condition,

$$\mathrm{d}H_{i_0 i_1 i_2;\delta_0\delta_1\delta_2}(t=0) = \widehat{\mathrm{d}H}_{i_0 i_1 i_2;\delta_0\delta_1\delta_2}\,, \tag{∞.100}$$

after iterating the dynamics, we have

$$\begin{aligned}
&\mathrm{d}H_{i_0 i_1 i_2;\delta_0\delta_1\delta_2}(t) \qquad\qquad (\infty.101)\\
=&\widehat{\mathrm{d}H}_{i_0 i_1 i_2;\delta_0\delta_1\delta_2} - \sum_{j,\tilde{\alpha}} \Big(\widehat{\mathrm{dd_I}H}_{i_0 i_1 i_2 j;\delta_0\delta_1\delta_2\tilde{\alpha}} + \widehat{\mathrm{dd_{II}}H}_{i_0 i_1 i_2 j;\delta_0\delta_1\delta_2\tilde{\alpha}} + \widehat{\mathrm{dd_{II}}H}_{i_0 i_2 i_1 j;\delta_0\delta_2\delta_1\tilde{\alpha}}\Big)\\
&\qquad \times \left\{\eta \sum_{s=0}^{t-1}\Big[z^{\mathrm{F}}_{j;\tilde{\alpha}}(s) + z^{\mathrm{I}}_{j;\tilde{\alpha}}(s) - y_{j;\tilde{\alpha}}\Big]\right\}\\
=&\widehat{\mathrm{d}H}_{i_0 i_1 i_2;\delta_0\delta_1\delta_2} - \sum_{j,\tilde{\alpha}} \Big(\widehat{\mathrm{dd_I}H}_{i_0 i_1 i_2 j;\delta_0\delta_1\delta_2\tilde{\alpha}} + \widehat{\mathrm{dd_{II}}H}_{i_0 i_1 i_2 j;\delta_0\delta_1\delta_2\tilde{\alpha}} + \widehat{\mathrm{dd_{II}}H}_{i_0 i_2 i_1 j;\delta_0\delta_2\delta_1\tilde{\alpha}}\Big) a_{j;\tilde{\alpha}}(t)\,.
\end{aligned}$$

Here in the last step, we first dropped the product of $z^{\mathrm{I}}_{i;\delta}(t)$ with $\widehat{\mathrm{dd_I}H}$ as explained earlier, and then substituted in our dynamical helper function (∞.92). Now, plugging in our final expression from (∞.92) and neglecting the fluctuation part of the NTK inverse as subleading, we find an expression for the dynamical dNTK:

$$\begin{aligned}
&\widehat{\mathrm{d}H}_{i_0 i_1 i_2;\delta_0\delta_1\delta_2}(t)\\
=&\widehat{\mathrm{d}H}_{i_0 i_1 i_2;\delta_0\delta_1\delta_2} - \sum_{k,\tilde{\alpha}_1,\tilde{\alpha}_2} \Big(\widehat{\mathrm{dd_I}H}_{i_0 i_1 i_2 j;\delta_0\delta_1\delta_2\tilde{\alpha}_1} + \widehat{\mathrm{dd_{II}}H}_{i_0 i_1 i_2 j;\delta_0\delta_1\delta_2\tilde{\alpha}_1} + \widehat{\mathrm{dd_{II}}H}_{i_0 i_2 i_1 j;\delta_0\delta_2\delta_1\tilde{\alpha}_1}\Big)\\
&\qquad \times \widetilde{H}^{\tilde{\alpha}_1\tilde{\alpha}_2}\left\{z_{j;\tilde{\alpha}_2} - y_{j;\tilde{\alpha}_2} - \Big[z^{\mathrm{F}}_{j;\tilde{\alpha}_2}(t) - y_{j;\tilde{\alpha}_2}\Big]\right\}\,. \qquad\qquad (\infty.102)
\end{aligned}$$

In particular, the quantity in the curly brackets represents the difference in training errors between initialization and step t: the larger this difference – i.e. the more the residual training error decreases – the greater the evolution of the dNTK, and the more the meta feature functions evolve, undergoing their own form of *meta representation learning* over the course of training.

Next, using our decompositions for the network output and NTK, (∞.94) and (∞.95), we can rewrite the NTK dynamics (∞.82) as a difference equation for the interaction NTK:

$$\begin{aligned}
&H^{\mathrm{I}}_{i_1 i_2;\delta_1\delta_2}(t+1) \qquad\qquad (\infty.103)\\
=&H^{\mathrm{I}}_{i_1 i_2;\delta_1\delta_2}(t) - \eta \sum_{j,\tilde{\alpha}} \Big(\mathrm{d}H_{i_1 i_2 j;\delta_1\delta_2\tilde{\alpha}}(t) + \mathrm{d}H_{i_2 i_1 j;\delta_2\delta_1\tilde{\alpha}}(t)\Big)\Big[z^{\mathrm{F}}_{j;\tilde{\alpha}}(t) - y_{j;\tilde{\alpha}}\Big]\\
&+\frac{\eta^2}{2}\sum_{j_1,j_2,\tilde{\alpha}_1,\tilde{\alpha}_2} \Big(\widehat{\mathrm{dd_I}H}_{i_1 i_2 j_1 j_2;\delta_1\delta_2\tilde{\alpha}_1\tilde{\alpha}_2} + \widehat{\mathrm{dd_I}H}_{i_2 i_1 j_1 j_2;\delta_2\delta_1\tilde{\alpha}_1\tilde{\alpha}_2} + 2\widehat{\mathrm{dd_{II}}H}_{i_1 i_2 j_1 j_2;\delta_1\delta_2\tilde{\alpha}_1\tilde{\alpha}_2}\Big)\\
&\qquad\qquad \times \Big[z^{\mathrm{F}}_{j_1;\tilde{\alpha}_1}(t) - y_{j_1;\tilde{\alpha}_1}\Big]\Big[z^{\mathrm{F}}_{j_2;\tilde{\alpha}_2}(t) - y_{j_2;\tilde{\alpha}_2}\Big]\,.
\end{aligned}$$

Here, once again, we've dropped the interacting part of the network output on the right-hand side, as it is always multiplied by either the dNTK or the ddNTKs and thus will be subleading in $1/n$. Denoting the free part of the residual training error (∞.88) as

$$\epsilon^{\mathrm{F}}_{j;\tilde{\alpha}}(t) \equiv z^{\mathrm{F}}_{j;\tilde{\alpha}}(t) - y_{j;\tilde{\alpha}}\,, \tag{∞.104}$$

for notational convenience, and substituting in our solution for the dynamical dNTK (∞.102), we get

$$\begin{aligned}
&H^{\mathrm{I}}_{i_1 i_2;\delta_1\delta_2}(t+1) - H^{\mathrm{I}}_{i_1 i_2;\delta_1\delta_2}(t) \qquad\qquad (\infty.105)\\
&= -\eta \sum_{j,\tilde{\alpha}} \Big(\widehat{\mathrm{d}H}_{i_1 i_2 j;\delta_1\delta_2\tilde{\alpha}} + \widehat{\mathrm{d}H}_{i_2 i_1 j;\delta_2\delta_1\tilde{\alpha}}\Big)\epsilon^{\mathrm{F}}_{j;\tilde{\alpha}}(t)\\
&\quad + \eta \sum_{j,k,\tilde{\alpha},\tilde{\alpha}_1,\tilde{\alpha}_2} \Big(\widehat{\mathrm{dd_I}H}_{i_1 i_2 jk;\delta_1\delta_2\tilde{\alpha}\tilde{\alpha}_1} + \widehat{\mathrm{dd_{II}}H}_{i_1 i_2 jk;\delta_1\delta_2\tilde{\alpha}\tilde{\alpha}_1} + \widehat{\mathrm{dd_{II}}H}_{i_1 j i_2 k;\delta_1\tilde{\alpha}\delta_2\tilde{\alpha}_1}\\
&\qquad\qquad + \widehat{\mathrm{dd_I}H}_{i_2 i_1 jk;\delta_2\delta_1\tilde{\alpha}\tilde{\alpha}_1} + \widehat{\mathrm{dd_{II}}H}_{i_2 i_1 jk;\delta_2\delta_1\tilde{\alpha}\tilde{\alpha}_1} + \widehat{\mathrm{dd_{II}}H}_{i_2 j i_1 k;\delta_2\tilde{\alpha}\delta_1\tilde{\alpha}_1}\Big)\\
&\qquad\qquad\qquad \times \widetilde{H}^{\tilde{\alpha}_1\tilde{\alpha}_2}\epsilon^{\mathrm{F}}_{j;\tilde{\alpha}}(t)\left[z_{k;\tilde{\alpha}_2} - y_{k;\tilde{\alpha}_2} - \epsilon^{\mathrm{F}}_{k;\tilde{\alpha}_2}(t)\right]\\
&\quad + \frac{\eta^2}{2} \sum_{j_1,j_2,\tilde{\alpha}_1,\tilde{\alpha}_2} \Big(\widehat{\mathrm{dd_I}H}_{i_1 i_2 j_1 j_2;\delta_1\delta_2\tilde{\alpha}_1\tilde{\alpha}_2} + \widehat{\mathrm{dd_I}H}_{i_2 i_1 j_1 j_2;\delta_2\delta_1\tilde{\alpha}_1\tilde{\alpha}_2} + 2\widehat{\mathrm{dd_{II}}H}_{i_1 i_2 j_1 j_2;\delta_1\delta_2\tilde{\alpha}_1\tilde{\alpha}_2}\Big)\\
&\qquad\qquad\qquad \times \epsilon^{\mathrm{F}}_{j_1;\tilde{\alpha}_1}(t)\,\epsilon^{\mathrm{F}}_{j_2;\tilde{\alpha}_2}(t)\,.
\end{aligned}$$

In particular, the step dependence on the right-hand side is expressed entirely in terms of the free residual training error $\epsilon^{\mathrm{F}}_{j;\tilde{\alpha}}(t)$, and each term is either linear or quadratic in $\epsilon^{\mathrm{F}}_{j;\tilde{\alpha}}(t)$. Thus, in order to solve this difference equation and get $H^{\mathrm{I}}_{i_1 i_2;\delta_1\delta_2}(t)$, we'll just have to compute sums over these terms.

One of those sums – the one that's linear in the free residual training error – is the dynamical helper function $a_{j;\tilde{\alpha}}(t) \equiv \eta \sum_{s=0}^{t-1} \epsilon^{\mathrm{F}}_{j;\tilde{\alpha}}(t)$ that we evaluated in (∞.92). The other type of sum is quadratic in the free residual training error, which will define a second dynamical helper function:

$$\begin{aligned}
&b_{j_1 j_2;\tilde{\alpha}_1\tilde{\alpha}_2}(t) \qquad\qquad (\infty.106)\\
\equiv&\eta \sum_{s=0}^{t-1} \epsilon^{\mathrm{F}}_{j_1;\tilde{\alpha}_1}(t)\epsilon^{\mathrm{F}}_{j_2;\tilde{\alpha}_2}(t)\\
=&\eta \sum_{k_1,k_2,\tilde{\alpha}_3,\tilde{\alpha}_4} \sum_{s=0}^{t-1} \Big[\Big(I - \eta\widehat{H}\Big)^s\Big]_{j_1 k_1;\tilde{\alpha}_1\tilde{\alpha}_3}\Big[\Big(I - \eta\widehat{H}\Big)^s\Big]_{j_2 k_2;\tilde{\alpha}_2\tilde{\alpha}_4}(z_{k_1;\tilde{\alpha}_3} - y_{k_1;\tilde{\alpha}_3})\,(z_{k_2;\tilde{\alpha}_4} - y_{k_2;\tilde{\alpha}_4})\\
=&\sum_{\tilde{\alpha}_3,\tilde{\alpha}_4} X^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4}_{\mathrm{II}} \left[(z_{j_1;\tilde{\alpha}_3} - y_{j_1;\tilde{\alpha}_3})\,(z_{j_2;\tilde{\alpha}_4} - y_{j_2;\tilde{\alpha}_4}) - \epsilon^{\mathrm{F}}_{j_1;\tilde{\alpha}_3}(t)\epsilon^{\mathrm{F}}_{j_2;\tilde{\alpha}_4}(t)\right] + O\Big(\frac{1}{n}\Big)\,.
\end{aligned}$$

To evaluate this sum, we again used our expression for the free residual training error, (∞.88), in conjunction with the definition of the step-evolution operator, (∞.89). Then, we replaced the stochastic NTK $\widehat{H}_{ij;\tilde{\alpha}_1\tilde{\alpha}_2}$ of the training set by its mean $\delta_{ij}H_{\tilde{\alpha}_1\tilde{\alpha}_2}$ at the

cost of subleading corrections, and formally performed the geometric sum as in (∞.92). This last operation yielded an *inverting tensor* $X_{\mathrm{II}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4}$ implicitly defined by

$$\begin{aligned}\delta^{\tilde{\alpha}_1}_{\ \tilde{\alpha}_5}\delta^{\tilde{\alpha}_2}_{\ \tilde{\alpha}_6} &= \sum_{\tilde{\alpha}_3,\tilde{\alpha}_4\in\mathcal{A}} X_{\mathrm{II}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4}\frac{1}{\eta}\Big[\delta_{\tilde{\alpha}_3\tilde{\alpha}_5}\delta_{\tilde{\alpha}_4\tilde{\alpha}_6} - (\delta_{\tilde{\alpha}_3\tilde{\alpha}_5} - \eta\widetilde{H}_{\tilde{\alpha}_3\tilde{\alpha}_5})(\delta_{\tilde{\alpha}_4\tilde{\alpha}_6} - \eta\widetilde{H}_{\tilde{\alpha}_4\tilde{\alpha}_6})\Big] \\ &= \sum_{\tilde{\alpha}_3,\tilde{\alpha}_4\in\mathcal{A}} X_{\mathrm{II}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4}\Big(\widetilde{H}_{\tilde{\alpha}_3\tilde{\alpha}_5}\delta_{\tilde{\alpha}_4\tilde{\alpha}_6} + \delta_{\tilde{\alpha}_3\tilde{\alpha}_5}\widetilde{H}_{\tilde{\alpha}_4\tilde{\alpha}_6} - \eta\widetilde{H}_{\tilde{\alpha}_3\tilde{\alpha}_5}\widetilde{H}_{\tilde{\alpha}_4\tilde{\alpha}_6}\Big)\,. \qquad (\infty.107)\end{aligned}$$

This tensor is a generalization of the familiar inverting matrix $X_{\mathrm{I}}^{\tilde{\alpha}_1\tilde{\alpha}_2} \equiv \widetilde{H}^{\tilde{\alpha}_1\tilde{\alpha}_2}$ for geometric sums over matrices that satisfies

$$\delta^{\tilde{\alpha}_1}_{\ \tilde{\alpha}_3} = \sum_{\tilde{\alpha}_2\in\mathcal{A}} X_{\mathrm{I}}^{\tilde{\alpha}_1\tilde{\alpha}_2}\frac{1}{\eta}\Big[\delta_{\tilde{\alpha}_2\tilde{\alpha}_3} - (\delta_{\tilde{\alpha}_2\tilde{\alpha}_3} - \eta\widetilde{H}_{\tilde{\alpha}_2\tilde{\alpha}_3})\Big] = \sum_{\tilde{\alpha}_2\in\mathcal{A}} X_{\mathrm{I}}^{\tilde{\alpha}_1\tilde{\alpha}_2}\widetilde{H}_{\tilde{\alpha}_2\tilde{\alpha}_3}\,, \qquad (\infty.108)$$

and appeared in the last expression of the dynamical helper function $a_{j;\tilde{\alpha}}(t)$ (∞.92). While we are on fire like an activated neuron, let's also define a final dynamical helper function for sums that are cubic in the free residual training error:

$$\begin{aligned}&c_{j_1j_2j_3;\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3}(t) \qquad (\infty.109)\\ \equiv&\eta\sum_{s=0}^{t}\epsilon^{\mathrm{F}}_{j_1;\tilde{\alpha}_1}(t)\epsilon^{\mathrm{F}}_{j_2;\tilde{\alpha}_2}(t)\epsilon^{\mathrm{F}}_{j_3;\tilde{\alpha}_3}(t)\\ =&\sum_{\tilde{\alpha}_4,\tilde{\alpha}_5,\tilde{\alpha}_6} X_{\mathrm{III}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4\tilde{\alpha}_5\tilde{\alpha}_6}\Big[\left(z_{k_1;\tilde{\alpha}_4} - y_{k_1;\tilde{\alpha}_4}\right)\left(z_{k_2;\tilde{\alpha}_5} - y_{k_2;\tilde{\alpha}_5}\right)\left(z_{k_3;\tilde{\alpha}_6} - y_{k_3;\tilde{\alpha}_6}\right)\\ &\qquad - \epsilon^{\mathrm{F}}_{j_1;\tilde{\alpha}_4}(t)\,\epsilon^{\mathrm{F}}_{j_2;\tilde{\alpha}_5}(t)\,\epsilon^{\mathrm{F}}_{j_3;\tilde{\alpha}_6}(t)\Big] + O\Big(\frac{1}{n}\Big)\,.\end{aligned}$$

In this case, the inverting tensor $X_{\mathrm{III}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4\tilde{\alpha}_5\tilde{\alpha}_6}$ is implicitly defined by

$$\begin{aligned}&\delta^{\tilde{\alpha}_1}_{\ \tilde{\alpha}_7}\delta^{\tilde{\alpha}_2}_{\ \tilde{\alpha}_8}\delta^{\tilde{\alpha}_3}_{\ \tilde{\alpha}_9} \qquad (\infty.110)\\ =&\sum_{\tilde{\alpha}_4,\tilde{\alpha}_5,\tilde{\alpha}_6} X_{\mathrm{III}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4\tilde{\alpha}_5\tilde{\alpha}_6}\Big[\Big(\widetilde{H}_{\tilde{\alpha}_4\tilde{\alpha}_7}\delta_{\tilde{\alpha}_5\tilde{\alpha}_8}\delta_{\tilde{\alpha}_6\tilde{\alpha}_9} + \delta_{\tilde{\alpha}_4\tilde{\alpha}_7}\widetilde{H}_{\tilde{\alpha}_5\tilde{\alpha}_8}\delta_{\tilde{\alpha}_6\tilde{\alpha}_9} + \delta_{\tilde{\alpha}_4\tilde{\alpha}_7}\delta_{\tilde{\alpha}_5\tilde{\alpha}_8}\widetilde{H}_{\tilde{\alpha}_6\tilde{\alpha}_9}\Big)\\ &\qquad - \eta\Big(\widetilde{H}_{\tilde{\alpha}_4\tilde{\alpha}_7}\widetilde{H}_{\tilde{\alpha}_5\tilde{\alpha}_8}\delta_{\tilde{\alpha}_6\tilde{\alpha}_9} + \widetilde{H}_{\tilde{\alpha}_4\tilde{\alpha}_7}\delta_{\tilde{\alpha}_5\tilde{\alpha}_8}\widetilde{H}_{\tilde{\alpha}_6\tilde{\alpha}_9} + \delta_{\tilde{\alpha}_4\tilde{\alpha}_7}\widetilde{H}_{\tilde{\alpha}_5\tilde{\alpha}_8}\widetilde{H}_{\tilde{\alpha}_6\tilde{\alpha}_9}\Big)\\ &\qquad + \eta^2\Big(\widetilde{H}_{\tilde{\alpha}_4\tilde{\alpha}_7}\widetilde{H}_{\tilde{\alpha}_5\tilde{\alpha}_8}\widetilde{H}_{\tilde{\alpha}_6\tilde{\alpha}_9}\Big)\Big]\,.\end{aligned}$$

Essentially, these three dynamical helper functions encode the step dependence of various geometric sums of the free residual training error $\epsilon^{\mathrm{F}}_{j;\tilde{\alpha}}(t)$. Note that we have

$$a_{j_1;\tilde{\alpha}_1}(t=0) = 0\,, \qquad (\infty.111)$$

$$b_{j_1j_2;\tilde{\alpha}_1\tilde{\alpha}_2}(t=0) = 0\,, \qquad (\infty.112)$$

$$c_{j_1j_2j_3;\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3}(t=0) = 0\,, \qquad (\infty.113)$$

and so they all vanish at initialization, while at the end of training we have

$$a_{j_1;\tilde{\alpha}_1}(\infty) = \sum_{\tilde{\alpha}_2} X_{\mathrm{I}}^{\tilde{\alpha}_1\tilde{\alpha}_2} \left(z_{j_1;\tilde{\alpha}_2} - y_{j_1;\tilde{\alpha}_2}\right) + O\left(\frac{1}{n}\right) , \tag{∞.114}$$

$$b_{j_1j_2;\tilde{\alpha}_1\tilde{\alpha}_2}(\infty) = \sum_{\tilde{\alpha}_3,\tilde{\alpha}_4} X_{\mathrm{II}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4} \left(z_{j_1;\tilde{\alpha}_3} - y_{j_1;\tilde{\alpha}_3}\right)\left(z_{j_2;\tilde{\alpha}_4} - y_{j_2;\tilde{\alpha}_4}\right) + O\left(\frac{1}{n}\right) , \tag{∞.115}$$

$$\begin{aligned} c_{j_1j_2j_3;\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3}(\infty) = & \sum_{\tilde{\alpha}_4,\tilde{\alpha}_5,\tilde{\alpha}_6} X_{\mathrm{III}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4\tilde{\alpha}_5\tilde{\alpha}_6} \left(z_{j_1;\tilde{\alpha}_4} - y_{j_1;\tilde{\alpha}_4}\right)\left(z_{j_2;\tilde{\alpha}_5} - y_{j_2;\tilde{\alpha}_5}\right)\left(z_{j_3;\tilde{\alpha}_6} - y_{j_3;\tilde{\alpha}_6}\right) \\ & + O\left(\frac{1}{n}\right) , \end{aligned} \tag{∞.116}$$

since the free residual training error $\epsilon^{\mathrm{F}}_{j;\tilde{\alpha}}(t)$ vanishes exponentially quickly, cf. (∞.90).

With the help of the first two of these dynamical helper functions, we can semi-compactly write the solution to the difference equation for the interaction NTK (∞.105) as

$$\begin{aligned} & H^{\mathrm{I}}_{i_1i_2;\delta_1\delta_2}(t) \\ = & -\sum_{j,\tilde{\alpha}} \Big(\widehat{\mathrm{d}H}_{i_1i_2j;\delta_1\delta_2\tilde{\alpha}} + \widehat{\mathrm{d}H}_{i_2i_1j;\delta_2\delta_1\tilde{\alpha}}\Big) a_{j;\tilde{\alpha}}(t) \\ & + \sum_{j,k,\tilde{\alpha}_1,\tilde{\alpha}_2,\tilde{\alpha}_3} \Big(\widehat{\mathrm{dd}_{\mathrm{I}}H}_{i_1i_2jk;\delta_1\delta_2\tilde{\alpha}_1\tilde{\alpha}_2} + \widehat{\mathrm{dd}_{\mathrm{II}}H}_{i_1i_2jk;\delta_1\delta_2\tilde{\alpha}_1\tilde{\alpha}_2} + \widehat{\mathrm{dd}_{\mathrm{II}}H}_{i_1ji_2k;\delta_1\tilde{\alpha}_1\delta_2\tilde{\alpha}_2} \\ & \qquad + \widehat{\mathrm{dd}_{\mathrm{I}}H}_{i_2i_1jk;\delta_2\delta_1\tilde{\alpha}_1\tilde{\alpha}_2} + \widehat{\mathrm{dd}_{\mathrm{II}}H}_{i_2i_1jk;\delta_2\delta_1\tilde{\alpha}_1\tilde{\alpha}_2} + \widehat{\mathrm{dd}_{\mathrm{II}}H}_{i_2ji_1k;\delta_2\tilde{\alpha}_1\delta_1\tilde{\alpha}_2}\Big) \\ & \qquad\qquad \times \widetilde{H}^{\tilde{\alpha}_2\tilde{\alpha}_3}\Big[\left(z_{k;\tilde{\alpha}_3} - y_{k;\tilde{\alpha}_3}\right) a_{j;\tilde{\alpha}_1}(t) - b_{jk;\tilde{\alpha}_1\tilde{\alpha}_3}(t)\Big] \\ & + \frac{\eta}{2} \sum_{j_1,j_2,\tilde{\alpha}_1,\tilde{\alpha}_2} \Big(\widehat{\mathrm{dd}_{\mathrm{I}}H}_{i_1i_2j_1j_2;\delta_1\delta_2\tilde{\alpha}_1\tilde{\alpha}_2} + \widehat{\mathrm{dd}_{\mathrm{I}}H}_{i_2i_1j_1j_2;\delta_2\delta_1\tilde{\alpha}_1\tilde{\alpha}_2} + 2\widehat{\mathrm{dd}_{\mathrm{II}}H}_{i_1i_2j_1j_2;\delta_1\delta_2\tilde{\alpha}_1\tilde{\alpha}_2}\Big) b_{j_1j_2;\tilde{\alpha}_1\tilde{\alpha}_2}(t) . \end{aligned} \tag{∞.117}$$

As a quick sanity check, the vanishing initial condition for the interaction NTK, (∞.99), is satisfied due to the vanishing of the helper functions at initialization, (∞.111) and (∞.112). We can also plug in (∞.114) and (∞.115) to evaluate the change in NTK at the end of training. In particular, we see that the larger the change in the NTK, the more the feature functions evolve. Thus, more initial error entails more representation learning over the course of training.

Lastly, we need to determine the step dependence of the interacting part of the network output, $z^{\mathrm{I}}_{i;\delta}(t)$. Inserting our free-interacting decomposition, (∞.94), into our dynamics for the network output, (∞.81), and using the fact that the free part satisfies the step-independent evolution equation (∞.86), we can reorganize terms to find a dynamical equation for the interacting part only:

$$z^{\mathrm{I}}_{i;\delta}(t+1) = z^{\mathrm{I}}_{i;\delta}(t) - \sum_{j,\tilde{\alpha}} \eta\, \widehat{H}_{ij;\delta\tilde{\alpha}}\, z^{\mathrm{I}}_{j;\tilde{\alpha}}(t) + \eta\, \mathbb{F}_{i;\delta}(t) . \tag{∞.118}$$

Here, we've defined a *damping force*:

$$\mathbb{F}_{i;\delta}(t) \equiv -\sum_{j,\tilde{\alpha}} H^{\mathrm{I}}_{ij;\delta\tilde{\alpha}}(t)\,\epsilon^{\mathrm{F}}_{j;\tilde{\alpha}}(t) + \frac{\eta}{2}\sum_{j_1,j_2,\tilde{\alpha}_1,\tilde{\alpha}_2} \mathrm{d}H_{ij_1j_2;\delta\tilde{\alpha}_1\tilde{\alpha}_2}(t)\,\epsilon^{\mathrm{F}}_{j_1;\tilde{\alpha}_1}(t)\,\epsilon^{\mathrm{F}}_{j_2;\tilde{\alpha}_2}(t) \quad (\infty.119)$$
$$- \frac{\eta^2}{6}\sum_{j_1,j_2,j_3,\tilde{\alpha}_1,\tilde{\alpha}_2,\tilde{\alpha}_3} \widehat{\mathrm{dd}_{\mathrm{I}}H}_{ij_1j_2j_3;\delta\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3}\,\epsilon^{\mathrm{F}}_{j_1;\tilde{\alpha}_1}(t)\,\epsilon^{\mathrm{F}}_{j_2;\tilde{\alpha}_2}(t)\,\epsilon^{\mathrm{F}}_{j_3;\tilde{\alpha}_3}(t)\,.$$

Since we have solutions for the interaction NTK and the dNTK dynamics in terms of the free residual training error solution $\epsilon^{\mathrm{F}}_{j;\tilde{\alpha}}(t)$, ($\infty$.117) and ($\infty$.102), this damping force is an explicitly known function of the step t.

Let us now try to implicitly express the solution to this difference equation (∞.118) as a sum over steps. First for inputs in the training set $\tilde{\alpha} \in \mathcal{A}$, the dynamics of the interacting part of the output, (∞.118), reduce to a first-order inhomogeneous linear difference equation, with the damping force ruining the homogeneity:

$$z^{\mathrm{I}}_{i;\tilde{\alpha}}(t+1) = \sum_{j,\tilde{\alpha}_1}\left(I_{ij;\tilde{\alpha}\tilde{\alpha}_1} - \eta\widehat{H}_{ij;\tilde{\alpha}\tilde{\alpha}_1}\right) z^{\mathrm{I}}_{j;\tilde{\alpha}_1}(t) + \eta\,\mathbb{F}_{i;\tilde{\alpha}}(t)\,. \quad (\infty.120)$$

We can formally solve this equation as a convolution of the damping force with the free step-evolution operator (∞.89),

$$z^{\mathrm{I}}_{i;\tilde{\alpha}}(t) = \eta\sum_{s=0}^{t-1}\sum_{j,\tilde{\alpha}_1} U_{ij;\tilde{\alpha}\tilde{\alpha}_1}(t-1-s)\,\mathbb{F}_{j;\tilde{\alpha}_1}(s)\,, \quad (\infty.121)$$

which satisfies the initial condition $z^{\mathrm{I}}_{i;\tilde{\alpha}}(t=0) = 0$. Plugging this result back into the dynamical equation for general inputs $\delta \in \mathcal{D}$, (∞.118), we can then find a solution for the interacting part of the network output for such general inputs:

$$z^{\mathrm{I}}_{i;\delta}(t) = \eta\sum_{s=0}^{t-1}\left[\mathbb{F}_{i;\delta}(s) - \sum_{j,\tilde{\alpha}}\widehat{H}_{ij;\delta\tilde{\alpha}}\,z^{\mathrm{I}}_{j;\tilde{\alpha}}(s)\right]. \quad (\infty.122)$$

In order to further simplify these expressions, we need to work out one convoluted sum:

$$\begin{aligned}
\eta\sum_{s=0}^{t-1} z^{\mathrm{I}}_{j;\tilde{\alpha}}(s) =& \eta^2\sum_{s=0}^{t-1}\sum_{\tilde{s}=0}^{s-1}\sum_{k,\tilde{\alpha}_1} U_{jk;\tilde{\alpha}\tilde{\alpha}_1}(s-1-\tilde{s})\,\mathbb{F}_{k;\tilde{\alpha}_1}(\tilde{s}) \quad (\infty.123)\\
=& \eta\sum_{u=0}^{t-2}\sum_{k,\tilde{\alpha}_1}\left[\eta\sum_{\tilde{u}=0}^{t-u-2} U_{jk;\tilde{\alpha}\tilde{\alpha}_1}(\tilde{u})\right]\mathbb{F}_{k;\tilde{\alpha}_1}(u)\\
=& \eta\sum_{u=0}^{t-2}\sum_{k,m,\tilde{\alpha}_1,\tilde{\alpha}_2}\left(\widehat{H}^{-1}\right)^{\tilde{\alpha}\tilde{\alpha}_2}_{jm}\left[I - U(t-u-1)\right]_{mk;\tilde{\alpha}_2\tilde{\alpha}_1}\mathbb{F}_{k;\tilde{\alpha}_1}(u)\\
=& \sum_{m,\tilde{\alpha}_2}\left(\widehat{H}^{-1}\right)^{\tilde{\alpha}\tilde{\alpha}_2}_{jm}\left\{\left[\eta\sum_{u=0}^{t-2}\mathbb{F}_{m;\tilde{\alpha}_2}(u)\right] + \left[\eta\mathbb{F}_{m;\tilde{\alpha}_2}(t-1) - z^{\mathrm{I}}_{m;\tilde{\alpha}_2}(t)\right]\right\}\\
=& \sum_{m,\tilde{\alpha}_2}\left(\widehat{H}^{-1}\right)^{\tilde{\alpha}\tilde{\alpha}_2}_{jm}\left\{\left[\eta\sum_{u=0}^{t-1}\mathbb{F}_{m;\tilde{\alpha}_2}(u)\right] - z^{\mathrm{I}}_{m;\tilde{\alpha}_2}(t)\right\},
\end{aligned}$$

Step by step, on the first line we used the expression for the formal solution (∞.121); on the second line we first reversed the order of the sums as $\sum_{s=0}^{t-1}\sum_{\tilde{s}=0}^{s-1} = \sum_{\tilde{s}=0}^{t-2}\sum_{s=\tilde{s}+1}^{t-1}$, and then we rewrote these sums in terms of new variables, $u \equiv \tilde{s}$ and $\tilde{u} \equiv s - 1 - \tilde{s}$; on the third line, we performed the geometric sum exactly as in (∞.92); on the fourth line, we used our formal solution (∞.121) at step t; and on the final line, we combined terms to extend the limits of the sum over the damping force. Plugging this evaluation back into our formal solution for $z^{\mathrm{I}}_{i;\delta}(t)$, ($\infty$.122), we get a slightly less formal solution:

$$z^{\mathrm{I}}_{i;\delta}(t) = \eta \sum_{s=0}^{t-1} \left[\mathbb{F}_{i;\delta}(s) - \sum_{\tilde{\alpha}_1,\tilde{\alpha}_2} H_{\delta\tilde{\alpha}_1} \widetilde{H}^{\tilde{\alpha}_1\tilde{\alpha}_2} \mathbb{F}_{i;\tilde{\alpha}_2}(s) \right] + \sum_{\tilde{\alpha}_1,\tilde{\alpha}_2} H_{\delta\tilde{\alpha}_1} \widetilde{H}^{\tilde{\alpha}_1\tilde{\alpha}_2} z^{\mathrm{I}}_{i;\tilde{\alpha}_2}(t) + O\left(\frac{1}{n^2}\right) . \tag{∞.124}$$

In this result, we've replaced the stochastic NTK by its mean, $\widehat{H}_{ij;\delta_1\delta_2} \to \delta_{ij} H_{\delta_1,\delta_2}$, as every term is otherwise already proportional to the dNTK or ddNTKs. As a quick sanity check, note that for inputs in the training set, $\delta \to \tilde{\alpha} \in \mathcal{A}$, this general expression reduces to an identity on $z^{\mathrm{I}}_{i;\tilde{\alpha}}(t)$.

Ultimately, what we care about is the interacting solution at the end of training, $t \to \infty$. As before, let's assume that the product of η with the stochastic NTK is sufficiently small such that the free step-evolution operator,

$$\lim_{t\to\infty} U(t) \propto \exp\Big(-\eta \widehat{H} t\Big) , \tag{∞.125}$$

exponentially decays to zero.[19] Then, for training inputs, the interacting part of the network outputs, $z^{\mathrm{I}}_{i;\tilde{\alpha}}(t)$, will exponentially converge to zero

$$\lim_{t\to\infty} z^{\mathrm{I}}_{i;\tilde{\alpha}}(t) = 0 . \tag{∞.126}$$

To see why this holds, note that the dampening force (∞.119) decays exponentially as

$$\lim_{s\to\infty} \mathbb{F}(s) \propto \exp\Big(-\eta \widehat{H} s\Big) , \tag{∞.127}$$

since its leading behavior is linearly proportional to the free residual training error $\epsilon^{\mathrm{F}}_{j;\tilde{\alpha}}(t)$. Combined with ($\infty$.125), this means that the interacting solution (∞.121) converges as

$$\begin{aligned} \lim_{t\to\infty} z^{\mathrm{I}}_{i;\tilde{\alpha}}(t) &= \eta \lim_{t\to\infty} \sum_{s=0}^{t-1} \sum_{j,\tilde{\alpha}_1} U_{ij;\tilde{\alpha}\tilde{\alpha}_1}(t-1-s)\, \mathbb{F}_{j;\tilde{\alpha}_1}(s) \\ &\propto \lim_{t\to\infty} \left\{ \eta t \, \exp\Big[-(t-1)\eta \widehat{H}\Big] \right\} = 0 , \end{aligned} \tag{∞.128}$$

which is slightly slower than the convergence of the free solution $z^{\mathrm{F}}_{i;\tilde{\alpha}}(t) \propto \exp(-\widehat{H} t)$. Thus, overall the training algorithm converges:

$$\lim_{t\to\infty} z_{i;\tilde{\alpha}}(t) - y_{i;\tilde{\alpha}} = \lim_{t\to\infty} \Big[z^{\mathrm{F}}_{i;\tilde{\alpha}}(t) + z^{\mathrm{I}}_{i;\tilde{\alpha}}(t) \Big] - y_{i;\tilde{\alpha}} = 0 , \tag{∞.129}$$

[19] Note that since the step-evolution operator $U(t)$ is constructed in (∞.89) from the *step-independent* NTK, $\widehat{H}_{ij;\delta\tilde{\alpha}}$, the condition for this convergence is the same as the free analysis discussed in footnote 16.

where here we also used the free solution (∞.90).[20] Incidentally, and of possible broader interest, this altogether shows that gradient descent converges exponentially quickly to a zero-error minimum for realistic deep neural networks of finite width and nonzero depth, up to errors that are at most quadratic in our effective theory cutoff L/n.

For general inputs, the interacting part of the output, $z^{\mathrm{I}}_{i;\delta}(t)$, is given by the expression (∞.124). With the convergence on the training set in mind (∞.126), the expression in the end-of-training limit reduces to

$$\lim_{t\to\infty} z^{\mathrm{I}}_{i;\delta}(t) = \left[\eta \sum_{s=0}^{\infty} \mathbb{F}_{i;\delta}(s)\right] - \sum_{\tilde{\alpha}_1,\tilde{\alpha}_2} H_{\delta\tilde{\alpha}_1} \widetilde{H}^{\tilde{\alpha}_1\tilde{\alpha}_2} \left[\eta \sum_{s=0}^{\infty} \mathbb{F}_{i;\tilde{\alpha}_2}(s)\right] . \tag{∞.130}$$

Thus, all that remains is for us to perform an infinite sum over the damping force:

$$\begin{aligned}
\eta \sum_{s=0}^{\infty} \mathbb{F}_{i;\delta}(s) \equiv & - \sum_{j,\tilde{\alpha}_1} \left[\eta \sum_{s=0}^{\infty} H^{\mathrm{I}}_{ij;\delta\tilde{\alpha}_1}(s)\, \epsilon^{\mathrm{F}}_{j;\tilde{\alpha}_1}(s)\right] \\
& + \frac{\eta}{2} \sum_{j_1,j_2,\tilde{\alpha}_1,\tilde{\alpha}_2} \left[\eta \sum_{s=0}^{\infty} \mathrm{d}H_{ij_1j_2;\delta\tilde{\alpha}_1\tilde{\alpha}_2}(s)\, \epsilon^{\mathrm{F}}_{j_1;\tilde{\alpha}_1}(s)\, \epsilon^{\mathrm{F}}_{j_2;\tilde{\alpha}_2}(s)\right] \\
& - \frac{\eta^2}{6} \sum_{j_1,j_2,j_3,\tilde{\alpha}_1,\tilde{\alpha}_2,\tilde{\alpha}_3} \left[\eta \sum_{s=0}^{\infty} \widehat{\mathrm{dd_I} H}_{ij_1j_2j_3;\delta\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3}\, \epsilon^{\mathrm{F}}_{j_1;\tilde{\alpha}_1}(s)\, \epsilon^{\mathrm{F}}_{j_2;\tilde{\alpha}_2}(s)\, \epsilon^{\mathrm{F}}_{j_3;\tilde{\alpha}_3}(s)\right] .
\end{aligned} \tag{∞.131}$$

The third sum is exactly the end-of-training limit of the third dynamical helper function $c_{j_1j_2j_3;\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3}(t)$ that we already evaluated in (∞.116), giving

$$\begin{aligned}
& \eta \sum_{s=0}^{\infty} \widehat{\mathrm{dd_I} H}_{ij_1j_2j_3;\delta\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3}\, \epsilon^{\mathrm{F}}_{j_1;\tilde{\alpha}_1}(s)\, \epsilon^{\mathrm{F}}_{j_2;\tilde{\alpha}_2}(s)\, \epsilon^{\mathrm{F}}_{j_3;\tilde{\alpha}_3}(s) \\
= & \sum_{\tilde{\alpha}_4,\tilde{\alpha}_5,\tilde{\alpha}_6} \widehat{\mathrm{dd_I} H}_{ij_1j_2j_3;\delta\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3} X^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4\tilde{\alpha}_5\tilde{\alpha}_6}_{\mathrm{III}} \left(z_{j_1;\tilde{\alpha}_4} - y_{j_1;\tilde{\alpha}_4}\right) \left(z_{j_2;\tilde{\alpha}_5} - y_{j_2;\tilde{\alpha}_5}\right) \left(z_{j_3;\tilde{\alpha}_6} - y_{j_3;\tilde{\alpha}_6}\right) .
\end{aligned} \tag{∞.132}$$

Thus, we are left with two more sums to evaluate. Let's proceed slowly: everything is simple, but there are a lot of terms to get right.

To start, we can evaluate the second sum in (∞.131) as

$$\begin{aligned}
& \eta \sum_{s=0}^{\infty} \widehat{\mathrm{d}H}_{ij_1j_2;\delta\tilde{\alpha}_1\tilde{\alpha}_2}(s)\, \epsilon^{\mathrm{F}}_{j_1;\tilde{\alpha}_1}(s)\, \epsilon^{\mathrm{F}}_{j_2;\tilde{\alpha}_2}(s) \\
= & \widehat{\mathrm{d}H}_{ij_1j_2;\delta\tilde{\alpha}_1\tilde{\alpha}_2} \sum_{s=0}^{\infty} \left[\eta\, \epsilon^{\mathrm{F}}_{j_1;\tilde{\alpha}_1}(s)\, \epsilon^{\mathrm{F}}_{j_2;\tilde{\alpha}_2}(s)\right] \\
& - \sum_{k,\tilde{\alpha}_3,\tilde{\alpha}_4} \left(\widehat{\mathrm{dd_I} H}_{ij_1j_2k;\delta\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3} + 2\widehat{\mathrm{dd_{II}} H}_{ij_1j_2k;\delta\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3}\right) \\
& \qquad \times \widetilde{H}^{\tilde{\alpha}_3\tilde{\alpha}_4} \sum_{s=0}^{\infty} \left\{\eta\, \epsilon^{\mathrm{F}}_{j_1;\tilde{\alpha}_1}(s)\, \epsilon^{\mathrm{F}}_{j_2;\tilde{\alpha}_2}(s) \left[z_{k;\tilde{\alpha}_4} - y_{k;\tilde{\alpha}_4} - \epsilon^{\mathrm{F}}_{k;\tilde{\alpha}_4}(s)\right]\right\} ,
\end{aligned} \tag{∞.133}$$

[20] Remember that in this section we have stopped explicitly denoting that there are $O\left(1/n^2\right)$ corrections. Recalling our fully-trained condition (∞.66), this result (∞.129) should be understood to be true up to such corrections, cf. (∞.78) for our two-step solution where the situation was analogous.

where we substituted in our dynamical dNTK solution, (∞.102) and used the fact that the overall expression is symmetric under $(\tilde{\alpha}_1, j_1) \leftrightarrow (\tilde{\alpha}_2, j_2)$ to combine the two $\mathrm{dd_{II}}H$ terms. Then, using our already evaluated sums, (∞.115) and (∞.116), we get

$$\begin{aligned}
&\eta \sum_{s=0}^{\infty} \widehat{\mathrm{d}H}_{ij_1j_2;\delta\tilde{\alpha}_1\tilde{\alpha}_2}(s)\, \epsilon^{\mathrm{F}}_{j_1;\tilde{\alpha}_1}(s)\, \epsilon^{\mathrm{F}}_{j_2;\tilde{\alpha}_2}(s) \qquad (\infty.134)\\
&= \sum_{\tilde{\alpha}_3,\tilde{\alpha}_4} \widehat{\mathrm{d}H}_{ij_1j_2;\delta\tilde{\alpha}_1\tilde{\alpha}_2} X_{\mathrm{II}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4} \left(z_{j_1;\tilde{\alpha}_3} - y_{j_1;\tilde{\alpha}_3}\right)\left(z_{j_2;\tilde{\alpha}_4} - y_{j_2;\tilde{\alpha}_4}\right)\\
&\quad - \sum_{k,\tilde{\alpha}_3,\ldots,\tilde{\alpha}_6} \Big(\widehat{\mathrm{dd_I}H}_{ij_1j_2k;\delta\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3} + 2\widehat{\mathrm{dd_{II}}H}_{ij_1j_2k;\delta\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3}\Big)\\
&\qquad\qquad \times Y_1^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4\tilde{\alpha}_5\tilde{\alpha}_6} \left(z_{j_1;\tilde{\alpha}_4} - y_{j_1;\tilde{\alpha}_4}\right)\left(z_{j_2;\tilde{\alpha}_5} - y_{j_2;\tilde{\alpha}_5}\right)\left(z_{k;\tilde{\alpha}_6} - y_{k;\tilde{\alpha}_6}\right) .
\end{aligned}$$

where we introduced a shorthand

$$Y_1^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4\tilde{\alpha}_5\tilde{\alpha}_6} \equiv X_{\mathrm{II}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_4\tilde{\alpha}_5} \widetilde{H}^{\tilde{\alpha}_3\tilde{\alpha}_6} - \sum_{\tilde{\alpha}_7} \widetilde{H}^{\tilde{\alpha}_3\tilde{\alpha}_7} X_{\mathrm{III}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_7\tilde{\alpha}_4\tilde{\alpha}_5\tilde{\alpha}_6} , \qquad (\infty.135)$$

to ease the collection of terms later.

To finish, let us write the first sum in (∞.131) as

$$\begin{aligned}
&\eta \sum_{s=0}^{\infty} H^{\mathrm{I}}_{ij;\delta\tilde{\alpha}_1}(s)\, \epsilon^{\mathrm{F}}_{j;\tilde{\alpha}_1}(s) \qquad (\infty.136)\\
&= -\sum_{k,\tilde{\alpha}_2} \Big(\widehat{\mathrm{d}H}_{ijk;\delta\tilde{\alpha}_1\tilde{\alpha}_2} + \widehat{\mathrm{d}H}_{jik;\tilde{\alpha}_1\delta\tilde{\alpha}_2}\Big) \sum_{s=0}^{\infty} \Big[\eta\, \epsilon^{\mathrm{F}}_{j;\tilde{\alpha}_1}(s)\, a_{k;\tilde{\alpha}_2}(s)\Big]\\
&\quad + \sum_{k_1,k_2,\tilde{\alpha}_2,\tilde{\alpha}_3,\tilde{\alpha}_4} \Big(\widehat{\mathrm{dd_I}H}_{ijk_1k_2;\delta\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3} + \widehat{\mathrm{dd_{II}}H}_{ijk_1k_2;\delta\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3} + \widehat{\mathrm{dd_{II}}H}_{ik_1jk_2;\delta\tilde{\alpha}_2\tilde{\alpha}_1\tilde{\alpha}_3}\\
&\qquad\qquad + \widehat{\mathrm{dd_I}H}_{jik_1k_2;\tilde{\alpha}_1\delta\tilde{\alpha}_2\tilde{\alpha}_3} + \widehat{\mathrm{dd_{II}}H}_{jik_1k_2;\tilde{\alpha}_1\delta\tilde{\alpha}_2\tilde{\alpha}_3} + \widehat{\mathrm{dd_{II}}H}_{jk_1ik_2;\tilde{\alpha}_1\tilde{\alpha}_2\delta\tilde{\alpha}_3}\Big)\\
&\qquad\qquad\qquad \times \widetilde{H}^{\tilde{\alpha}_3\tilde{\alpha}_4} \sum_{s=0}^{\infty} \Big\{\eta\, \epsilon^{\mathrm{F}}_{j;\tilde{\alpha}_1}(s) \Big[\left(z_{k_2;\tilde{\alpha}_4} - y_{k_2;\tilde{\alpha}_4}\right) a_{k_1;\tilde{\alpha}_2}(s) - b_{k_1k_2;\tilde{\alpha}_2\tilde{\alpha}_4}(s)\Big]\Big\}\\
&\quad + \frac{\eta}{2} \sum_{k_1,k_2,\tilde{\alpha}_2,\tilde{\alpha}_3} \Big(\widehat{\mathrm{dd_I}H}_{ijk_1k_2;\delta\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3} + \widehat{\mathrm{dd_I}H}_{jik_1k_2;\tilde{\alpha}_1\delta\tilde{\alpha}_2\tilde{\alpha}_3} + 2\widehat{\mathrm{dd_{II}}H}_{ijk_1k_2;\delta\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3}\Big)\\
&\qquad\qquad\qquad \times \sum_{s=0}^{\infty} \Big[\eta\, \epsilon^{\mathrm{F}}_{j;\tilde{\alpha}_1}(s)\, b_{k_1k_2;\tilde{\alpha}_2\tilde{\alpha}_3}(s)\Big] ,
\end{aligned}$$

where we substituted in our solution for the interaction NTK, (∞.117). Then, substituting for the helper functions with (∞.92) and (∞.106), performing the *additional* sums

over these terms, and then using the end-of-training limits (∞.114)–(∞.116), we get

$$
\begin{aligned}
&\eta \sum_{s=0}^{\infty} H^{\mathrm{I}}_{ij;\delta\tilde{\alpha}_1}(s)\epsilon^{\mathrm{F}}_{j;\tilde{\alpha}_1}(s) \qquad (\infty.137)\\
&= - \sum_{k,\tilde{\alpha}_2,\tilde{\alpha}_3,\tilde{\alpha}_4} \Big(\widehat{\mathrm{d}H}_{ijk;\delta\tilde{\alpha}_1\tilde{\alpha}_2} + \widehat{\mathrm{d}H}_{jik;\tilde{\alpha}_1\delta\tilde{\alpha}_2}\Big) Y_2^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4} \left(z_{j;\tilde{\alpha}_3} - y_{j;\tilde{\alpha}_3}\right)\left(z_{k;\tilde{\alpha}_4} - y_{k;\tilde{\alpha}_4}\right)\\
&\quad + \sum_{k_1,k_2,\tilde{\alpha}_2,\tilde{\alpha}_3,\tilde{\alpha}_4,\tilde{\alpha}_5,\tilde{\alpha}_6} \Big(\widehat{\mathrm{dd}_{\mathrm{I}}H}_{ijk_1k_2;\delta\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3} + \widehat{\mathrm{dd}_{\mathrm{II}}H}_{ijk_1k_2;\delta\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3} + \widehat{\mathrm{dd}_{\mathrm{II}}H}_{ik_1jk_2;\delta\tilde{\alpha}_2\tilde{\alpha}_1\tilde{\alpha}_3}\\
&\qquad\qquad + \widehat{\mathrm{dd}_{\mathrm{I}}H}_{jik_1k_2;\tilde{\alpha}_1\delta\tilde{\alpha}_2\tilde{\alpha}_3} + \widehat{\mathrm{dd}_{\mathrm{II}}H}_{jik_1k_2;\tilde{\alpha}_1\delta\tilde{\alpha}_2\tilde{\alpha}_3} + \widehat{\mathrm{dd}_{\mathrm{II}}H}_{jk_1ik_2;\tilde{\alpha}_1\tilde{\alpha}_2\delta\tilde{\alpha}_3}\Big)\\
&\qquad\qquad \times Y_3^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4\tilde{\alpha}_5\tilde{\alpha}_6} \left(z_{j;\tilde{\alpha}_4} - y_{j;\tilde{\alpha}_4}\right)\left(z_{k_1;\tilde{\alpha}_5} - y_{k_1;\tilde{\alpha}_5}\right)\left(z_{k_2;\tilde{\alpha}_6} - y_{k_2;\tilde{\alpha}_6}\right)\\
&\quad + \frac{\eta}{2} \sum_{k_1,k_2,\tilde{\alpha}_2,\tilde{\alpha}_3,\tilde{\alpha}_4,\tilde{\alpha}_5,\tilde{\alpha}_6} \Big(\widehat{\mathrm{dd}_{\mathrm{I}}H}_{ijk_1k_2;\delta\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3} + \widehat{\mathrm{dd}_{\mathrm{I}}H}_{jik_1k_2;\tilde{\alpha}_1\delta\tilde{\alpha}_2\tilde{\alpha}_3} + 2\widehat{\mathrm{dd}_{\mathrm{II}}H}_{ijk_1k_2;\delta\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3}\Big)\\
&\qquad\qquad \times Y_4^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4\tilde{\alpha}_5\tilde{\alpha}_6} \left(z_{j;\tilde{\alpha}_4} - y_{j;\tilde{\alpha}_4}\right)\left(z_{k_1;\tilde{\alpha}_5} - y_{k_1;\tilde{\alpha}_5}\right)\left(z_{k_2;\tilde{\alpha}_6} - y_{k_2;\tilde{\alpha}_6}\right) ,
\end{aligned}
$$

where we introduced additional shorthands

$$
\begin{aligned}
Y_2^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4} \equiv& \widetilde{H}^{\tilde{\alpha}_1\tilde{\alpha}_3}\widetilde{H}^{\tilde{\alpha}_2\tilde{\alpha}_4} - \sum_{\tilde{\alpha}_5} \widetilde{H}^{\tilde{\alpha}_2\tilde{\alpha}_5} X_{\mathrm{II}}^{\tilde{\alpha}_1\tilde{\alpha}_5\tilde{\alpha}_3\tilde{\alpha}_4} , \qquad (\infty.138)\\
Y_3^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4\tilde{\alpha}_5\tilde{\alpha}_6} \equiv& \widetilde{H}^{\tilde{\alpha}_1\tilde{\alpha}_4}\widetilde{H}^{\tilde{\alpha}_2\tilde{\alpha}_5}\widetilde{H}^{\tilde{\alpha}_3\tilde{\alpha}_6} - \sum_{\tilde{\alpha}_7} \widetilde{H}^{\tilde{\alpha}_2\tilde{\alpha}_7} X_{\mathrm{II}}^{\tilde{\alpha}_1\tilde{\alpha}_7\tilde{\alpha}_4\tilde{\alpha}_5}\widetilde{H}^{\tilde{\alpha}_3\tilde{\alpha}_6} , \qquad (\infty.139)\\
&- \sum_{\tilde{\alpha}_7} \widetilde{H}^{\tilde{\alpha}_1\tilde{\alpha}_4}\widetilde{H}^{\tilde{\alpha}_3\tilde{\alpha}_7} X_{\mathrm{II}}^{\tilde{\alpha}_2\tilde{\alpha}_7\tilde{\alpha}_5\tilde{\alpha}_6} + \sum_{\tilde{\alpha}_7,\tilde{\alpha}_8,\tilde{\alpha}_9} \widetilde{H}^{\tilde{\alpha}_3\tilde{\alpha}_9} X_{\mathrm{II}}^{\tilde{\alpha}_2\tilde{\alpha}_9\tilde{\alpha}_7\tilde{\alpha}_8} X_{\mathrm{III}}^{\tilde{\alpha}_1\tilde{\alpha}_7\tilde{\alpha}_8\tilde{\alpha}_4\tilde{\alpha}_5\tilde{\alpha}_6}\\
Y_4^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4\tilde{\alpha}_5\tilde{\alpha}_6} \equiv& \widetilde{H}^{\tilde{\alpha}_1\tilde{\alpha}_4} X_{\mathrm{II}}^{\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_5\tilde{\alpha}_6} - \sum_{\tilde{\alpha}_7,\tilde{\alpha}_8} X_{\mathrm{II}}^{\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_7\tilde{\alpha}_8} X_{\mathrm{III}}^{\tilde{\alpha}_1\tilde{\alpha}_7\tilde{\alpha}_8\tilde{\alpha}_4\tilde{\alpha}_5\tilde{\alpha}_6} . \qquad (\infty.140)
\end{aligned}
$$

Now, it's time to frantically flip through pages and collect everything we computed. Plugging the sums (∞.137), (∞.134), and (∞.132) back into our expression for the sum over the damping force, (∞.131), plugging that back into our expression for the interaction part of the network output (∞.130), and then combining with the free part of the network output, (∞.93), we obtain our fully-trained solution for finite-width

networks trained by gradient descent:

$$
\begin{aligned}
& z_{i;\delta}(t=\infty) \qquad (\infty.141)\\
\equiv & z^{\mathrm{F}}_{i;\delta}(t=\infty) + z^{\mathrm{I}}_{i;\delta}(t=\infty)\\
= & z_{i;\delta} - \sum_{j,k,\tilde{\alpha}_1,\tilde{\alpha}_2} \widehat{H}_{ij;\delta\tilde{\alpha}_1} \left(\widehat{H}^{-1}\right)^{\tilde{\alpha}_1\tilde{\alpha}_2}_{jk} \left(z_{k;\tilde{\alpha}_2} - y_{k;\tilde{\alpha}_2}\right)\\
& + \sum_{j_1,j_2,\tilde{\alpha}_1,\tilde{\alpha}_2,\tilde{\alpha}_3,\tilde{\alpha}_4} \left[\widehat{\mathrm{d}H}_{j_1ij_2;\tilde{\alpha}_1\delta\tilde{\alpha}_2} - \sum_{\tilde{\alpha}_5,\tilde{\alpha}_6} H_{\delta\tilde{\alpha}_5} \widetilde{H}^{\tilde{\alpha}_5\tilde{\alpha}_6} \widehat{\mathrm{d}H}_{j_1ij_2;\tilde{\alpha}_1\tilde{\alpha}_6\tilde{\alpha}_2} \right]\\
& \qquad \times Z^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4}_{\mathrm{A}} \left(z_{j_1;\tilde{\alpha}_3} - y_{j_1;\tilde{\alpha}_3}\right)\left(z_{j_2;\tilde{\alpha}_4} - y_{j_2;\tilde{\alpha}_4}\right)\\
& + \sum_{j_1,j_2,\tilde{\alpha}_1,\tilde{\alpha}_2,\tilde{\alpha}_3,\tilde{\alpha}_4} \left[\widehat{\mathrm{d}H}_{ij_1j_2;\delta\tilde{\alpha}_1\tilde{\alpha}_2} - \sum_{\tilde{\alpha}_5,\tilde{\alpha}_6} H_{\delta\tilde{\alpha}_5} \widetilde{H}^{\tilde{\alpha}_5\tilde{\alpha}_6} \widehat{\mathrm{d}H}_{ij_1j_2;\tilde{\alpha}_6\tilde{\alpha}_1\tilde{\alpha}_2} \right]\\
& \qquad \times Z^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4}_{\mathrm{B}} \left(z_{j_1;\tilde{\alpha}_3} - y_{j_1;\tilde{\alpha}_3}\right)\left(z_{j_2;\tilde{\alpha}_4} - y_{j_2;\tilde{\alpha}_4}\right)\\
& + \sum_{\substack{j_1,j_2,j_3,\\ \tilde{\alpha}_1,\tilde{\alpha}_2,\tilde{\alpha}_3,\tilde{\alpha}_4,\tilde{\alpha}_5,\tilde{\alpha}_6}} \left[\widehat{\mathrm{dd}_{\mathrm{I}}H}_{j_1ij_2j_3;\tilde{\alpha}_1\delta\tilde{\alpha}_2\tilde{\alpha}_3} - \sum_{\tilde{\alpha}_7,\tilde{\alpha}_8} H_{\delta\tilde{\alpha}_7} \widetilde{H}^{\tilde{\alpha}_7\tilde{\alpha}_8} \widehat{\mathrm{dd}_{\mathrm{I}}H}_{j_1ij_2j_3;\tilde{\alpha}_1\tilde{\alpha}_8\tilde{\alpha}_2\tilde{\alpha}_3} \right]\\
& \qquad \times Z^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4\tilde{\alpha}_5\tilde{\alpha}_6}_{\mathrm{IA}} \left(z_{j_1;\tilde{\alpha}_4} - y_{j_1;\tilde{\alpha}_4}\right)\left(z_{j_2;\tilde{\alpha}_5} - y_{j_2;\tilde{\alpha}_5}\right)\left(z_{j_3;\tilde{\alpha}_6} - y_{j_3;\tilde{\alpha}_6}\right)\\
& + \sum_{\substack{j_1,j_2,j_3,\\ \tilde{\alpha}_1,\tilde{\alpha}_2,\tilde{\alpha}_3,\tilde{\alpha}_4,\tilde{\alpha}_5,\tilde{\alpha}_6}} \left[\widehat{\mathrm{dd}_{\mathrm{I}}H}_{ij_1j_2j_3;\delta\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3} - \sum_{\tilde{\alpha}_7,\tilde{\alpha}_8} H_{\delta\tilde{\alpha}_7} \widetilde{H}^{\tilde{\alpha}_7\tilde{\alpha}_8} \widehat{\mathrm{dd}_{\mathrm{I}}H}_{ij_1j_2j_3;\tilde{\alpha}_8\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3} \right]\\
& \qquad \times Z^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4\tilde{\alpha}_5\tilde{\alpha}_6}_{\mathrm{IB}} \left(z_{j_1;\tilde{\alpha}_4} - y_{j_1;\tilde{\alpha}_4}\right)\left(z_{j_2;\tilde{\alpha}_5} - y_{j_2;\tilde{\alpha}_5}\right)\left(z_{j_3;\tilde{\alpha}_6} - y_{j_3;\tilde{\alpha}_6}\right)\\
& + \sum_{\substack{j_1,j_2,j_3,\\ \tilde{\alpha}_1,\tilde{\alpha}_2,\tilde{\alpha}_3,\tilde{\alpha}_4,\tilde{\alpha}_5,\tilde{\alpha}_6}} \left[\widehat{\mathrm{dd}_{\mathrm{II}}H}_{j_1j_2ij_3;\tilde{\alpha}_1\tilde{\alpha}_2\delta\tilde{\alpha}_3} - \sum_{\tilde{\alpha}_7,\tilde{\alpha}_8} H_{\delta\tilde{\alpha}_7} \widetilde{H}^{\tilde{\alpha}_7\tilde{\alpha}_8} \widehat{\mathrm{dd}_{\mathrm{II}}H}_{j_1j_2ij_3;\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_8\tilde{\alpha}_3} \right]\\
& \qquad \times Z^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4\tilde{\alpha}_5\tilde{\alpha}_6}_{\mathrm{IIA}} \left(z_{j_1;\tilde{\alpha}_4} - y_{j_1;\tilde{\alpha}_4}\right)\left(z_{j_2;\tilde{\alpha}_5} - y_{j_2;\tilde{\alpha}_5}\right)\left(z_{j_3;\tilde{\alpha}_6} - y_{j_3;\tilde{\alpha}_6}\right)\\
& + \sum_{\substack{j_1,j_2,j_3,\\ \tilde{\alpha}_1,\tilde{\alpha}_2,\tilde{\alpha}_3,\tilde{\alpha}_4,\tilde{\alpha}_5,\tilde{\alpha}_6}} \left[\widehat{\mathrm{dd}_{\mathrm{II}}H}_{ij_1j_2j_3;\delta\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3} - \sum_{\tilde{\alpha}_7,\tilde{\alpha}_8} H_{\delta\tilde{\alpha}_7} \widetilde{H}^{\tilde{\alpha}_7\tilde{\alpha}_8} \widehat{\mathrm{dd}_{\mathrm{II}}H}_{ij_1j_2j_3;\tilde{\alpha}_8\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3} \right]\\
& \qquad \times Z^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4\tilde{\alpha}_5\tilde{\alpha}_6}_{\mathrm{IIB}} \left(z_{j_1;\tilde{\alpha}_4} - y_{j_1;\tilde{\alpha}_4}\right)\left(z_{j_2;\tilde{\alpha}_5} - y_{j_2;\tilde{\alpha}_5}\right)\left(z_{j_3;\tilde{\alpha}_6} - y_{j_3;\tilde{\alpha}_6}\right)\\
& + O\left(\frac{1}{n^2}\right) .
\end{aligned}
$$

Here we defined our final tensors with our final alphabet letter (and various subscripts),

$$Z_{\text{A}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4} \equiv Y_2^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4}\,, \tag{∞.142}$$

$$Z_{\text{B}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4} \equiv Y_2^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4} + \frac{\eta}{2} X_{\text{II}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4}\,, \tag{∞.143}$$

$$Z_{\text{IA}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4\tilde{\alpha}_5\tilde{\alpha}_6} \equiv - Y_3^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4\tilde{\alpha}_5\tilde{\alpha}_6} - \frac{\eta}{2} Y_4^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4\tilde{\alpha}_5\tilde{\alpha}_6}\,, \tag{∞.144}$$

$$\begin{aligned} Z_{\text{IB}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4\tilde{\alpha}_5\tilde{\alpha}_6} \equiv & - Y_3^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4\tilde{\alpha}_5\tilde{\alpha}_6} - \frac{\eta}{2} Y_4^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4\tilde{\alpha}_5\tilde{\alpha}_6} \\ & - \frac{\eta}{2} Y_1^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4\tilde{\alpha}_5\tilde{\alpha}_6} - \frac{\eta^2}{6} X_{\text{III}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4\tilde{\alpha}_5\tilde{\alpha}_6}\,, \end{aligned} \tag{∞.145}$$

$$Z_{\text{IIA}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4\tilde{\alpha}_5\tilde{\alpha}_6} \equiv - Y_3^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4\tilde{\alpha}_5\tilde{\alpha}_6}\,, \tag{∞.146}$$

$$\begin{aligned} Z_{\text{IIB}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4\tilde{\alpha}_5\tilde{\alpha}_6} \equiv & - Y_3^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4\tilde{\alpha}_5\tilde{\alpha}_6} - Y_3^{\tilde{\alpha}_2\tilde{\alpha}_1\tilde{\alpha}_3\tilde{\alpha}_5\tilde{\alpha}_4\tilde{\alpha}_6} - Y_3^{\tilde{\alpha}_1\tilde{\alpha}_3\tilde{\alpha}_2\tilde{\alpha}_4\tilde{\alpha}_6\tilde{\alpha}_5} \\ & - \eta Y_4^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4\tilde{\alpha}_5\tilde{\alpha}_6} - \eta Y_1^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4\tilde{\alpha}_5\tilde{\alpha}_6}\,, \end{aligned} \tag{∞.147}$$

making use of our previous shorthand tensors (∞.135) and (∞.138)–(∞.140).[21] These **algorithm projectors**, (∞.142)–(∞.147), serve to project the initial training error onto two different combinations of the dNTK, two different combinations of the first ddNTK, and two other different combinations of the second ddNTK, all according to the details of the gradient descent algorithm. As a final sanity check, note that as for inputs in the training set, $\delta \to \tilde{\alpha} \in \mathcal{A}$, the quantities in the square brackets in the finite-width solution (∞.141) each vanish, and we recover our fully-trained condition (∞.66).

Before we retire this subsection, let us elaborate on the algorithm dependence. First, note that our two-update solution (∞.79) has the same form as the gradient-descent solution (∞.141), but with different algorithm projectors:

$$Z_{\text{B}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4} \equiv \frac{1}{2} \widetilde{H}^{\tilde{\alpha}_1\tilde{\alpha}_3} \widetilde{H}^{\tilde{\alpha}_2\tilde{\alpha}_4}\,, \qquad Z_{\text{IB}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4} \equiv -\frac{1}{6} \widetilde{H}^{\tilde{\alpha}_1\tilde{\alpha}_4} \widetilde{H}^{\tilde{\alpha}_2\tilde{\alpha}_5} \widetilde{H}^{\tilde{\alpha}_3\tilde{\alpha}_6}\,, \tag{∞.148}$$

and all the others vanishing. Clearly these algorithms have very different inductive biases! Second, we can study the ODE limit of the dynamics by taking $\eta \to 0$, cf. footnote 17: in this case, we see that the ODE dynamics have a solution given by

$$Z_{\text{A}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4} = Z_{\text{B}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4} \equiv Y_2^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4}\,, \tag{∞.149}$$

$$Z_{\text{IA}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4\tilde{\alpha}_5\tilde{\alpha}_6} = Z_{\text{IB}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4\tilde{\alpha}_5\tilde{\alpha}_6} = Z_{\text{IIA}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4\tilde{\alpha}_5\tilde{\alpha}_6} \equiv -Y_3^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4\tilde{\alpha}_5\tilde{\alpha}_6} \tag{∞.150}$$

$$Z_{\text{IIB}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4\tilde{\alpha}_5\tilde{\alpha}_6} \equiv -Y_3^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4\tilde{\alpha}_5\tilde{\alpha}_6} - Y_3^{\tilde{\alpha}_2\tilde{\alpha}_1\tilde{\alpha}_3\tilde{\alpha}_5\tilde{\alpha}_4\tilde{\alpha}_6} - Y_3^{\tilde{\alpha}_1\tilde{\alpha}_3\tilde{\alpha}_2\tilde{\alpha}_4\tilde{\alpha}_6\tilde{\alpha}_5}\,, \tag{∞.151}$$

where $Y_2^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4}$ and $Y_3^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4\tilde{\alpha}_5\tilde{\alpha}_6}$ are given by (∞.138) and (∞.139), respectively, and the inverting tensor $X_{\text{II}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4}$, ($\infty$.107), now satisfies

$$\sum_{\tilde{\alpha}_3,\tilde{\alpha}_4 \in \mathcal{A}} X_{\text{II}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4} \left(\widetilde{H}_{\tilde{\alpha}_3\tilde{\alpha}_5} \delta_{\tilde{\alpha}_4\tilde{\alpha}_6} + \delta_{\tilde{\alpha}_3\tilde{\alpha}_5} \widetilde{H}_{\tilde{\alpha}_4\tilde{\alpha}_6} \right) = \delta^{\tilde{\alpha}_1}_{\ \tilde{\alpha}_5} \delta^{\tilde{\alpha}_2}_{\ \tilde{\alpha}_6}\,, \tag{∞.152}$$

[21] Note that for the last of these tensors, (∞.147), in order to coax the various contributions in our solution (∞.141) into the proper form, we used the symmetry of $\widehat{\text{dd}_{\text{II}} H}_{ij_1j_2j_3;\delta\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3}$ and relabeled various dummy sample indices.

and the other inverting tensor $X_{\text{II}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4\tilde{\alpha}_5\tilde{\alpha}_6}$, ($\infty$.110), now satisfies

$$\begin{aligned} &\delta^{\tilde{\alpha}_1}_{\tilde{\alpha}_7}\delta^{\tilde{\alpha}_2}_{\tilde{\alpha}_8}\delta^{\tilde{\alpha}_3}_{\tilde{\alpha}_9} \qquad (\infty.153)\\ &= \sum_{\tilde{\alpha}_4,\tilde{\alpha}_5,\tilde{\alpha}_6} X_{\text{III}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4\tilde{\alpha}_5\tilde{\alpha}_6}\Big(\widetilde{H}_{\tilde{\alpha}_4\tilde{\alpha}_7}\delta_{\tilde{\alpha}_5\tilde{\alpha}_8}\delta_{\tilde{\alpha}_6\tilde{\alpha}_9} + \delta_{\tilde{\alpha}_4\tilde{\alpha}_7}\widetilde{H}_{\tilde{\alpha}_5\tilde{\alpha}_8}\delta_{\tilde{\alpha}_6\tilde{\alpha}_9} + \delta_{\tilde{\alpha}_4\tilde{\alpha}_7}\delta_{\tilde{\alpha}_5\tilde{\alpha}_8}\widetilde{H}_{\tilde{\alpha}_6\tilde{\alpha}_9}\Big)\,. \end{aligned}$$

This entirely captures the difference between gradient flow and gradient descent for these fully-trained networks. In general, we conjecture that for finite-width networks, at leading order the fully-trained solution takes the universal form of (∞.141), with all of the *algorithm dependence* encoded by the six *algorithm projectors*: $Z_{\text{A}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4}$, $Z_{\text{B}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4}$, $Z_{\text{IA}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4\tilde{\alpha}_5\tilde{\alpha}_6}$, $Z_{\text{IB}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4\tilde{\alpha}_5\tilde{\alpha}_6}$, $Z_{\text{IIA}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4\tilde{\alpha}_5\tilde{\alpha}_6}$, and $Z_{\text{IIB}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4\tilde{\alpha}_5\tilde{\alpha}_6}$.[22]

Finally, let's understand what is meant by the remaining error in our solution (∞.141). It is not really the error of an actual network that is instantiated and then fully trained through many many gradient-descent steps, but instead it is the error in our effective description of such a particular fully-trained network. Of course, our effective theory formalism can compute higher-order corrections, if they're of interest. However, the leading-order finite-width corrections should really be sufficient for most cases: for instance, for a network of depth $L = 10$ layers and of hidden-layer width $n = 100$ neurons each, our effective description will only be off by $\sim (L/n)^2 = 1\%$.

Furthermore, for any theoretical analysis, the main qualitative difference in the solution appears when going from infinite width to finite width, as we go from a free theory to an interacting theory and from linear dynamics to nonlinear dynamics. Thus, the effective theory that gave us the solution (∞.141) really is "as simple ... as possible" while still providing an extremely accurate description of real deep learning models.

∞.2.3 Prediction at Finite Width

Having solved the training dynamics in two different ways, we can now rather generally study the predictions of our networks on novel inputs $x_{\dot{\beta}}$ from the test set $\dot{\beta} \in \mathcal{B}$.

At finite width, the predictions of a fully-trained network are universally governed

[22] More precisely, we conjecture that our conjecture, as stated, holds for the MSE loss. For the cross-entropy loss the solution will take a slightly different form, but with a similar partition into an algorithm-independent part and an algorithm-dependent part described by similar algorithm projectors.

by the stochastic equation

$$
\begin{aligned}
&z_{i;\dot\beta}(t=T) \qquad\qquad (\infty.154)\\
=&z_{i;\dot\beta} - \sum_{\tilde\alpha_1,\tilde\alpha_2} H_{\dot\beta\tilde\alpha_1} H^{\tilde\alpha_1\tilde\alpha_2}(z_{i;\tilde\alpha_2} - y_{i;\tilde\alpha_2})\\
&+ \sum_{j,\tilde\alpha_1,\tilde\alpha_2} \left[\widehat{\Delta H}_{ij;\dot\beta\tilde\alpha_1} - \sum_{\tilde\alpha_3,\tilde\alpha_4\in\mathcal{A}} H_{\dot\beta\tilde\alpha_3}\widetilde{H}^{\tilde\alpha_3\tilde\alpha_4}\widehat{\Delta H}_{ij;\tilde\alpha_4\tilde\alpha_1}\right]\widetilde{H}^{\tilde\alpha_1\tilde\alpha_2}(z_{j;\tilde\alpha_2} - y_{j;\tilde\alpha_2})\\
&- \sum_{\substack{j,k\\ \tilde\alpha_1,\ldots,\tilde\alpha_4}} \left[\widehat{\Delta H}_{ij;\dot\beta\tilde\alpha_1} - \sum_{\tilde\alpha_5,\tilde\alpha_6} H_{\dot\beta\tilde\alpha_5}\widetilde{H}^{\tilde\alpha_5\tilde\alpha_6}\widehat{\Delta H}_{ij;\tilde\alpha_6\tilde\alpha_1}\right]\widetilde{H}^{\tilde\alpha_1\tilde\alpha_2}\widehat{\Delta H}_{jk;\tilde\alpha_2\tilde\alpha_3}\widetilde{H}^{\tilde\alpha_3\tilde\alpha_4}(z_{k;\tilde\alpha_4} - y_{k;\tilde\alpha_4})\\
&+ \sum_{j_1,j_2,\tilde\alpha_1,\tilde\alpha_2,\tilde\alpha_3,\tilde\alpha_4} \left[\widehat{\mathrm{d}H}_{j_1ij_2;\tilde\alpha_1\dot\beta\tilde\alpha_2} - \sum_{\tilde\alpha_5,\tilde\alpha_6} H_{\dot\beta\tilde\alpha_5}\widetilde{H}^{\tilde\alpha_5\tilde\alpha_6}\widehat{\mathrm{d}H}_{j_1ij_2;\tilde\alpha_1\tilde\alpha_6\tilde\alpha_2}\right]\\
&\qquad\qquad \times Z_{\mathrm{A}}^{\tilde\alpha_1\tilde\alpha_2\tilde\alpha_3\tilde\alpha_4}\,(z_{j_1;\tilde\alpha_3} - y_{j_1;\tilde\alpha_3})\,(z_{j_2;\tilde\alpha_4} - y_{j_2;\tilde\alpha_4})\\
&+ \sum_{j_1,j_2,\tilde\alpha_1,\tilde\alpha_2,\tilde\alpha_3,\tilde\alpha_4} \left[\widehat{\mathrm{d}H}_{ij_1j_2;\dot\beta\tilde\alpha_1\tilde\alpha_2} - \sum_{\tilde\alpha_5,\tilde\alpha_6} H_{\dot\beta\tilde\alpha_5}\widetilde{H}^{\tilde\alpha_5\tilde\alpha_6}\widehat{\mathrm{d}H}_{ij_1j_2;\tilde\alpha_6\tilde\alpha_1\tilde\alpha_2}\right]\\
&\qquad\qquad \times Z_{\mathrm{B}}^{\tilde\alpha_1\tilde\alpha_2\tilde\alpha_3\tilde\alpha_4}\,(z_{j_1;\tilde\alpha_3} - y_{j_1;\tilde\alpha_3})\,(z_{j_2;\tilde\alpha_4} - y_{j_2;\tilde\alpha_4})\\
&+ \sum_{\substack{j_1,j_2,j_3,\\ \tilde\alpha_1,\tilde\alpha_2,\tilde\alpha_3,\tilde\alpha_4,\tilde\alpha_5,\tilde\alpha_6}} \left[\widehat{\mathrm{dd_I}H}_{j_1ij_2j_3;\tilde\alpha_1\dot\beta\tilde\alpha_2\tilde\alpha_3} - \sum_{\tilde\alpha_7,\tilde\alpha_8} H_{\dot\beta\tilde\alpha_7}\widetilde{H}^{\tilde\alpha_7\tilde\alpha_8}\widehat{\mathrm{dd_I}H}_{j_1ij_2j_3;\tilde\alpha_1\tilde\alpha_8\tilde\alpha_2\tilde\alpha_3}\right]\\
&\qquad\qquad \times Z_{\mathrm{IA}}^{\tilde\alpha_1\tilde\alpha_2\tilde\alpha_3\tilde\alpha_4\tilde\alpha_5\tilde\alpha_6}\,(z_{j_1;\tilde\alpha_4} - y_{j_1;\tilde\alpha_4})\,(z_{j_2;\tilde\alpha_5} - y_{j_2;\tilde\alpha_5})\,(z_{j_3;\tilde\alpha_6} - y_{j_3;\tilde\alpha_6})\\
&+ \sum_{\substack{j_1,j_2,j_3,\\ \tilde\alpha_1,\tilde\alpha_2,\tilde\alpha_3,\tilde\alpha_4,\tilde\alpha_5,\tilde\alpha_6}} \left[\widehat{\mathrm{dd_I}H}_{ij_1j_2j_3;\dot\beta\tilde\alpha_1\tilde\alpha_2\tilde\alpha_3} - \sum_{\tilde\alpha_7,\tilde\alpha_8} H_{\dot\beta\tilde\alpha_7}\widetilde{H}^{\tilde\alpha_7\tilde\alpha_8}\widehat{\mathrm{dd_I}H}_{ij_1j_2j_3;\tilde\alpha_8\tilde\alpha_1\tilde\alpha_2\tilde\alpha_3}\right]\\
&\qquad\qquad \times Z_{\mathrm{IB}}^{\tilde\alpha_1\tilde\alpha_2\tilde\alpha_3\tilde\alpha_4\tilde\alpha_5\tilde\alpha_6}\,(z_{j_1;\tilde\alpha_4} - y_{j_1;\tilde\alpha_4})\,(z_{j_2;\tilde\alpha_5} - y_{j_2;\tilde\alpha_5})\,(z_{j_3;\tilde\alpha_6} - y_{j_3;\tilde\alpha_6})\\
&+ \sum_{\substack{j_1,j_2,j_3,\\ \tilde\alpha_1,\tilde\alpha_2,\tilde\alpha_3,\tilde\alpha_4,\tilde\alpha_5,\tilde\alpha_6}} \left[\widehat{\mathrm{dd_{II}}H}_{j_1j_2ij_3;\tilde\alpha_1\tilde\alpha_2\dot\beta\tilde\alpha_3} - \sum_{\tilde\alpha_7,\tilde\alpha_8} H_{\dot\beta\tilde\alpha_7}\widetilde{H}^{\tilde\alpha_7\tilde\alpha_8}\widehat{\mathrm{dd_{II}}H}_{j_1j_2ij_3;\tilde\alpha_1\tilde\alpha_2\tilde\alpha_8\tilde\alpha_3}\right]\\
&\qquad\qquad \times Z_{\mathrm{IIA}}^{\tilde\alpha_1\tilde\alpha_2\tilde\alpha_3\tilde\alpha_4\tilde\alpha_5\tilde\alpha_6}\,(z_{j_1;\tilde\alpha_4} - y_{j_1;\tilde\alpha_4})\,(z_{j_2;\tilde\alpha_5} - y_{j_2;\tilde\alpha_5})\,(z_{j_3;\tilde\alpha_6} - y_{j_3;\tilde\alpha_6})\\
&+ \sum_{\substack{j_1,j_2,j_3,\\ \tilde\alpha_1,\tilde\alpha_2,\tilde\alpha_3,\tilde\alpha_4,\tilde\alpha_5,\tilde\alpha_6}} \left[\widehat{\mathrm{dd_{II}}H}_{ij_1j_2j_3;\dot\beta\tilde\alpha_1\tilde\alpha_2\tilde\alpha_3} - \sum_{\tilde\alpha_7,\tilde\alpha_8} H_{\dot\beta\tilde\alpha_7}\widetilde{H}^{\tilde\alpha_7\tilde\alpha_8}\widehat{\mathrm{dd_{II}}H}_{ij_1j_2j_3;\tilde\alpha_8\tilde\alpha_1\tilde\alpha_2\tilde\alpha_3}\right]\\
&\qquad\qquad \times Z_{\mathrm{IIB}}^{\tilde\alpha_1\tilde\alpha_2\tilde\alpha_3\tilde\alpha_4\tilde\alpha_5\tilde\alpha_6}\,(z_{j_1;\tilde\alpha_4} - y_{j_1;\tilde\alpha_4})\,(z_{j_2;\tilde\alpha_5} - y_{j_2;\tilde\alpha_5})\,(z_{j_3;\tilde\alpha_6} - y_{j_3;\tilde\alpha_6})\\
&+ O\Big(\frac{1}{n^2}\Big)\,.
\end{aligned}
$$

This formula could boarder-line be fit on a T-shirt. For this expression, we've expanded the complete inverse of the stochastic NTK sub-tensor as per (∞.71); in particular,

the second line is more or less the infinite-width kernel prediction (10.39), and the terms on the third and fourth lines are finite-width corrections due to NTK fluctuations. Further, the algorithm projectors (∞.142)–(∞.147) contain all the finite-width algorithm dependence of the solution, and thus this general solution (∞.154) can describe both our explicit solutions, whether we train in two steps (∞.79), in many many steps (∞.141), or with any other choice of optimization algorithm that uses the MSE loss.

Furthermore, this solution (∞.154) describes the predictions of a *particular* network from our ensemble: the instantiation-to-instantiation difference is encoded in the particular initial network output, z, the NTK fluctuation, $\widehat{\Delta H}$, the dNTK, $\widehat{\mathrm{d}H}$, the first ddNTK, $\widehat{\mathrm{dd}_{\mathrm{I}}H}$, and the second ddNTK, $\widehat{\mathrm{dd}_{\mathrm{II}}H}$. Since we know explicitly the statistics of these variables at initialization, we can also analyze the statistics of the fully-trained distribution in full. With an eye towards a discussion of the depth dependence of these statistics, we'll now revive the layer indices.

First and foremost, the mean prediction is given by

$$\begin{aligned} m_{i;\dot\beta} \equiv& \mathbb{E}\left[z^{(L)}_{i;\dot\beta}(T)\right] \qquad (\infty.155)\\ =& m^{\text{NTK}}_{i;\dot\beta} + \frac{1}{n_{L-1}}\left(m^{\Delta\text{NTK}}_{i;\dot\beta} + m^{\text{dNTK}}_{i;\dot\beta} + m^{\text{ddNTK-I}}_{i;\dot\beta} + m^{\text{ddNTK-II}}_{i;\dot\beta}\right)\\ &- \frac{1}{n_{L-1}}\sum_{\tilde\alpha_1,\tilde\alpha_2} H^{(L)}_{\dot\beta\tilde\alpha_1} H^{\tilde\alpha_1\tilde\alpha_2}_{(L)}\left(m^{\Delta\text{NTK}}_{i;\tilde\alpha_2} + m^{\text{dNTK}}_{i;\tilde\alpha_2} + m^{\text{ddNTK-I}}_{i;\tilde\alpha_2} + m^{\text{ddNTK-II}}_{i;\tilde\alpha_2}\right) + O\left(\frac{1}{n^2}\right), \end{aligned}$$

where the first term is the (neural tangent) kernel prediction

$$m^{\text{NTK}}_{i;\dot\beta} \equiv \sum_{\tilde\alpha_1,\tilde\alpha_2} H^{(L)}_{\dot\beta\tilde\alpha_1} H^{\tilde\alpha_1\tilde\alpha_2}_{(L)} y_{i;\tilde\alpha_2}\,, \qquad (\infty.156)$$

and the four other terms come from the leading-order finite-width corrections. Specifically, *(i)* the fluctuation in the NTK gives

$$m^{\Delta\text{NTK}}_{i;\delta} = \sum_{\tilde\alpha_1,\ldots,\tilde\alpha_4}\left(A^{(L)}_{(\delta\tilde\alpha_1)(\tilde\alpha_2\tilde\alpha_3)} + B^{(L)}_{\delta\tilde\alpha_2\tilde\alpha_1\tilde\alpha_3} + n_L B^{(L)}_{\delta\tilde\alpha_3\tilde\alpha_1\tilde\alpha_2}\right)\widetilde{H}^{\tilde\alpha_1\tilde\alpha_2}_{(L)}\widetilde{H}^{\tilde\alpha_3\tilde\alpha_4}_{(L)} y_{i;\tilde\alpha_4} \qquad (\infty.157)$$

where we used (8.82) to evaluate the NTK variance in terms of our decomposition into tensors $A^{(L)}$ and $B^{(L)}$; *(ii)* the dNTK gives

$$\begin{aligned} m^{\text{dNTK}}_{i;\delta} = -\sum_{\tilde\alpha_1,\ldots,\tilde\alpha_4}\Bigg[& 2\left(P^{(L)}_{\delta\tilde\alpha_1\tilde\alpha_2\tilde\alpha_3} + Q^{(L)}_{\delta\tilde\alpha_1\tilde\alpha_2\tilde\alpha_3} + n_L Q^{(L)}_{\delta\tilde\alpha_2\tilde\alpha_1\tilde\alpha_3}\right) Z^{\tilde\alpha_1\tilde\alpha_2\tilde\alpha_3\tilde\alpha_4}_{\text{B}} \qquad (\infty.158)\\ &+ \left(n_L P^{(L)}_{\tilde\alpha_1\delta\tilde\alpha_2\tilde\alpha_3} + Q^{(L)}_{\tilde\alpha_1\delta\tilde\alpha_2\tilde\alpha_3} + Q^{(L)}_{\tilde\alpha_1\tilde\alpha_2\delta\tilde\alpha_3}\right) Z^{\tilde\alpha_1\tilde\alpha_2\tilde\alpha_3\tilde\alpha_4}_{\text{A}}\\ &+ \left(P^{(L)}_{\tilde\alpha_1\delta\tilde\alpha_2\tilde\alpha_3} + n_L Q^{(L)}_{\tilde\alpha_1\delta\tilde\alpha_2\tilde\alpha_3} + Q^{(L)}_{\tilde\alpha_1\tilde\alpha_2\delta\tilde\alpha_3}\right) Z^{\tilde\alpha_1\tilde\alpha_2\tilde\alpha_4\tilde\alpha_3}_{\text{A}}\Bigg] y_{i;\tilde\alpha_4}\,, \end{aligned}$$

where we used (11.42) to evaluate the dNTK-preactivation cross correlators in terms of

our decomposition into tensors $P^{(L)}$ and $Q^{(L)}$; *(iii)* the first ddNTK gives

$$\begin{aligned} & m_{i;\delta}^{\text{ddNTK-I}} \qquad (\infty.159) \\ = & -\sum_{\tilde{\alpha}_1,\ldots,\tilde{\alpha}_6} \Big[R^{(L)}_{\delta\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3} \Big(Z_{\text{IB}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4\tilde{\alpha}_5\tilde{\alpha}_6} + Z_{\text{IB}}^{\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_1\tilde{\alpha}_5\tilde{\alpha}_6\tilde{\alpha}_4} + Z_{\text{IB}}^{\tilde{\alpha}_3\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_6\tilde{\alpha}_4\tilde{\alpha}_5} \Big) \\ & \qquad + R^{(L)}_{\tilde{\alpha}_1\delta\tilde{\alpha}_2\tilde{\alpha}_3} Z_{\text{IA}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4\tilde{\alpha}_5\tilde{\alpha}_6} + R^{(L)}_{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\delta} Z_{\text{IA}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_5\tilde{\alpha}_6\tilde{\alpha}_4} + R^{(L)}_{\tilde{\alpha}_1\tilde{\alpha}_3\delta\tilde{\alpha}_2} Z_{\text{IA}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_6\tilde{\alpha}_4\tilde{\alpha}_5} \Big] \\ & \qquad\qquad \times \Big[y_{i;\tilde{\alpha}_4} \Big(\sum_j y_{j;\tilde{\alpha}_5} y_{j;\tilde{\alpha}_6} + n_L K^{(L)}_{\tilde{\alpha}_5\tilde{\alpha}_6} \Big) + y_{i;\tilde{\alpha}_5} K^{(L)}_{\tilde{\alpha}_6\tilde{\alpha}_4} + y_{i;\tilde{\alpha}_6} K^{(L)}_{\tilde{\alpha}_4\tilde{\alpha}_5} \Big] \end{aligned}$$

where we used $(\infty.11)$ to evaluate the first ddNTK mean in terms of decomposition into the tensor $R^{(L)}$; and *(iv)* the second ddNTK gives

$$\begin{aligned} & m_{i;\delta}^{\text{ddNTK-II}} \qquad (\infty.160) \\ = & -\sum_{\tilde{\alpha}_1,\ldots,\tilde{\alpha}_6} \Big[S^{(L)}_{\delta\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3} Z_{\text{IIB}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4\tilde{\alpha}_5\tilde{\alpha}_6} + T^{(L)}_{\delta\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3} Z_{\text{IIB}}^{\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_1\tilde{\alpha}_5\tilde{\alpha}_6\tilde{\alpha}_4} + U^{(L)}_{\delta\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3} Z_{\text{IIB}}^{\tilde{\alpha}_3\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_6\tilde{\alpha}_4\tilde{\alpha}_5} \\ & \qquad + S^{(L)}_{\tilde{\alpha}_1\tilde{\alpha}_2\delta\tilde{\alpha}_3} Z_{\text{IIA}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_5\tilde{\alpha}_6\tilde{\alpha}_4} + T^{(L)}_{\tilde{\alpha}_1\delta\tilde{\alpha}_3\tilde{\alpha}_2} Z_{\text{IIA}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4\tilde{\alpha}_5\tilde{\alpha}_6} + U^{(L)}_{\tilde{\alpha}_1\tilde{\alpha}_3\tilde{\alpha}_2\delta} Z_{\text{IIA}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_6\tilde{\alpha}_4\tilde{\alpha}_5} \Big] \\ & \qquad\qquad \times \Big[y_{i;\tilde{\alpha}_4} \Big(\sum_j y_{j;\tilde{\alpha}_5} y_{j;\tilde{\alpha}_6} + n_L K^{(L)}_{\tilde{\alpha}_5\tilde{\alpha}_6} \Big) + y_{i;\tilde{\alpha}_5} K^{(L)}_{\tilde{\alpha}_6\tilde{\alpha}_4} + y_{i;\tilde{\alpha}_6} K^{(L)}_{\tilde{\alpha}_4\tilde{\alpha}_5} \Big] \end{aligned}$$

where we used $(\infty.12)$ to evaluate the second ddNTK mean in terms of decomposition into the tensors $S^{(L)}$, $T^{(L)}$, and $U^{(L)}$.

Interestingly, we see that not only does the mean prediction depend on the NTK differentials – as we expect, given the nontrivial representation learning at finite width – but also it depends on the NTK variance as well, cf. $(\infty.157)$. This is natural as each network fits the training data with its own *particular* NTK, and so the resulting fully-trained particular output $(\infty.154)$ depends on the NTK fluctuation, as we've already emphasized. In general, it is really important to understand the tradeoffs between both contributions, and in the next subsubsection we will return to comment on this interplay between random fluctuations and directed representation learning in the overall ensemble.

In addition, looking at the terms that are cubic in $y_{i;\tilde{\alpha}}$ in the contributions from the ddNTKs, $(\infty.159)$ and $(\infty.160)$, we see that the j-th component of the observed outputs influences the i-th component of the mean prediction for $i \neq j$. Just as we saw for the mean prediction of Bayesian inference, (6.88), this is one consequence of the wiring property of finite-width neural networks. To see another effect of this wiring property, let's consider the covariance:

$$\text{Cov}\Big[z^{(L)}_{i_1;\dot{\beta}_1}(T),\, z^{(L)}_{i_2;\dot{\beta}_2}(T) \Big] \equiv \mathbb{E}\Big[z^{(L)}_{i_1;\dot{\beta}_1}(T) z^{(L)}_{i_2;\dot{\beta}_2}(T) \Big] - \mathbb{E}\Big[z^{(L)}_{i_1;\dot{\beta}_1}(T) \Big] \mathbb{E}\Big[z^{(L)}_{i_2;\dot{\beta}_2}(T) \Big] \ . \quad (\infty.161)$$

While we won't print this quantity in full – the full expression doesn't really play nicely with the constraints of the page – you can easily extract insight by considering some

specific contributions. For instance, we can see the imprint of output-component wiring by looking at the following contribution to the covariance,

$$\begin{aligned}
&\sum_{\substack{j_1,j_2\\ \tilde{\alpha}_1,\ldots,\tilde{\alpha}_4}} \mathbb{E}\left[z^{(L)}_{i_2;\dot{\beta}_2}\widehat{\mathrm{d}H}^{(L)}_{i_1 j_1 j_2;\dot{\beta}_1\tilde{\alpha}_1\tilde{\alpha}_2}\right] Z_{\mathrm{B}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4}\mathbb{E}\left[\left(z^{(L)}_{j_1;\tilde{\alpha}_3}-y_{j_1;\tilde{\alpha}_3}\right)\left(z^{(L)}_{j_2;\tilde{\alpha}_4}-y_{j_2;\tilde{\alpha}_4}\right)\right]\\
=&\frac{1}{n_{L-1}}\delta_{i_1 i_2}\sum_{\tilde{\alpha}_1,\ldots,\tilde{\alpha}_4}\Bigg[\left(n_L P^{(L)}_{\dot{\beta}_1\tilde{\alpha}_1\tilde{\alpha}_2\dot{\beta}_2}+Q^{(L)}_{\dot{\beta}_1\tilde{\alpha}_1\tilde{\alpha}_2\dot{\beta}_2}+Q^{(L)}_{\dot{\beta}_1\tilde{\alpha}_2\tilde{\alpha}_1\dot{\beta}_2}\right)G^{(L)}_{\tilde{\alpha}_3\tilde{\alpha}_4}\\
&\qquad\qquad +P^{(L)}_{\dot{\beta}_1\tilde{\alpha}_1\tilde{\alpha}_2\dot{\beta}_2}\Big(\sum_j y_{j;\tilde{\alpha}_3}y_{j;\tilde{\alpha}_4}\Big)\Bigg]\widetilde{Y}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4}\\
&+\frac{1}{n_{L-1}}\sum_{\tilde{\alpha}_1,\ldots,\tilde{\alpha}_4}Q^{(L)}_{\dot{\beta}_1\tilde{\alpha}_1\tilde{\alpha}_2\dot{\beta}_2}Z_{\mathrm{B}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4}\left(y_{i_1;\tilde{\alpha}_3}y_{i_2;\tilde{\alpha}_4}+y_{i_1;\tilde{\alpha}_4}y_{i_2;\tilde{\alpha}_3}\right), \qquad (\infty.162)
\end{aligned}$$

which comes from the cross correlation between a factor of $z^{(L)}_{i_2;\dot{\beta}_2}$ from $z^{(L)}_{i_2;\dot{\beta}_2}(T)$ and one of the dNTK terms from $z^{(L)}_{i_1;\dot{\beta}_1}(T)$ that involves the algorithm projector $Z_{\mathrm{B}}^{\tilde{\alpha}_1\tilde{\alpha}_2\tilde{\alpha}_3\tilde{\alpha}_4}$. In particular, we see that wiring is exhibited in the last term on the final line when the true outputs have nonzero components for both i_1 and i_2. In this case, this correlation (∞.162) implies that the test-set predictions for $z^{(L)}_{i_1;\dot{\beta}_1}(T)$ can be shifted given $z^{(L)}_{i_2;\dot{\beta}_2}(T)$, for $i_1 \neq i_2$: cf. our related discussion of Hebbian learning in terms of the *fire-together* inductive bias in the finite-width prior in §6.4.1 and then our following discussion of how that leads to the posterior distribution's *wiring together* in §6.4.2. Here, the presence of this wiring in the covariance of the solution means that this contribution to wiring occurs differently for each network in the ensemble: the fluctuations from instantiation to instantiation of the parameters at initialization break the permutation symmetry among the output neurons. This type of wiring suggests that each network is able to use the fluctuations between the different initial output components to its advantage in order to correlate such components together over the course of learning.

More broadly, unlike the kernel prediction at infinite width (10.39), the prediction at finite width (∞.154) now has non-Gaussian statistics. In particular, since finite-width prediction is a nontrivial functional of the network output, the NTK, the dNTK, and the ddNTKs – all at initialization – and since we know that the joint distribution of those quantities $p\Big(z^{(L)},\, \widehat{H}^{(L)},\, \widehat{\mathrm{d}H}^{(L)},\, \widehat{\mathrm{dd}_{\mathrm{I}}H}^{(L)},\, \widehat{\mathrm{dd}_{\mathrm{II}}H}^{(L)}\Big|\mathcal{D}\Big)$ is a *nearly-Gaussian distribution*, then so is the fully-trained distribution $p\Big(z^{(L)}(T)\,\Big|\mathcal{D}\Big)$. In particular, there are nontrivial higher-point connected correlators; the explicit expressions of such correlators are challenging to display in any media format, though all the information needed to do so is contained in (∞.154), and it's not that hard to zero in on any particular term of interest. The information carried in such correlators probably contains useful insight into some of the behavior of fully-trained networks and is likely worth further consideration.

Generalization at Finite Width

Having discussed many of the qualitative differences between the finite-width prediction (∞.154) and the infinite-width kernel prediction (10.39), now let's make some quantitative statements. In order to get a high-level understanding of how generalization is modified at finite width as compared to our extensive infinite-width analysis in §10.3, we need to determine the size of the relative importance of the finite-width corrections to our predictions, namely how these corrections depend on the widths of the hidden layers n_ℓ and the depth of the network L. In particular, we want to understand corrections to our generalized bias-variance tradeoff (10.52) at finite width.

Let's start by considering the bias term, $m_{i;\dot\beta} - y_{i;\dot\beta}$, for which we just need to look at the mean prediction $m_{i;\dot\beta}$ in (∞.155). In particular, we need to compare the first term, (∞.156), which more or less corresponds to the infinite-width kernel prediction $m^{\infty}_{i;\dot\beta}$ – up to the subleading and unimportant correction to the NTK mean – against the other set of the finite-width contributions to the mean, (∞.157)–(∞.160).

Now, among the finite-width contributions in (∞.155), the terms inside the first set of large parentheses have a sample index corresponding to a test input, $\dot\beta$, in each of the tensors $A^{(L)}$, $B^{(L)}$, $P^{(L)}$, $Q^{(L)}$, $R^{(L)}$, $S^{(L)}$, $T^{(L)}$, and $U^{(L)}$. Thus, even for a training set with one training sample as we studied in §10.3.1, the particular details of these terms require an understanding of asymptotic multiple-input solutions of the recursions for the NTK variance, the preactivation-dNTK cross correlation, and the ddNTK mean; such an analysis is kind of annoying and was previously left as an adventure for thrill seekers, with a brief instruction manual for those interested buried in footnote 8 of §11.3. Unfortunately, we have no plans here to update that manual any further, and we don't expect to miss much by not doing so.[23]

In contrast, the terms inside the second set of large parentheses in (∞.155) can be analyzed with the results we already have in hand. Thus, to nonlinearly gain a large amount of intuition with a small amount of effort, let's compare the infinite-width term (∞.156) against only this last set of terms. In particular, since both of these terms are preceded by a common prefactor $\sum_{\tilde\alpha_1,\tilde\alpha_2} H^{(L)}_{\dot\beta\tilde\alpha_1} H^{\tilde\alpha_1\tilde\alpha_2}_{(L)}$ – whose effect was analyzed in §10.3 – we simply need to understand the depth and width dependence of the tensors inside the second set of brackets in (∞.155). Here we'll evaluate these tensors for a single training set input $x_{\tilde\alpha} = x$, dropping the training sample indices for the rest of this analysis for notational simplicity.

To see the physics, it's simplest to look at the ODE limit of many many steps of gradient descent, $\eta \searrow 0$, for which the algorithm projectors take particularly simple

[23] In particular, we expect that this first set of terms will behave qualitatively similar to the second set of terms that we will discuss in the next paragraph, though the particular order-one-level details may differ.

form,

$$Z_{\text{A}} = Z_{\text{B}} = \frac{1}{2\left(\widetilde{H}^{(L)}\right)^2}\,, \quad Z_{\text{IA}} = Z_{\text{IB}} = Z_{\text{IIA}} = \frac{-1}{6\left(\widetilde{H}^{(L)}\right)^3}\,, \quad Z_{\text{IIB}} = \frac{-1}{2\left(\widetilde{H}^{(L)}\right)^3}\,, \tag{∞.163}$$

cf. (∞.138), (∞.139), and (∞.149)–(∞.153) to derive these expressions.[24] Substituting these projectors (∞.163) into the dNTK and ddNTK contributions to the mean prediction, (∞.158)–(∞.160), and then considering the $1/n$ part of the mean prediction $m_{i;\dot{\beta}}$ (∞.155), we see that all the individual terms are proportional to one of the following dimensionless ratios:

$$\frac{A^{(L)}}{n_{L-1}\left(\widetilde{H}^{(L)}\right)^2}\,, \quad \frac{B^{(L)}}{n_{L-1}\left(\widetilde{H}^{(L)}\right)^2}\,, \quad \frac{P^{(L)}}{n_{L-1}\left(\widetilde{H}^{(L)}\right)^2}\,, \quad \frac{Q^{(L)}}{n_{L-1}\left(\widetilde{H}^{(L)}\right)^2}\,, \tag{∞.165}$$

$$\frac{R^{(L)}K^{(L)}}{n_{L-1}\left(\widetilde{H}^{(L)}\right)^3}\,, \quad \frac{S^{(L)}K^{(L)}}{n_{L-1}\left(\widetilde{H}^{(L)}\right)^3}\,, \quad \frac{T^{(L)}K^{(L)}}{n_{L-1}\left(\widetilde{H}^{(L)}\right)^3}\,, \quad \frac{U^{(L)}K^{(L)}}{n_{L-1}\left(\widetilde{H}^{(L)}\right)^3}\,.$$

Thus, these eight ratios determine the size of the finite-width effects as compared to the leading infinite-width term.[25]

Importantly, recalling our scaling laws for the NTK variance, (9.27), for the dNTK-preactivation cross correlation, (11.65), and for the dNTK-preactivation cross correlation, (∞.21) and (∞.22), we see that each of these dimensionless ratios will scale like the depth-to-width ratio L/n (except for the subdominant $U^{(L)}$ contribution). Thus, overall we should find for the finite-width corrections

$$m_{i;\dot{\beta}} - m^{\infty}_{i;\dot{\beta}} = O\left(\frac{L}{n}\right)\,, \tag{∞.166}$$

[24] Backing off the ODE limit, for many many steps of gradient descent with a finite learning rate η, the single-input dNTK algorithm projectors, (∞.142) and (∞.143), are instead given by

$$Z_{\text{A}} = \frac{1}{\left(\widetilde{H}^{(L)}\right)^2}\left(\frac{1-\eta\widetilde{H}^{(L)}}{2-\eta\widetilde{H}^{(L)}}\right)\,, \qquad Z_{\text{B}} = \frac{1}{2\left(\widetilde{H}^{(L)}\right)^2}\,. \tag{∞.164}$$

In conjunction with similar single-input limits of the ddNTK algorithm projectors, this will mean that the same set of ratios, (∞.165), determine the generalization error bias term, though now with an additional η dependence. In particular, the projector Z_{A} diverges as the global learning rate approaches from below $\eta \nearrow 2/\widetilde{H}^{(L)}$. This divergence is expected: as we noted in footnote 16, we need $||I - \eta\widehat{H}||_\infty < 1$ for the training dynamics to converge after many many steps $t \to \infty$. If you check, you'll also see that some of the ddNTK algorithm projectors have the same divergence.

[25] Incidentally, the form of the final two ratios on the first line of (∞.165) is the ultimate justification for why in (11.62) we divided the dNTK-preactivation cross-correlation tensors by two factors of the NTK mean, cf. our discussion of dimensional analysis around (11.64). Similarly the form of the four ratios on the second line of (∞.165) is the ultimate justification for the ratios (∞.21) and (∞.22). For this latter set of ratios, we should have really written $K^{(L)} + n_L^{-1}\sum_j y_j^2$ instead of just $K^{(L)}$, especially when $K^{(L)}$ behaves as a nontrivial power law in L; in such cases, we should really rescale the target output $y_{i;\tilde{\alpha}}$ by the power-law factor $L^{-p_0/2}$ as discussed around (∞.26).

where the exact order-one constant is not important. What is important is that we confirmed our expectation for the overall scaling of this correction, going as the *cutoff* of our effective theory: $r \equiv L/n$. Similarly, we could do an analysis for the variance term of the generalization error by looking at the covariance (∞.161), which will be a whole lot of work with very little payoff; such an analysis would merely confirm that the leading finite-width corrections will again be of order $O(L/n)$.

In general, given that the aspect ratio L/n controls both the fluctuations in the ensemble *and* representation learning, the optimal value of the ratio is likely nonzero but also small. In particular, representation learning is enhanced by the depth, but networks with too large a value of L/n will both have an effect on the mean prediction of the ensemble, but perhaps even more importantly lead to exponentially-large problems when working with only a *particular* network: for large enough L/n our principle of typicality can break down, and so the generalization error can begin to exhibit exponential behavior.[26]

This further explains the success of our effective theory description at $O(L/n)$: a description with vanishing L/n, i.e. the infinite-width limit, is too simple to model the properties of deep neural networks in practice; a description for larger values of L/n, i.e. a *small*-width or *overly*-deep regime that includes many higher-order corrections, describes networks that are unlikely to be trainable; a description with small but nonzero L/n, i.e. the leading finite-width effective theory accurate to $O(L/n)$, is as simple as it can be and still accurately describe the networks that work well in practice.[27]

In summary, we've seen that the leading finite-width corrections all scale exactly according to our long-standing expectations, and we've discussed at a high level the potential tradeoff of depth versus width. In principle, we could go further in our analysis, evaluating the multi-input recursions for nearby inputs, evaluating all the terms in the covariance, and finding all the $O(L/n)$ contributions to the generalization error with specific coefficients.[28] If we did this, what could it tell us? In particular, can we theoretically optimize the aspect ratio L/n for a particular activation function without any experimentation?

Unfortunately at the order we're working, we won't be able to optimize the aspect ratio L/n using the prediction formula (∞.154) from the end of training: the linear dependence of the generalization on the ratio means that its derivative is independent of the ratio; instead, we'd need to compute the higher-order corrections of order $O(L^2/n^2)$

[26] In our discussion of *fluctuations* in §5.4, we explained that too large a value of L/n may lead to a difficult fine-tuning problem in terms of getting a particular networks to behave critically.

On the one hand, the contribution of such fluctuations to the bias part of the generalization error is one way to see how the downstream effect of such fluctuations may lead to problems after training. On the other hand, the fact that fluctuations can ruin criticality for a particular network is another problem. The former problem affects the ensemble as a whole, while the latter problem affects individual networks.

[27] To better understand the scale that separates the trainable regime from the overly-deep regime, see Appendix A. For a discussion of how to extend this trainable regime to greater depths for fixed width, see our discussion of residual networks in Appendix B.

[28] You can also more easily evaluate the multi-input version of the relevant recursions numerically with this scaling in mind in order to determine the overall coefficient for a particular activation function of interest. This would be one way to analyze these solutions for the `ReLU`-like `GELU` and `SWISH` networks.

to optimize the ratio – which is hard, though at least straightforward to do in the effective theory framework we've developed.[29] At order $O(L/n)$, the best we could hope to see is whether nonzero L/n improves generalization or not by looking at the sign of its overall coefficient. Rather than going through that hassle, we'll instead get a little more mileage out of the mean prediction $m_{i;\dot{\beta}}$ by trying to understand how the *algorithm dependence* at finite width leads to additional tradeoffs within the bias term of the generalization error.

Inductive Bias of the Training Algorithm

As multiply mentioned, one of the key differences between the infinite-width and finite-width networks optimized via gradient-based learning is the *algorithm dependence* of the fully-trained solution for finite-width networks. In particular, the solution – at least for MSE losses – takes a universal form, (∞.154), with all of the dependence of the solution on the training algorithm encoded in the *algorithm projectors*, Z_{A}, Z_{B}, Z_{IA}, Z_{IB}, Z_{IIA}, and Z_{IIB}, whose functional forms we've established explicitly for the two second-order updates in (∞.148), for many many steps of gradient descent in (∞.142)–(∞.147), and for the ODE limit of gradient descent in (∞.149)–(∞.151). Through this universal projection, we now have a theoretical means of isolating the *inductive bias of the training algorithm* from all the other inductive biases that are present in a fully-trained neural network. This provides a natural way of theoretically evaluating the relative merits of different training algorithms.

One aspect that is apparent just from staring at the stochastic prediction (∞.154) is that the algorithm projectors only induce projections on the NTK-differential part of the finite-width prediction and cannot affect the NTK-fluctuation contribution. More concretely, let's continue our discussion of the bias term in the generalization error. In particular, the bias is given by the following difference:

$$m_{i;\dot{\beta}} - y_{i;\dot{\beta}}\,. \tag{∞.167}$$

On the one hand, it's clear that the NTK-variance contribution to the mean prediction, (∞.157), encoded in $A^{(L)}$ and $B^{(L)}$ is irreducible, entirely independent of the training algorithm; this irreducible NTK-variance contribution depends on the training data in a fixed way and arises due to instantiation-to-instantiation fluctuations in the NTK across different realizations of the parameters. On the other hand, the projectors always act on the NTK-differential tensors $P^{(L)}$, $Q^{(L)}$, $R^{(L)}$, $S^{(L)}$, $T^{(L)}$, and $U^{(L)}$; this means that the dNTK's and ddNTKs' effect on the network's predictions is adjustable by the training algorithm and can be tuned differently for different datasets and tasks.

[29] In §A.3, we'll compute such higher-order effects from an *information-theoretic* perspective. This analysis will give a heuristic prescription for optimizing the network's depth-to-width ratio r: we consider an auxiliary *unsupervised learning* objective thought to be beneficial for building representations and optimize the aspect ratio in terms of *that* criterion rather than the generalization error. In this setting, we're able to determine optimal aspect ratios for different activation functions by trading off leading-order effects against higher-order effects. This is somewhat similar in spirit to the way in which Newton's method trades off the leading decrease of the loss against the subleading loss increase in order to optimize the overall learning rate, cf. our discussion of Newton's method in footnote 8 of §10.2.

To reiterate our previous discussion of the tradeoff, on the one hand the NTK-fluctuation contribution is likely harmful as it leads to a breakdown of criticality for any *particular* network. On the other hand, the latter NTK-differential contribution is the ultimate source of the nontrivial representation learning at finite width and so it would be nice to make this contribution large. With these algorithm projectors, we now have direct means to enhance the latter benefit while keeping fixed the former cost.

One way of understanding these algorithm projectors is as the sample-space *dual description* of a learning algorithm, analogous to the relationship between the feature functions $\phi_j(x)$ and the kernel $k(x_{\delta_1}, x_{\delta_2})$ that we explored in §10.4 and §11.4. From the parameter-space microscopic perspective, a learning algorithm, such as gradient descent, explicitly operates on the parameters. At the end of training, all the details of the algorithm are left implicit in the trained parameters $\theta^\star$, making it difficult to understand its effect on the model's predictions. From this perspective, the algorithm is simple to define (§7.2), but decidedly difficult to analyze for general interacting machine-learning models (§∞.2.2). However, if you can analyze or *solve* a model to find its sample-space dual description, then the algorithm projectors make the influence of the learning algorithm explicit.[30]

Analogously to the (nearly-)kernel perspective on (nearly-)linear models, the appearance of the algorithm projectors in the generalization error means that we can begin to think about *engineering* them directly, either by picking them to be directly used with a prediction formula such as (∞.154), or alternatively by attempting to find the parameter-space dual of an algorithm projector. While we expect this latter task to be difficult – just as it's often hard to find the feature functions that correspond to a particular kernel – it could be very worthwhile to explore as a means of engineering the inductive bias of the training algorithm directly.[31]

In particular, we expect that this could significantly improve generalization by designing algorithms that are better tailored to the details of an architecture or on the properties of the underlying dataset or task.[32] This design process could in principle proceed as follows: *(i)* engineer a desired functional form for the algorithm projectors and then *(ii)* determine the parameter-space training algorithm that leads to the projectors having those desired properties or specific functional form. While further details of such **inverse algorithm design** is outside the scope of this book, we think that this line of *dual* analysis has the potential to unlock a much deeper understanding of the relationship among the inductive bias of the training algorithm, the inductive bias of the network architecture, and the ultimate success of the model.

[30] This is the sense in which we meant that the conditioning on the learning algorithm would be *simple* for the trained ensemble in (0.9).

[31] This can also be seen as another advantage of nonlinear or interacting machine learning models. For linear models, the solution is always independent of the details of the learning algorithm (§10.2.2).

[32] Since the algorithm projectors have sample indices, the *optimal* choice of the algorithm will in general depend on the details and structure of the training set in addition to the definition of the model.

There's No Place Where Gradient Descent = Exact Bayesian Inference

Finally, a curious reader might wonder whether the connection that we detailed in §10.2.4 between gradient descent optimization and exact Bayesian inference for the infinite-width limit persists for more realistic networks at finite width. In short, the answer is no.

In long, recall the training hyperparameter settings (10.46) and (10.48) that matched gradient-based learning with Bayesian inference at infinite width. In particular, the latter condition (10.48) required turning off any learning in the hidden layers:

$$\lambda_b^{(\ell)} = 0\,, \qquad \lambda_W^{(\ell)} = 0\,, \qquad \text{for} \quad \ell < L\,. \tag{∞.168}$$

Importantly, this makes the hidden-layer NTK vanish exactly: $\widehat{H}^{(\ell)}_{i_1 i_2;\delta_1\delta_2} = 0$, for $\ell < L$ at any finite width, cf. (8.12). Next, flip back and look at the P- and Q-recursions for the dNTK-preactivation cross-correlation tensors, (11.49) and (11.52), and then you'll probably also want to flip further back and visit the F- and B-recursions, (8.79) and (8.89), respectively. (We'll be waiting here for you when you return.) Immediately, you should notice a problem: if the hidden-layer NTK mean vanishes, then at this order $B^{(\ell)} = 0$, $F^{(\ell)} = 0$, and all this together implies that $P^{(L)} = Q^{(L)} = 0$. Thus, all the effects of the dNTK are turned off. Similarly, if you flip forth and look at the R-, S-, T-, and U-recursions, (∞.172) and (∞.174)–(∞.176), then you'll notice that all the effects of the ddNTKs are turned off as well. Thus, the mean prediction of our finite-width gradient-based-learning ensemble (∞.155) *cannot* match the exact Bayesian posterior mean at finite width (6.88).

Note that this mismatch is obvious in hindsight; by only training the last layer (∞.168), we get a linear model (10.134). In other words, since the hyperparameter choices of (∞.168) lead to a model with random features that are fixed over the course of training, there's no representation learning possible for these settings. In contrast, we found nontrivial representation learning when studying exact Bayesian inference at finite width in §6.4.3.

Ultimately, for Bayesian inference we only care about the preactivation distribution $p\Big(z^{(\ell)}\Big|\mathcal{D}\Big)$, while for gradient-based learning we need to consider the joint preactivation-NTK-dNTK-ddNTKs distribution $p\Big(z^{(L)}, \widehat{H}^{(L)}, \widehat{\mathrm{d}H}^{(L)}, \widehat{\mathrm{dd}_{\mathrm{I}}H}^{(L)}, \widehat{\mathrm{dd}_{\mathrm{II}}H}^{(L)}\Big|\mathcal{D}\Big)$, which incorporates the statistics of the derivatives of the preactivations at initialization: such derivatives of the model output are invisible to the exact Bayesian inferencer.

∞.3 RG Flow of the ddNTKs: The Full Expressions

These expressions were kind of horrible, so we decided to hide them here at the end of the chapter. As such, they are only really needed for three reasons: *(i)* to explicitly check the absence of any NTK differentials for the hyperparameter setup (∞.168) and thus confirm that the connection between gradient descent and Bayesian inference doesn't persist at finite width, *(ii)* to check the details of the depth-to-width scaling of the ddNTKs that we discussed in §∞.1, and *(iii)* to more generally evaluate the ddNTKs' contributions to

the ensemble's mean prediction, (∞.159) and (∞.160), for multiple inputs – analytically or numerically – or to compute other higher-order statistics of our stochastic prediction (∞.154).

$\widehat{\mathbf{dd_I}H}$ Stochastic Forward Equation

$$
\begin{aligned}
&\widehat{\mathrm{dd_I}H}^{(\ell+1)}_{i_0 i_1 i_2 i_3;\delta_0\delta_1\delta_2\delta_3} \qquad (\infty.169)\\
= \;& \delta_{i_0 i_1}\frac{\lambda_W^{(\ell_+1)}}{n_\ell}\sum_{j_0,j_1,j_2,j_3=1}^{n_\ell}\delta_{j_0 j_1}W^{(\ell+1)}_{i_2 j_2}W^{(\ell+1)}_{i_3 j_3}\sigma^{(\ell)}_{j_1;\delta_1}\sigma'^{(\ell)}_{j_2;\delta_2}\sigma'^{(\ell)}_{j_3;\delta_3}\\
&\qquad\times\Big[\sigma''^{(\ell)}_{j_0;\delta_0}\widehat{H}^{(\ell)}_{j_0 j_2;\delta_0\delta_2}\widehat{H}^{(\ell)}_{j_0 j_3;\delta_0\delta_3}+\sigma'^{(\ell)}_{j_0;\delta_0}\widehat{\mathrm{d}H}^{(\ell)}_{j_0 j_2 j_3;\delta_0\delta_2\delta_3}\Big]\\
&+\delta_{i_0 i_2}\frac{\lambda_W^{(\ell_+1)}}{n_\ell}\sum_{j_0,j_1,j_2,j_3=1}^{n_\ell}\delta_{j_0 j_2}W^{(\ell+1)}_{i_3 j_3}W^{(\ell+1)}_{i_1 j_1}\sigma^{(\ell)}_{j_2;\delta_2}\sigma'^{(\ell)}_{j_3;\delta_3}\sigma'^{(\ell)}_{j_1;\delta_1}\\
&\qquad\times\Big[\sigma''^{(\ell)}_{j_0;\delta_0}\widehat{H}^{(\ell)}_{j_0 j_3;\delta_0\delta_3}\widehat{H}^{(\ell)}_{j_0 j_1;\delta_0\delta_1}+\sigma'^{(\ell)}_{j_0;\delta_0}\widehat{\mathrm{d}H}^{(\ell)}_{j_0 j_3 j_1;\delta_0\delta_3\delta_1}\Big]\\
&+\delta_{i_0 i_3}\frac{\lambda_W^{(\ell_+1)}}{n_\ell}\sum_{j_0,j_1,j_2,j_3=1}^{n_\ell}\delta_{j_0 j_3}W^{(\ell+1)}_{i_1 j_1}W^{(\ell+1)}_{i_2 j_2}\sigma^{(\ell)}_{j_3;\delta_3}\sigma'^{(\ell)}_{j_1;\delta_1}\sigma'^{(\ell)}_{j_2;\delta_2}\\
&\qquad\times\Big[\sigma''^{(\ell)}_{j_0;\delta_0}\widehat{H}^{(\ell)}_{j_0 j_1;\delta_0\delta_1}\widehat{H}^{(\ell)}_{j_0 j_2;\delta_0\delta_2}+\sigma'^{(\ell)}_{j_0;\delta_0}\widehat{\mathrm{d}H}^{(\ell)}_{j_0 j_1 j_2;\delta_0\delta_1\delta_2}\Big]\\
&+\sum_{j_0,j_1,j_2,j_3=1}^{n_\ell}W^{(\ell+1)}_{i_0 j_0}W^{(\ell+1)}_{i_1 j_1}W^{(\ell+1)}_{i_2 j_2}W^{(\ell+1)}_{i_3 j_3}\sigma'^{(\ell)}_{j_1;\delta_1}\sigma'^{(\ell)}_{j_2;\delta_2}\sigma'^{(\ell)}_{j_3;\delta_3}\\
&\qquad\times\Big[\sigma'^{(\ell)}_{j_0;\delta_0}\widehat{\mathrm{dd_I}H}^{(\ell)}_{j_0 j_1 j_2 j_3;\delta_0\delta_1\delta_2\delta_3}+\sigma''^{(\ell)}_{j_0;\delta_0}\widehat{\mathrm{d}H}^{(\ell)}_{j_0 j_1 j_2;\delta_0\delta_1\delta_2}\widehat{H}^{(\ell)}_{j_0 j_3;\delta_0\delta_3}\\
&\qquad\quad+\sigma''^{(\ell)}_{j_0;\delta_0}\widehat{\mathrm{d}H}^{(\ell)}_{j_0 j_2 j_3;\delta_0\delta_2\delta_3}\widehat{H}^{(\ell)}_{j_0 j_1;\delta_0\delta_1}+\sigma''^{(\ell)}_{j_0;\delta_0}\widehat{\mathrm{d}H}^{(\ell)}_{j_0 j_3 j_1;\delta_0\delta_3\delta_1}\widehat{H}^{(\ell)}_{j_0 j_2;\delta_0\delta_2}\\
&\qquad\quad+\sigma'''^{(\ell)}_{j_0\delta_0}\widehat{H}^{(\ell)}_{j_0 j_1;\delta_0\delta_1}\widehat{H}^{(\ell)}_{j_0 j_2;\delta_0\delta_2}\widehat{H}^{(\ell)}_{j_0 j_3;\delta_0\delta_3}\Big],
\end{aligned}
$$

$\widehat{\mathrm{dd}_{\mathrm{II}}H}$ Stochastic Forward Equation

$$
\begin{aligned}
&\widehat{\mathrm{dd}_{\mathrm{II}}H}^{(\ell+1)}_{i_1i_2i_3i_4;\delta_1\delta_2\delta_3\delta_4} \qquad (\infty.170)\\
=&\delta_{i_1i_3}\delta_{i_2i_4}\left(\frac{\lambda_W^{(\ell+1)}}{n_\ell}\right)^2\sum_{j,k=1}^{n_\ell}\sigma'^{(\ell)}_{j;\delta_1}\sigma'^{(\ell)}_{k;\delta_2}\sigma^{(\ell)}_{j;\delta_3}\sigma^{(\ell)}_{k;\delta_4}\widehat{H}^{(\ell)}_{jk;\delta_1\delta_2}\\
&+\delta_{i_1i_2}\frac{\lambda_W^{(\ell+1)}}{n_\ell}\sum_{j_1,\ldots,j_4=1}^{n_\ell}\delta_{j_1j_2}W^{(\ell+1)}_{i_3j_3}W^{(\ell+1)}_{i_4j_4}\sigma'^{(\ell)}_{j_1;\delta_1}\sigma'^{(\ell)}_{j_2;\delta_2}\sigma'^{(\ell)}_{j_3;\delta_3}\sigma'^{(\ell)}_{j_4;\delta_4}\widehat{H}^{(\ell)}_{j_1j_3;\delta_1\delta_3}\widehat{H}^{(\ell)}_{j_2j_4;\delta_2\delta_4}\\
&+\delta_{i_1i_3}\frac{\lambda_W^{(\ell+1)}}{n_\ell}\sum_{j_1,\ldots,j_4=1}^{n_\ell}\delta_{j_1j_3}W^{(\ell+1)}_{i_2j_2}W^{(\ell+1)}_{i_4j_4}\sigma^{(\ell)}_{j_3;\delta_3}\sigma'^{(\ell)}_{j_4;\delta_4}\sigma'^{(\ell)}_{j_1;\delta_1}\\
&\qquad\times\left[\sigma''^{(\ell)}_{j_2;\delta_2}\widehat{H}^{(\ell)}_{j_2j_1;\delta_2\delta_1}\widehat{H}^{(\ell)}_{j_2j_4;\delta_2\delta_4}+\sigma'^{(\ell)}_{j_2;\delta_2}\widehat{\mathrm{d}H}^{(\ell)}_{j_2j_1j_4;\delta_2\delta_1\delta_4}\right]\\
&+\delta_{i_2i_4}\frac{\lambda_W^{(\ell+1)}}{n_\ell}\sum_{j_1,\ldots,j_4=1}^{n_\ell}\delta_{j_2j_4}W^{(\ell+1)}_{i_1j_1}W^{(\ell+1)}_{i_3j_3}\sigma^{(\ell)}_{j_4;\delta_4}\sigma'^{(\ell)}_{j_3;\delta_3}\sigma'^{(\ell)}_{j_2;\delta_2}\\
&\qquad\times\left[\sigma''^{(\ell)}_{j_1;\delta_1}\widehat{H}^{(\ell)}_{j_1j_2;\delta_1\delta_2}\widehat{H}^{(\ell)}_{j_1j_3;\delta_1\delta_3}+\sigma'^{(\ell)}_{j_1;\delta_1}\widehat{\mathrm{d}H}^{(\ell)}_{j_1j_2j_3;\delta_1\delta_2\delta_3}\right]\\
&+\sum_{j_1,j_2,j_3,j_4=1}^{n_\ell}W^{(\ell+1)}_{i_1j_1}W^{(\ell+1)}_{i_2j_2}W^{(\ell+1)}_{i_3j_3}W^{(\ell+1)}_{i_4j_4}\sigma'^{(\ell)}_{j_3;\delta_3}\sigma'^{(\ell)}_{j_4;\delta_4}\\
&\qquad\times\Big[\sigma'^{(\ell)}_{j_1;\delta_1}\sigma'^{(\ell)}_{j_2;\delta_2}\widehat{\mathrm{dd}_{\mathrm{II}}H}^{(\ell)}_{j_1j_2j_3j_4;\delta_1\delta_2\delta_3\delta_4}+\sigma''^{(\ell)}_{j_1;\delta_1}\sigma''^{(\ell)}_{j_2;\delta_2}\widehat{H}^{(\ell)}_{j_1j_2;\delta_1\delta_2}\widehat{H}^{(\ell)}_{j_1j_3;\delta_1\delta_3}\widehat{H}^{(\ell)}_{j_2j_4;\delta_2\delta_4}\\
&\qquad\quad+\sigma'^{(\ell)}_{j_1;\delta_1}\sigma''^{(\ell)}_{j_2;\delta_2}\widehat{H}^{(\ell)}_{j_2j_4;\delta_2\delta_4}\widehat{\mathrm{d}H}^{(\ell)}_{j_1j_2j_3;\delta_1\delta_2\delta_3}+\sigma'^{(\ell)}_{j_2;\delta_2}\sigma''^{(\ell)}_{j_1;\delta_1}\widehat{H}^{(\ell)}_{j_1j_3;\delta_1\delta_3}\widehat{\mathrm{d}H}^{(\ell)}_{j_2j_1j_4;\delta_2\delta_1\delta_4}\Big],
\end{aligned}
$$

$\widehat{\mathrm{dd}_{\mathrm{I}}H}$ Recursion

The mean of the first ddNTK decomposes as

$$
\begin{aligned}
&\mathbb{E}\left[\widehat{\mathrm{dd}_{\mathrm{I}}H}^{(\ell)}_{i_0i_1i_2i_3;\delta_0\delta_1\delta_2\delta_3}\right] \qquad (\infty.171)\\
=&\frac{1}{n_{\ell-1}}\left[\delta_{i_0i_1}\delta_{i_2i_3}R^{(\ell)}_{\delta_0\delta_1\delta_2\delta_3}+\delta_{i_0i_2}\delta_{i_3i_1}R^{(\ell)}_{\delta_0\delta_2\delta_3\delta_1}+\delta_{i_0i_3}\delta_{i_1i_2}R^{(\ell)}_{\delta_0\delta_3\delta_1\delta_2}\right].
\end{aligned}
$$

The tensor $R^{(\ell)}$ satisfies the following layer-to-layer recursion:

$$
\begin{aligned}
&R^{(\ell+1)}_{\delta_0\delta_1\delta_2\delta_3} \qquad (\infty.172)\\
=&\lambda_W^{(\ell+1)} C_W^{(\ell+1)} \left\langle \sigma''_{\delta_0}\sigma_{\delta_1}\sigma'_{\delta_2}\sigma'_{\delta_3}\right\rangle_{G^{(\ell)}} H^{(\ell)}_{\delta_0\delta_2} H^{(\ell)}_{\delta_0\delta_3}\\
&+\left(\frac{n_\ell}{n_{\ell-1}}\right)\left(C_W^{(\ell+1)}\left\langle \sigma'_{\delta_2}\sigma'_{\delta_3}\right\rangle_{G^{(\ell)}}\right)\left(\lambda_W^{(\ell+1)}\left\langle \sigma''_{\delta_0}\sigma_{\delta_1}\right\rangle_{G^{(\ell)}}\right) B^{(\ell)}_{\delta_0\delta_0\delta_2\delta_3}\\
&+\left(\frac{n_\ell}{n_{\ell-1}}\right)\left(C_W^{(\ell+1)}\left\langle \sigma'_{\delta_2}\sigma'_{\delta_3}\right\rangle_{G^{(\ell)}}\right)\left[\lambda_W^{(\ell+1)}\left(\left\langle \sigma''_{\delta_0}\sigma_{\delta_1}\right\rangle_{G^{(\ell)}} P^{(\ell)}_{\delta_0\delta_2\delta_3\delta_0}+\left\langle \sigma'_{\delta_0}\sigma'_{\delta_1}\right\rangle_{G^{(\ell)}} P^{(\ell)}_{\delta_0\delta_2\delta_3\delta_1}\right)\right]\\
&+\left(C_W^{(\ell+1)}\right)^2\left\langle \sigma'''_{\delta_0}\sigma'_{\delta_1}\sigma'_{\delta_2}\sigma'_{\delta_3}\right\rangle_{G^{(\ell)}} H^{(\ell)}_{\delta_0\delta_1} H^{(\ell)}_{\delta_0\delta_2} H^{(\ell)}_{\delta_0\delta_3}\\
&+\left(\frac{n_\ell}{n_{\ell-1}}\right)\left(C_W^{(\ell+1)}\left\langle \sigma'_{\delta_2}\sigma'_{\delta_3}\right\rangle_{G^{(\ell)}}\right)\left(C_W^{(\ell+1)}\left\langle \sigma'''_{\delta_0}\sigma'_{\delta_1}\right\rangle_{G^{(\ell)}}\right) B^{(\ell)}_{\delta_0\delta_0\delta_2\delta_3} H^{(\ell)}_{\delta_0\delta_1}\\
&+\left(\frac{n_\ell}{n_{\ell-1}}\right)\left(C_W^{(\ell+1)}\left\langle \sigma'_{\delta_2}\sigma'_{\delta_3}\right\rangle_{G^{(\ell)}}\right)\left[C_W^{(\ell+1)}\left(\left\langle \sigma'''_{\delta_0}\sigma'_{\delta_1}\right\rangle_{G^{(\ell)}} P^{(\ell)}_{\delta_0\delta_2\delta_3\delta_0}+\left\langle \sigma''_{\delta_0}\sigma''_{\delta_1}\right\rangle_{G^{(\ell)}} P^{(\ell)}_{\delta_0\delta_2\delta_3\delta_1}\right)\right] H^{(\ell)}_{\delta_0\delta_1}\\
&+\left(\frac{n_\ell}{n_{\ell-1}}\right)\left(C_W^{(\ell+1)}\left\langle \sigma'_{\delta_2}\sigma'_{\delta_3}\right\rangle_{G^{(\ell)}}\right)\left(C_W^{(\ell+1)}\left\langle \sigma'_{\delta_0}\sigma'_{\delta_1}\right\rangle_{G^{(\ell)}}\right) R^{(\ell)}_{\delta_0\delta_1\delta_2\delta_3}+O\left(\frac{1}{n}\right).
\end{aligned}
$$

$\widehat{\mathbf{dd_{II}}H}$ Recursions

The mean of the second ddNTK decomposes as

$$
\begin{aligned}
&\mathbb{E}\left[\widehat{\mathrm{dd}_{\mathrm{II}}H}^{(\ell)}_{i_1i_2i_3i_4;\delta_1\delta_2\delta_3\delta_4}\right] \qquad (\infty.173)\\
=&\frac{1}{n_{\ell-1}}\left[\delta_{i_1i_2}\delta_{i_3i_4}S^{(\ell)}_{\delta_1\delta_2\delta_3\delta_4}+\delta_{i_1i_3}\delta_{i_4i_2}T^{(\ell)}_{\delta_1\delta_3\delta_4\delta_2}+\delta_{i_1i_4}\delta_{i_2i_3}U^{(\ell)}_{\delta_1\delta_4\delta_2\delta_3}\right].
\end{aligned}
$$

The tensor $S^{(\ell)}$ satisfies the following layer-to-layer recursion:

$$
\begin{aligned}
&S^{(\ell+1)}_{\delta_1\delta_2\delta_3\delta_4} \qquad (\infty.174)\\
=&C_W^{(\ell+1)}\lambda_W^{(\ell+1)}\left\langle \sigma'_{\delta_1}\sigma'_{\delta_2}\sigma'_{\delta_3}\sigma'_{\delta_4}\right\rangle_{G^{(\ell)}} H^{(\ell)}_{\delta_1\delta_3} H^{(\ell)}_{\delta_2\delta_4}\\
&+\left(\frac{n_\ell}{n_{\ell-1}}\right)\left(C_W^{(\ell+1)}\left\langle \sigma'_{\delta_3}\sigma'_{\delta_4}\right\rangle_{G^{(\ell)}}\right)\left[\lambda_W^{(\ell+1)}\left\langle \sigma'_{\delta_1}\sigma'_{\delta_2}\right\rangle_{G^{(\ell)}}+C_W^{(\ell+1)}H^{(\ell)}_{\delta_1\delta_2}\left\langle \sigma''_{\delta_1}\sigma''_{\delta_2}\right\rangle_{G^{(\ell)}}\right] B^{(\ell)}_{\delta_1\delta_2\delta_3\delta_4}\\
&+\left(C_W^{(\ell+1)}\right)^2\left\langle \sigma''_{\delta_1}\sigma''_{\delta_2}\sigma'_{\delta_3}\sigma'_{\delta_4}\right\rangle_{G^{(\ell)}} H^{(\ell)}_{\delta_1\delta_2} H^{(\ell)}_{\delta_1\delta_3} H^{(\ell)}_{\delta_2\delta_4}\\
&+\left(\frac{n_\ell}{n_{\ell-1}}\right)\left(C_W^{(\ell+1)}\left\langle \sigma'_{\delta_1}\sigma'_{\delta_2}\right\rangle_{G^{(\ell)}}\right)\left(C_W^{(\ell+1)}\left\langle \sigma'_{\delta_3}\sigma'_{\delta_4}\right\rangle_{G^{(\ell)}}\right) S^{(\ell)}_{\delta_1\delta_2\delta_3\delta_4}+O\left(\frac{1}{n}\right).
\end{aligned}
$$

The tensor $T^{(\ell)}$ satisfies the following layer-to-layer recursion:

$$
\begin{aligned}
&T^{(\ell+1)}_{\delta_1\delta_3\delta_4\delta_2} \qquad (\infty.175)\\
&= \left(\lambda_W^{(\ell+1)}\right)^2 \left\langle \sigma'_{\delta_1}\sigma'_{\delta_2}\sigma_{\delta_3}\sigma_{\delta_4}\right\rangle_{G^{(\ell)}} H^{(\ell)}_{\delta_1\delta_2}\\
&\quad + \left(\frac{n_\ell}{n_{\ell-1}}\right)\left(\lambda_W^{(\ell+1)}\right)^2 \sum_{\delta_5,\ldots,\delta_8\in\mathcal{D}} \left\langle z_{\delta_5}\sigma'_{\delta_1}\sigma_{\delta_3}\right\rangle_{G^{(\ell)}} \left\langle z_{\delta_6}\sigma'_{\delta_2}\sigma_{\delta_4}\right\rangle_{G^{(\ell)}} G_{(\ell)}^{\delta_5\delta_7} G_{(\ell)}^{\delta_6\delta_8} F^{(\ell)}_{\delta_7\delta_1\delta_8\delta_2}\\
&\quad + C_W^{(\ell+1)}\lambda_W^{(\ell+1)} \left\langle \sigma'_{\delta_1}\sigma''_{\delta_2}\sigma_{\delta_3}\sigma'_{\delta_4}\right\rangle_{G^{(\ell)}} H^{(\ell)}_{\delta_2\delta_1} H^{(\ell)}_{\delta_2\delta_4}\\
&\quad + \left(\frac{n_\ell}{n_{\ell-1}}\right) C_W^{(\ell+1)}\lambda_W^{(\ell+1)} H^{(\ell)}_{\delta_2\delta_4} \sum_{\delta_5,\ldots,\delta_8\in\mathcal{D}} \left\langle z_{\delta_5}\sigma'_{\delta_1}\sigma_{\delta_3}\right\rangle_{G^{(\ell)}} \left\langle z_{\delta_6}\sigma''_{\delta_2}\sigma'_{\delta_4}\right\rangle_{G^{(\ell)}} G_{(\ell)}^{\delta_5\delta_7} G_{(\ell)}^{\delta_6\delta_8} F^{(\ell)}_{\delta_7\delta_1\delta_8\delta_2}\\
&\quad + \left(\frac{n_\ell}{n_{\ell-1}}\right) C_W^{(\ell+1)}\lambda_W^{(\ell+1)} \left\langle \sigma'_{\delta_2}\sigma'_{\delta_4}\right\rangle_{G^{(\ell)}} \left(\left\langle \sigma''_{\delta_1}\sigma_{\delta_3}\right\rangle_{G^{(\ell)}} Q^{(\ell)}_{\delta_2\delta_4\delta_1\delta_1} + \left\langle \sigma'_{\delta_1}\sigma'_{\delta_3}\right\rangle_{G^{(\ell)}} Q^{(\ell)}_{\delta_2\delta_4\delta_1\delta_3}\right)\\
&\quad + C_W^{(\ell+1)}\lambda_W^{(\ell+1)} \left\langle \sigma'_{\delta_2}\sigma''_{\delta_1}\sigma_{\delta_4}\sigma'_{\delta_3}\right\rangle_{G^{(\ell)}} H^{(\ell)}_{\delta_1\delta_2} H^{(\ell)}_{\delta_1\delta_3}\\
&\quad + \left(\frac{n_\ell}{n_{\ell-1}}\right) C_W^{(\ell+1)}\lambda_W^{(\ell+1)} H^{(\ell)}_{\delta_1\delta_3} \sum_{\delta_5,\ldots,\delta_8\in\mathcal{D}} \left\langle z_{\delta_5}\sigma'_{\delta_2}\sigma_{\delta_4}\right\rangle_{G^{(\ell)}} \left\langle z_{\delta_6}\sigma''_{\delta_1}\sigma'_{\delta_3}\right\rangle_{G^{(\ell)}} G_{(\ell)}^{\delta_5\delta_7} G_{(\ell)}^{\delta_6\delta_8} F^{(\ell)}_{\delta_7\delta_2\delta_8\delta_1}\\
&\quad + \left(\frac{n_\ell}{n_{\ell-1}}\right) C_W^{(\ell+1)}\lambda_W^{(\ell+1)} \left\langle \sigma'_{\delta_1}\sigma'_{\delta_3}\right\rangle_{G^{(\ell)}} \left(\left\langle \sigma''_{\delta_2}\sigma_{\delta_4}\right\rangle_{G^{(\ell)}} Q^{(\ell)}_{\delta_1\delta_3\delta_2\delta_2} + \left\langle \sigma'_{\delta_2}\sigma'_{\delta_4}\right\rangle_{G^{(\ell)}} Q^{(\ell)}_{\delta_1\delta_3\delta_2\delta_4}\right)\\
&\quad + \left(\frac{n_\ell}{n_{\ell-1}}\right)\left(C_W^{(\ell+1)}\right)^2 \left\langle \sigma'_{\delta_1}\sigma'_{\delta_3}\right\rangle_{G^{(\ell)}} \left\langle \sigma'_{\delta_2}\sigma'_{\delta_4}\right\rangle_{G^{(\ell)}} T^{(\ell)}_{\delta_1\delta_3\delta_4\delta_2}\\
&\quad + \left(C_W^{(\ell+1)}\right)^2 \left\langle \sigma''_{\delta_1}\sigma''_{\delta_2}\sigma'_{\delta_3}\sigma'_{\delta_4}\right\rangle_{G^{(\ell)}} H^{(\ell)}_{\delta_1\delta_2} H^{(\ell)}_{\delta_1\delta_3} H^{(\ell)}_{\delta_2\delta_4}\\
&\quad + \left(\frac{n_\ell}{n_{\ell-1}}\right)\left(C_W^{(\ell+1)}\right)^2 H^{(\ell)}_{\delta_1\delta_3} H^{(\ell)}_{\delta_2\delta_4} \sum_{\delta_5,\ldots,\delta_8\in\mathcal{D}} \left\langle z_{\delta_5}\sigma''_{\delta_1}\sigma'_{\delta_3}\right\rangle_{G^{(\ell)}} \left\langle z_{\delta_6}\sigma''_{\delta_2}\sigma'_{\delta_4}\right\rangle_{G^{(\ell)}} G_{(\ell)}^{\delta_5\delta_7} G_{(\ell)}^{\delta_6\delta_8} F^{(\ell)}_{\delta_7\delta_1\delta_8\delta_2}\\
&\quad + \left(\frac{n_\ell}{n_{\ell-1}}\right)\left(C_W^{(\ell+1)}\right)^2 H^{(\ell)}_{\delta_2\delta_4} \left\langle \sigma'_{\delta_1}\sigma'_{\delta_3}\right\rangle_{G^{(\ell)}} \left(\left\langle \sigma'''_{\delta_2}\sigma'_{\delta_4}\right\rangle_{G^{(\ell)}} Q^{(\ell)}_{\delta_1\delta_3\delta_2\delta_2} + \left\langle \sigma''_{\delta_2}\sigma''_{\delta_4}\right\rangle_{G^{(\ell)}} Q^{(\ell)}_{\delta_1\delta_3\delta_2\delta_4}\right)\\
&\quad + \left(\frac{n_\ell}{n_{\ell-1}}\right)\left(C_W^{(\ell+1)}\right)^2 H^{(\ell)}_{\delta_1\delta_3} \left\langle \sigma'_{\delta_2}\sigma'_{\delta_4}\right\rangle_{G^{(\ell)}} \left(\left\langle \sigma'''_{\delta_1}\sigma'_{\delta_3}\right\rangle_{G^{(\ell)}} Q^{(\ell)}_{\delta_2\delta_4\delta_1\delta_1} + \left\langle \sigma''_{\delta_1}\sigma''_{\delta_3}\right\rangle_{G^{(\ell)}} Q^{(\ell)}_{\delta_2\delta_4\delta_1\delta_3}\right)\\
&\quad + O\left(\frac{1}{n}\right).
\end{aligned}
$$

The tensor $U^{(\ell)}$ satisfies the following layer-to-layer recursion:

$$
\begin{aligned}
U^{(\ell+1)}_{\delta_1\delta_4\delta_2\delta_3} &= \left(C_W^{(\ell+1)}\right)^2 \left\langle \sigma''_{\delta_1}\sigma''_{\delta_2}\sigma'_{\delta_3}\sigma'_{\delta_4}\right\rangle_{G^{(\ell)}} H^{(\ell)}_{\delta_1\delta_2} H^{(\ell)}_{\delta_1\delta_3} H^{(\ell)}_{\delta_2\delta_4} \qquad (\infty.176)\\
&\quad + \left(\frac{n_\ell}{n_{\ell-1}}\right)\left(C_W^{(\ell+1)}\left\langle \sigma'_{\delta_1}\sigma'_{\delta_4}\right\rangle_{G^{(\ell)}}\right)\left(C_W^{(\ell+1)}\left\langle \sigma'_{\delta_2}\sigma'_{\delta_3}\right\rangle_{G^{(\ell)}}\right) U^{(\ell)}_{\delta_1\delta_4\delta_2\delta_3} + O\left(\frac{1}{n}\right).
\end{aligned}
$$

Epilogue ε

Model Complexity from the Macroscopic Perspective

> *According to the hype of 1987, neural networks were meant to be intelligent models that discovered features and patterns in data. Gaussian processes in contrast are simply smoothing devices. How can Gaussian processes possibly replace neural networks? Were neural networks over-hyped, or have we underestimated the power of smoothing methods?*
>
> David MacKay [69].

Throughout this book, we've focused on deep-learning models that are very wide but also deep. Our reason for this focus is that such large neural networks with many many model parameters work extremely well in practice and thus form the foundation of the modern approach to artificial intelligence.

The success of these **overparameterized** models with far more parameters than training data has led many to simply conjecture that "more is better" when it comes to deep learning. In a more refined sense, there's mounting empirical evidence that a **scaling hypothesis** can accurately capture the behavior of deep neural networks, and its associated *scaling laws* overall point towards the optimality of the overparameterized regime.[1] The simplicity of these empirical laws recalls an earlier period in statistical physics, when a similar scaling hypothesis was conjectured to govern the behavior of certain complicated systems in statistical mechanics.[2]

[1] See e.g. [70] for an empirical study of scaling laws in deep learning language models based on the transformer architecture. Empirically, it's observed that overparameterization is good; the optimal growth of the number of training samples $N_{\mathcal{A}}$ scales sublinearly with the growth in parameters P, though importantly they should still scale together with a power law: $N_{\mathcal{A}} \propto P^{\alpha}$ for $0 < \alpha < 1$.

[2] In physics, such scaling laws are an example of the phenomenon of *universality*, the fact that when a system has many elementary components, it can often be described by a very simple *effective theory* that's independent of the microscopic details of the underlying system [47]. The framework of *renormalization group* then offers an explanation for how this universality arises by characterizing the flow from the microscopic to the macroscopic [36, 37]. This perhaps suggests that the analogous notion of

However, the practical success of overparameterized models in deep learning appears to be in tension with orthodox machine learning and classic statistical theory. Heuristically, the *Occam's razor* principle of sparsity posits that we should favor the simplest hypothesis that explains our observations: in the context of machine learning, this is usually interpreted to mean that we should prefer models with fewer parameters when comparing models performing the same tasks. More quantitatively, we expect that models with fewer parameters will have smaller *generalization errors*, $\mathcal{E} \equiv \mathcal{L}_{\mathcal{B}} - \mathcal{L}_{\mathcal{A}}$, and will be less prone to overfit their training set $\mathcal{A}$. Vice versa, we should naively expect that overparameterized models *will* overfit their training data and generalize poorly. Thus, this orthodoxy is in direct conflict with the empirical success of overparameterized neural networks and is a big theoretical puzzle in understanding modern deep learning.[3]

In this brief epilogue, we're going to offer a resolution of this puzzle. The crux of the matter hinges on the notion of **model complexity**. On the one hand, our orthodox discussion of generalization above took a **microscopic perspective** – focusing on how a network works in terms of its explicit low-level components – and wanted to identify model complexity with model parameters. On the other hand, in this book we integrated out the model parameters and developed a **macroscopic perspective** – providing an effective theory description of the predictions of realistic fully-trained networks – for which this notion of model complexity is completely reversed.

Indeed, we motivated our effective theory approach in §0 on the basis that we would be able to find simplicity in the limit of a large number of model parameters, and from §1 to §∞ we've now seen how this hypothesis has been borne out again and again for realistic large-but-finite-width networks. Having finished all the technical calculations (outside of the appendices), we can see now that it's the depth-to-width aspect ratio,

$$r \equiv L/n\,, \tag{ε.1}$$

that controls the model complexity of overparameterized neural networks. To understand why, recall that this ratio emerged from our calculations as the expansion parameter or *cutoff* of our effective theory and determined how we could *truncate* the series expansion of the fully-trained distribution while still approximating the true behavior of networks with minimal error. This means that it's the number of data-dependent couplings of the truncated nearly-Gaussian distribution – and *not* the number of model parameters – that ultimately define the model complexity in deep learning. From this macroscopic perspective, there's absolutely no conflict between the sparse intuition of Occam's razor in theory and the simplicity of the scaling hypothesis in practice.

To see how this works in even greater detail, let us recall the three main problems we discussed at the beginning of the book in §0.2, and then review how the principle of sparsity enabled us to solve them.[4] Taylor expanding the trained network output around

representation group flow (cf. §4.6) may be able to explain the neural scaling laws of [70].

[3] See, e.g., the extensive discussion in [71] on the difficulty of trying to understand why large neural networks generalize so well according to traditional measures of model complexity.

[4] In this epilogue, we'll drop layer indices on our variables to ease the notation, as everything is evaluated at the output layer; we'll also sometimes drop the neural indices when they are unimportant.

the network's initialization (0.5),

$$z(x_\delta;\theta^\star) = z(x_\delta;\theta) + \sum_{\mu=1}^{P} (\theta^\star - \theta)_\mu \frac{dz_\delta}{d\theta_\mu} + \frac{1}{2}\sum_{\mu,\nu=1}^{P} (\theta^\star - \theta)_\mu (\theta^\star - \theta)_\nu \frac{d^2 z_\delta}{d\theta_\mu d\theta_\nu} + \ldots , \tag{ε.2}$$

we illustrated **Problem 1**, (0.6), that we might have to compute an infinite number of terms,

$$z\,, \quad \frac{dz}{d\theta}\,, \quad \frac{d^2 z}{d\theta^2}\,, \quad \frac{d^3 z}{d\theta^3}\,, \quad \frac{d^4 z}{d\theta^4}\,, \quad \ldots\,, \tag{ε.3}$$

Problem 2, (0.7), that we have to determine the map from the initialization distribution over the model parameters to the induced initialization distribution over the network output and its derivatives,

$$p(\theta) \to p\left(z, \frac{dz}{d\theta}, \frac{d^2 z}{d\theta^2}, \ldots\right) , \tag{ε.4}$$

and **Problem 3**,(0.8), that we have to solve the training dynamics, which can depend on *everything*,

$$\theta^\star \equiv [\theta^\star]\left(\theta,\, z,\, \frac{dz}{d\theta},\, \frac{d^2 z}{d\theta^2},\, \ldots;\, \text{learning algorithm};\, \text{training data}\right) . \tag{ε.5}$$

Let us now give our detailed solutions to this problem set – expanding on the schematic explanations that we gave in §0.2 – and then carefully examine them through the lens of *model complexity*. We'll begin first with the simple infinite-width limit and then discuss the nearly-simple-but-realistic $1/n$ truncation.

Sparsity at Infinite Width

In the infinite-width limit, we can now understand our solutions as follows:

- Addressing **Problem 1**, (0.11), all the higher derivative terms vanish, and we only need to keep track of two statistical variables:

$$z\,, \quad \frac{dz}{d\theta} \quad \Longrightarrow \quad z_\delta\,, \quad \widehat{H}_{\delta_1\delta_2}\,. \tag{ε.6}$$

  Note that this first derivative gives the *random features* of the linear model description, cf. (10.138), and the kernel associated with these features is just the NTK.

- Addressing **Problem 2**, we found that the network output and its first derivative are statistically independent, each governed by the simple distribution (0.12):

$$\lim_{n\to\infty} p\left(z, \frac{dz}{d\theta}, \frac{d^2 z}{d\theta^2}, \ldots\right) = p(z_\delta)\,\delta\left(\sum_{\mu,\nu} \lambda_{\mu\nu} \frac{dz_{\delta_1}}{d\theta_\mu}\frac{dz_{\delta_2}}{d\theta_\nu} - \Theta_{\delta_1\delta_2}\right) , \tag{ε.7}$$

where on the right-hand side, $p(z_\delta)$ is a zero-mean Gaussian distribution with its variance given by the kernel $K_{\delta_1\delta_2}$ (4.106), and the second factor is a Dirac delta function distribution that fixes the contraction of first derivatives that make up the NTK $\widehat{H}_{\delta_1\delta_2}$ to be deterministically given by the frozen NTK $\Theta_{\delta_1\delta_2}$ (9.4).

- Addressing **Problem 3**, we obtained a solution for the trained model parameters $\theta^\star$ in a *closed form* (0.13):

$$\lim_{n\to\infty}\theta^\star_\mu = \theta_\mu(t=0) - \sum_{\nu,\tilde{\alpha}_1,\tilde{\alpha}_2,i}\lambda_{\mu\nu}\frac{dz_{i;\tilde{\alpha}_1}}{d\theta_\nu}\widetilde{\Theta}^{\tilde{\alpha}_1\tilde{\alpha}_2}\left(z_{i;\tilde{\alpha}_2}-y_{i;\tilde{\alpha}_2}\right) , \tag{ε.8}$$

  with the associated fully-trained network outputs $z_\delta(T) = z(x_\delta;\theta^\star)$ given in (10.31). We further showed in §10.2.2 that this fully-trained solution is independent of the algorithm used to train the network.

Combining these insights, we found that the fully-trained distribution,

$$\lim_{n\to\infty} p\Big(z(T)\Big) \equiv p\Big(z(T)\Big|y_{\tilde{\alpha}}, K_{\delta_1\delta_2}, \Theta_{\delta_1\delta_2}\Big) , \tag{ε.9}$$

is a *Gaussian distribution*; the reason for writing it as a conditional distribution in this way is that the mean, (10.40), is only a function of vector of training set labels, $y_{\tilde{\alpha}}$, and the frozen NTK matrix, $\Theta^{(L)}_{\delta_1\delta_2}$, while the variance, (10.41), is only a function of the kernel matrix, $K^{(L)}_{\delta_1\delta_2}$, and the frozen NTK matrix, $\Theta^{(L)}_{\delta_1\delta_2}$. In other words, the distribution of predictions on a test sample, $x_{\dot{\beta}}$, will depend on the test sample together with all of the training data, $\mathcal{D} = \{\dot{\beta}\}\cup\mathcal{A}$, and the shape of that data dependence is governed by the specific functional forms of the data-dependent couplings, i.e. the output-layer kernel $K(x_{\delta_1}, x_{\delta_2})$ and output-layer frozen NTK $\Theta(x_{\delta_1}, x_{\delta_2})$. Thus, the infinite-width solution (ε.9) allows for a very sparse description, depending only on a few objects in a simple way.

Near-Sparsity at Finite Width

Similarly, at large-but-finite width, we can now understand our solutions as follows:

- Addressing **Problem 1**, (0.16), all derivatives $d^k f/d\theta^k$ for $k \geq 4$ are $O(1/n^2)$, and so we only need to keep track of the statistical variables up to the third derivative:

$$z\,,\quad \frac{dz}{d\theta}\,,\quad \frac{d^2z}{d\theta^2}\,,\quad \frac{d^3z}{d\theta^3} \quad\Longrightarrow\quad z_\delta\,,\widehat{H}_{\delta_1\delta_2}\,,\widehat{\mathrm{d}H}_{\delta_0\delta_1\delta_2}\,,\widehat{\mathrm{dd_I}H}_{\delta_0\delta_1\delta_2\delta_3}\,,\widehat{\mathrm{dd_{II}}H}_{\delta_1\delta_2\delta_3\delta_4}\,. \tag{ε.10}$$

  Here, we note that the NTK, dNTK, and ddNTKs capture all the terms up to the third derivative in our Taylor expansion (ε.2) for a gradient-based learning update, cf. (∞.4), (∞.6), and (∞.9).

- Addressing **Problem 2**, (0.17), we evaluated the distribution of all these statistical variables, and found that its joint distribution is nearly-Gaussian:

$$p\left(z, \frac{dz}{d\theta}, \frac{d^2z}{d\theta^2}, \ldots\right) = p\Big(z,\, \widehat{H},\, \widehat{\mathrm{d}H},\, \widehat{\mathrm{dd_I}H},\, \widehat{\mathrm{dd_{II}}H}\Big) + O\left(\frac{1}{n^2}\right) . \tag{ε.11}$$

- Addressing **Problem 3**, (0.18), we were able to use perturbation theory to solve the nonlinear training dynamics and evaluate the predictions of fully-trained networks at finite width:

$$z(x_\delta;\theta^\star) = [z(x_\delta;\theta^\star)]\left(z,\, \widehat{H},\, \widehat{\mathrm{d}H},\, \widehat{\mathrm{dd}_{\mathrm{I}}H},\, \widehat{\mathrm{dd}_{\mathrm{II}}H};\, \text{algorithm projectors}\right) . \qquad (\varepsilon.12)$$

  Here, the details of the *algorithm dependence* of the prediction is manifest, captured entirely by a handful of algorithm projectors, cf. (∞.142)–(∞.147).

Combining these insights, we found that the fully-trained distribution,

$$p\Big(z(T)\Big) \equiv p\Big(z(T)\Big|y, G, H, V, A, B, D, F, P, Q, R, S, T, U\Big) + O\left(\frac{1}{n^2}\right) , \qquad (\varepsilon.13)$$

is a *nearly-Gaussian distribution*; its statistics are entirely described by the conditional variables listed here, though we've suppressed the sample indices in order to fit them all on one line.[5] In addition to the metric G and the NTK mean H, in this conditioning we're accounting for the finite-width data-dependent couplings arising from our decompositions of the four-point connected correlator of preactivations $\mathbb{E}\left[zzzz\right]_{\text{connected}}$, (4.77), the NTK-preactivation cross correlator $\mathbb{E}\left[\widehat{\Delta H}zz\right]$, (8.67), the NTK variance $\mathbb{E}\left[\widehat{\Delta H}^2\right]$, (8.82), the dNTK-preactivation cross correlator $\mathbb{E}\left[\widehat{\mathrm{d}H}z\right]$, (11.45), and the means of the ddNTKs $\mathbb{E}\left[\widehat{\mathrm{dd}_{\mathrm{I}}H}\right]$ and $\mathbb{E}\left[\widehat{\mathrm{dd}_{\mathrm{II}}H}\right]$, ($\infty$.11) and ($\infty$.12). Importantly, all of these finite-width tensors at $O(1/n)$ are functions of exactly four input samples each, e.g. for the four-point vertex we have $V_{(\delta_1\delta_2)(\delta_3\delta_4)} \equiv V(x_{\delta_1}, x_{\delta_2}, x_{\delta_3}, x_{\delta_4})$, and the specific functional forms of these data-dependent couplings determine the overall data dependence of the distribution. Thus, though slightly more complicated than the infinite-width description (ε.9), the solution truncated to $O(1/n)$, (ε.13), is a nearly-sparse description, depending only on two-hands-full of objects in a nearly-simple way.

Model Complexity of Fully-Trained Neural Networks

These effective theory results, (ε.9) and (ε.13), should make it clear that for overparameterized neural networks, it is no longer appropriate to identify the number of model parameters with the model complexity. Consider a fixed combined training and test dataset of size $N_{\mathcal{D}}$:

- For the Gaussian distribution, (ε.9), that describes an ensemble of fully-trained infinite-width networks, we only need

$$n_{\text{out}}N_{\mathcal{A}} + \left[\frac{N_{\mathcal{D}}(N_{\mathcal{D}}+1)}{2}\right] + \left[\frac{N_{\mathcal{D}}(N_{\mathcal{D}}+1)}{2}\right] = O\Big(N_{\mathcal{D}}^2\Big) \qquad (\varepsilon.14)$$

[5] Note that we've also suppressed the algorithm projectors in this conditioning, presuming that we are considering a fixed learning algorithm: once an algorithm is fixed, the projectors can only be fixed functions of any training set tensors that contain $O(1)$ terms – i.e. the metric submatrix and the NTK mean submatrix, both evaluated on the training set only, $\widetilde{G}_{\tilde{\alpha}_1\tilde{\alpha}_2}$ and $\widetilde{H}_{\tilde{\alpha}_1\tilde{\alpha}_2}$ – and of the global learning rate, η, see e.g. (∞.142)–(∞.147) for gradient descent. Thus, the algorithm dependence is taken care of entirely by the tensors we're conditioning on already.

numbers in order to completely specify the distribution, with each term corresponding to the numbers needed to enumerate $y_{i;\tilde{\alpha}}$, $K_{\delta_1\delta_2}$, and $\Theta_{\delta_1\delta_2}$, respectively.

- For the nearly-Gaussian distribution, (ε.13), that describes an ensemble of fully-trained finite-width networks with small-but-nonzero aspect ratios, $0 < r \ll 1$, we now need
$$O\Big(N_{\mathcal{D}}^4\Big) \tag{ε.15}$$
numbers in order to completely specify the distribution, with the counting dominated by the finite-width tensors, each of which having exactly four sample indices.

Thus, while each infinite-width network has an *infinite* number of microscopic model parameters, its macroscopic data-dependent couplings are only *quadratic* in samples. Meanwhile, our finite-width networks have less model parameters than our infinite-width network – i.e. finite $<$ infinite – but their macroscopic effective description is more complicated! What we have found here is the manifestation of the **microscopic-macroscopic duality**: under this duality, complexity in *parameter space* is transformed into simplicity in *sample space*, and density in *model parameters* is exchanged for sparsity in *data-dependent couplings*. In the overparameterized regime, this duality indicates that we really should identify the model complexity with the data-dependent couplings rather than the model parameters.

To further elaborate on this general point, we could imagine carrying out our finite-width $1/n$ expansion, (0.15), to higher orders as

$$p\Big(z(T)\Big) = p^{\{0\}}\Big(z(T)\Big) + \frac{p^{\{1\}}\Big(z(T)\Big)}{n} + \frac{p^{\{2\}}\Big(z(T)\Big)}{n^2} + O\Big(\frac{1}{n^3}\Big) \ . \tag{ε.16}$$

To begin, now the preactivation distribution at initialization,

$$p\Big(z\Big|\mathcal{D}\Big) = \frac{1}{Z}e^{-S(z)} + O\Big(\frac{1}{n^3}\Big) \ , \tag{ε.17}$$

will be effectively described in terms of a *sextic action*,

$$\begin{aligned} S(z) \equiv & \frac{1}{2}\sum_{i=1}^{n_L}\sum_{\delta_1,\delta_2\in\mathcal{D}} g^{\delta_1\delta_2} z_{i;\delta_1} z_{i;\delta_2} \\ & - \frac{1}{8}\sum_{i_1,i_2=1}^{n_L}\sum_{\delta_1,\ldots,\delta_4\in\mathcal{D}} v^{(\delta_1\delta_2)(\delta_3\delta_4)} z_{i_1;\delta_1} z_{i_1;\delta_2}\, z_{i_2;\delta_3} z_{i_2;\delta_4} \\ & + \frac{1}{24}\sum_{i_1,i_2,i_3=1}^{n_L}\sum_{\delta_1,\ldots,\delta_6\in\mathcal{D}} u^{(\delta_1\delta_2)(\delta_3\delta_4)(\delta_5\delta_6)} z_{i_1;\delta_1} z_{i_1;\delta_2}\, z_{i_2;\delta_3} z_{i_2;\delta_4}\, z_{i_3;\delta_5} z_{i_3;\delta_6} \ . \end{aligned} \tag{ε.18}$$

In this action, the *sextic coupling* scales as

$$u^{(\delta_1\delta_2)(\delta_3\delta_4)(\delta_5\delta_6)} = O\Big(\frac{1}{n^2}\Big) \ , \tag{ε.19}$$

and leads to a nontrivial connected six-point correlator characterized by a *six-point vertex*: $U_{(\delta_1\delta_2)(\delta_3\delta_4)(\delta_5\delta_6)}$.[6] At criticality, this connected correlator scales as

$$\frac{1}{n^2}U_{(\delta_1\delta_2)(\delta_3\delta_4)(\delta_5\delta_6)} \propto O\left(\frac{L^2}{n^2}\right) , \tag{ε.22}$$

consistent with the expectations of our effective theory cutoff. Thus, we expect that the refined description, (ε.16), is accurate to order L^2/n^2, but at the cost of a significant increase in the model complexity: the counting will be dominated by the $1/n^2$ finite-width tensors – each of which has six sample indices – and so we now require

$$O\Big(N_{\mathcal{D}}^6\Big) \tag{ε.23}$$

numbers in order to specify all the data-dependent couplings of the distribution. In general, to achieve an accuracy of order L^k/n^k, we expect that a macroscopic description

$$p\Big(z(T)\Big) = \sum_{m=0}^{k} \frac{p^{\{m\}}\Big(z(T)\Big)}{n^m} + O\left(\frac{L^{k+1}}{n^{k+1}}\right) , \tag{ε.24}$$

will have its model complexity dominated by data-dependent couplings with $2k$-sample indices, requiring

$$O\Big(N_{\mathcal{D}}^{2k}\Big) \tag{ε.25}$$

numbers. In this way, the **$1/n$ expansion** gives a sequence of effective theories with increasing accuracy at the cost of increasing complexity.[7]

[6] We computed this vertex in the second layer in footnote 7 of §4.2, and we've further taught you everything that you need to know in order to extend the computation of such higher-point correlators to deeper layers, and then analyze their scaling at criticality. As a checkpoint for your algebra, the single-input recursion for the six-point vertex specialized to the `ReLU` activation function is

$$U^{(\ell+1)} = \frac{1}{8}\left[C_W^{(\ell+1)}\right]^3 \left[U^{(\ell)} + 30V^{(\ell)}K^{(\ell)} + 44\left(K^{(\ell)}\right)^3\right] , \tag{ε.20}$$

which at criticality has a solution

$$\frac{U^{(\ell)}}{n^2\left(K^{(\ell)}\right)^3} = 75\frac{\ell^2}{n^2} + \ldots . \tag{ε.21}$$

[7] Note here that this counting is for the union of the training set and the test set, $\mathcal{D} = \mathcal{A} \cup \mathcal{B}$, rather than just for the training set $\mathcal{A}$; in other words, the stochastic predictions made on a test sample, $x_{\hat{\beta}}$, will necessarily depend on that test sample. In particular, every time you want to make a prediction on a new sample that you haven't predicted before, $N_{\mathcal{D}}$ increases in size, and so does your description of the joint distribution – (ε.9) and (ε.13) – over the entire dataset $\mathcal{D}$. Statistical models that have this property are sometimes called **non-parametric models**, since the "parameters" of the full distribution – i.e. the *data-dependent couplings* – depend on data points that you may not have even seen yet.

From the macroscopic perspective, the description of any *single* prediction scales with the size of the training set as $O\big((N_{\mathcal{A}}+1)^p\big) \approx O(N_{\mathcal{A}}^p)$; in principle you can just plug $x_{\hat{\beta}}$ into a prediction formula such as (∞.154). From the microscopic perspective, if we train a model and find a solution $\theta^\star \equiv \theta^\star(\mathcal{A})$ for the model parameters given a training set $\mathcal{A}$, then in practice we can then forget about that training

Importantly, for any particular network architecture that we want to describe, as the depth-to-width ratio $r \equiv L/n$ increases, we'll in principle need to include more and more of these higher-order terms, making our macroscopic effective theory more and more complex:

- In the strict limit $r \to 0$, the interactions between neurons turn off, and the sparse $O(N_{\mathcal{D}}^2)$ *Gaussian* description of the infinite-width limit, (ε.9), will be accurate. Such networks are not really deep, as $L/n = 0$, and they do not learn representations (§10.4.3).

- In the regime $0 < r \ll 1$, there are small nontrivial interactions between neurons, and the nearly-sparse $O(N_{\mathcal{D}}^4)$ *nearly-Gaussian* description of the finite-width effective theory truncated at order $1/n$, (ε.13), will be accurate. Such networks are wide while at the same time having nontrivial depth, $L/n \neq 0$, and they do learn representations (§11.4.3).

- For larger r, the neurons are strongly-coupled, and a more generic $O(N_{\mathcal{D}}^{2k})$ *non-Gaussian* description, (ε.24), would in principle be necessary. However, in this case the the macroscopic perspective leads to an *ineffective* description that is not tractable, and relatedly, we do not expect such networks to be practically useful for machine learning tasks (§3.4).[8]

set and simply make predictions as $z(x_{\beta}; \theta^{\star})$ – paying only the computation complexity cost of using the model to make a prediction. Thus, in deep learning we can think of this non-parametric growth of macroscopic model complexity with the size of the test set as similar in nature to the microscopic $O(P)$ complexity of the forward pass needed to make a new prediction.

Potentially it may have been useful to tell you – at the very least in order to understand this epilogue's epigraph – that a non-parametric model based on a Gaussian distribution is called a **Gaussian process**, and accordingly people sometimes say that neural networks in the infinite-width limit are Gaussian processes. Similarly, if you wanted to talk about the distribution over finite-width networks in the context of non-parametric statistics, you might call it a **nearly-Gaussian process**. These processes are distributions over functions $z(x)$, where e.g. z is the trained model output function. However, the reason why we haven't brought this up before is that we find this terminology unnecessary: for any fixed dataset $\mathcal{D}$, we don't have to worry about distributions over *functions* and instead can just think about a joint distribution over the finite *sets* of outputs evaluated on the dataset.

[8] In §3.4, we were able to access this regime in the special case of *deep linear networks*. There, we saw that higher-point connected correlators can grow uncontrollably even when the network is tuned to criticality, and there's no reason to expect that this would be any more favorable for nonlinear activation functions. Moreover, as we discussed in §5.4, the growth of these higher-order correlators in networks for all choices of activation functions will lead to large fluctuations from instantiation-to-instantiation, meaning that the ensemble description (ε.24) can no longer be trusted for any *particular* network. Altogether, this suggests that networks of an aspect ratio r that require large sample-space complexity, $O\big(N_{\mathcal{D}}^{2k}\big)$ with large k, will generically exhibit strongly-coupled chaos; we do not expect such networks to be effectively describable or practically trainable.

Even if we could find a workable network with such a large complexity, would we ever need it? As a *gedanken model*, let's consider an ineffective description with almost no truncation, say $k \sim N_{\mathcal{D}}$, which comes with an exponential number of data-dependent couplings, $O\big(N_{\mathcal{D}}^{N_{\mathcal{D}}}\big)$. Such an exponentially complex description would only really be appropriate when we have *unstructured data*: e.g. if we have a binary labeling $f(x) = \{0, 1\}$ for which each label is chosen *randomly*, then the number of possible functions for $N_{\mathcal{D}}$ uncorrelated data points is $2^{N_{\mathcal{D}}}$; each time we want to incorporate a new input-

In this way, our effective theory cutoff scale r governs model complexity of the statistics needed to faithfully describe the behavior of different neural networks. The simplicity of the *macroscopic perspective* only emerges for small values of the cutoff r.

With that in mind, the practical success of deep learning in the overparameterized regime and the empirical accuracy of a simple scaling hypothesis is really telling us that useful neural networks should be *sparse* – hence the preference for larger and larger models – but not too sparse – so that they are also *deep*. Thus, from the macroscopic perspective, a **nearly-sparse** model complexity is perhaps the most important inductive bias of deep learning.

♦♦♦♦

For an *information-theoretic* estimate of the depth-to-width ratio $r^\star$ for which the wide attraction of simplicity and the deep need of complexity are balanced to the end of *near-sparsity*, please feel free to flip the page and make your way through Appendix A.

output pair into our dataset, we'd have to *double* the complexity of our description. Such unstructured data is at odds with one of the main purposes of machine learning: recognizing patterns in the data – i.e. correlations – which allow for the learning of representations and the finding of sparse descriptions. In fact, as there's no efficient way to learn such unstructured datasets – see e.g. the *no-free-lunch theorem* of [72, 73] – in practice we cannot possibly require these ineffective descriptions for any realistic machine learning scenarios.

Appendix A

Information in Deep Learning

What can we demand from any physical theory? . . . nature [is] a difficult problem, but not a mystery for the human mind.

Ludwig Boltzmann [74]

In our *Initialization*, §0, we introduced our effective theory approach to understanding neural networks via the lens of theoretical physics. In particular, we discussed how thermodynamics was used to clarify the behavior of artificial machines such as the steam engine, and then we described how statistical mechanics was developed to explain how these macroscopic laws arise from the statistical behavior of many microscopic elements. With this perspective, we suggested that a similar framework might be applied to the difficult problem of deep learning theory, which we have now demystified from *Pretraining* all the way to the *End of Training*, §1 to §∞.

In this first appendix, we'll make the connection between deep learning and these fields of physics even more detailed. To do so, we'll reformulate a few of our previous results in terms of **information theory**. Initially formalized by Shannon as a way of quantitatively understanding digital communication, information theory was developed about half a century after statistical mechanics and is the statistical microscopic theory most fitting for the digital Information Age.

Although they a priori consider very different subject matter, both statistical mechanics and information theory share a joint language and ultimate focus on the same main fundamental concept: *entropy*. A particularly nice organization of entropies defines the *mutual information*, a positive quantity that can be used to characterize how much the measurement of one observable can inform us about another. This gives a nice way to quantify the overall statistical dependence due to the interactions of random variables. As such, the first section of this appendix (§A.1) will give a self-contained introduction to entropy and mutual information, making sure to point out the connections between the very physical and analog setting of statistical mechanics and the very abstract and digital world of information theory.

With these new tools, we will be able to further understand information in deep

learning. In particular, for infinite-width neural networks (§A.2) we'll find new perspectives on the non-interaction of neurons within a layer and on the necessity of criticality for preserving the information essential for distinguishing between input samples in deep networks. Then, at finite width (§A.3) we'll use a *variational principle* to demonstrate how the *principle of maximal entropy* for nearly-Gaussian distributions enables us to compute entropies and informations up to order $1/n^3$ while only needing to know the effective ℓ-th-layer preactivation distribution truncated at order $1/n$.

At order $1/n^2$, this will let us see a nonzero mutual information between groups of neurons in a layer that grows quadratically with depth, further quantifying the interaction of neurons at finite width and providing an information-theoretic interpretation and generalization of our Hebbian learning result from §6.4.1.

At order $1/n^3$, we will see how to pick a depth-to-width ratio, $r \equiv L/n$, that maximizes the mutual information as a functional of the activation function. This optimal aspect ratio, $r^\star$, arises from an unsupervised learning objective, and defines an activation-function-dependent scale that separates *effectively-deep* networks – that perform well – from *overly-deep* networks – that are no longer trainable.

Also at order $1/n^3$, a generalization of the mutual information for three groups of neurons will show that the information in a finite-width layer is stored redundantly between neurons. This analysis provides a new perspective on coarse-graining mechanism of *RG flow* by enabling us to understand how the information from inputs gets represented by, and shared among, the deeper-layer neurons.

Finally, note that while most of the computations presented here focus on the prior distribution of preactivations as a means of investigating the inductive bias of the network architecture and activation function, these information-theoretic tools naturally extend to the various joint distributions we've considered throughout the book as well as the Bayesian posterior distribution and the complicated distributions of fully-trained networks. This leaves many more things to be computed, and so we hope that this chapter provides a useful introduction to a new toolkit that can be used for furthering your understanding of deep learning. In our second and final appendix, §B, we'll give a *residual* example to demonstrate this in the setting of *residual networks*.

A.1 Entropy and Mutual Information

In this section we give a very brief overview of the concepts of entropy and mutual information that play essential roles both in *statistical mechanics* [75] and *information theory* [76, 77].

Let's start with a discrete random variable $x \in \mathcal{X}$ governed by the probability distribution $p(x)$. For this discussion, it is nice to think of x as a particular observable outcome, and $\mathcal{X}$ as the set of possible outcomes. The **entropy** of a probability distribution is given by

$$\mathcal{S}[p(x)] \equiv -\sum_{x\in\mathcal{X}} p(x) \log p(x) \,, \tag{A.1}$$

which is a *functional* of the distribution, taking a distribution as an argument and

outputting a number. Thus, we should think of the entropy as a property of an entire probability distribution.

To gain some intuition for why this quantity could be useful, let's consider its two extremes. First, when the distribution is perfectly *ordered* – that is, $p(x) = \delta_{xx'}$ such that $p(x') = 1$ with *absolute certainty* for a particular outcome, $x' \in \mathcal{X}$, and zero for all others – then the entropy is minimal, given by

$$\mathcal{S}[p(x)] = -\sum_{x\in\mathcal{X}} \delta_{xx'} \log(\delta_{xx'}) = 0\,, \tag{A.2}$$

since x' contributes $1\log(1) = 0$ and the other values of x contribute $0\log(0) = 0$. Second, when the distribution is completely *disordered* – that is, $p(x) = 1/|\mathcal{X}|$, such that the possible outcomes x are distributed uniformly and no outcome is more likely than any other – then entropy is maximal, given by

$$\mathcal{S}[p(x)] = -\sum_{x\in\mathcal{X}} \frac{1}{|\mathcal{X}|} \log\left(\frac{1}{|\mathcal{X}|}\right) = \log(|\mathcal{X}|)\,. \tag{A.3}$$

For this reason, in physics the entropy is often regarded as a measure of the **disorder** of a system characterized by a random variable x.[1] In this maximal case when each outcome is equally likely, the entropy can also be interpreted as a way of counting the number of states of a system – i.e. the number of distinct outcomes in $\mathcal{X}$ – since it's equal to the logarithm of such a count.[2]

[1] Since we're currently wearing our physicist hats, one remark is in (dis)order for the units of the entropy. As per our discussion of *dimensional analysis* in footnote 15 of §1.3, the logarithm of a probability has the same units as the *action*, and for the same reason must be dimensionless. Nevertheless, by changing the base of the logarithm, we can change the multiplicative constant in front of (A.1), and thus change the meaning of the entropy:

- With our physicist hats still on, we would use the natural logarithm with base e, which measures entropy in units with a very silly name called *nats*. (Some physicists also multiply the expression (A.1) by the Boltzmann constant k_{B}, as is natural in macroscopic applications of the entropy in *thermodynamics*, which gives the entropy the unit of *joules per kelvin*.)
- With our computer scientist hats on, which we put on in anticipation of the next paragraph in the main text, we would use the logarithm with base 2, which measures units in *binary digits* or the hopefully familiar **bits**. This is most natural in an information theory context, for which the entropy (A.1) is sometimes called the *Shannon entropy*. The reason bits are also used as the units for computer memory is due to the counting-of-states intuition for the entropy: a hard drive that can store 10^9 `bits` has 2^{10^9} unique states, with a priori equal plausibility for any particular arrangement of those bits, i.e. for any particular state of the system.

In this chapter, we'll use the natural base e, which is also very natural when studying the entropy of Gaussian distributions and nearly-Gaussian distributions.

[2] This connection between the uniform distribution over a finite number of states and the maximum of the entropy is an example of the *principle of maximum entropy* and descends from what's called Laplace's *principle of indifference*: without any other information, all the a priori probabilities for a system to be in any particular state should be equal. (As we will discuss in §A.3, when we do have some information about the state, then the entropy is no longer maximized by a uniform distribution.)

Note that this indifference principle is applied to macroscopic thermodynamic states of (nearly-)equal

To gain even more intuition, let's consider a perspective from information theory. In the entropy formula (A.1), the quantity inside the expectation is called the **surprisal**:

$$s(x) \equiv -\log p(x) = \log\left[\frac{1}{p(x)}\right] ; \tag{A.4}$$

since for a particular outcome, x, the distribution $p(x)$ quantifies the frequency or plausibility of observing x, and since the logarithm is monotonic in its argument, the surprisal grows with the a priori rareness of the outcome x, and hence quantifies how surprising or informative actually observing a particular x is. As such, the surprisal is also a quantitative measure of how much new **information** is gained after making such an observation. Averaging the surprisal over all possible outcomes, $\mathcal{X}$, gives the entropy (A.1), which can thus be understood as the expected surprise or amount of information to be gained from making an observation:

$$\mathcal{S}[p(x)] \equiv \mathbb{E}\left[s(x)\right] . \tag{A.5}$$

In addition to admitting various nice interpretations, the entropy also obeys a few nice mathematical properties. To start, it is manifestly nonnegative:

$$\mathcal{S}[p(x)] \geq 0 . \tag{A.6}$$

You can see this by noting that the allowed range of a probability, $0 \leq p(x) \leq 1$, implies that the nonnegativity of the quantity, $-p(x)\log p(x) \geq 0$, which in turn implies that the entropy (A.1) is a sum of nonnegative terms. Moreover, the entropy takes its minimum value and vanishes, (A.2), if and only if the distribution is perfectly ordered, given by a Kronecker delta for one particular outcome.

Another important property of the entropy is its *additivity* for two random variables $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ that are described by a factorized joint distribution, $p(x,y) = p(x)\,p(y)$, and thus are statistically independent (1.79):

$$\begin{aligned}
\mathcal{S}[p(x,y)] &= -\sum_{x\in\mathcal{X},y\in\mathcal{Y}} p(x,y)\log p(x,y) \\
&= -\sum_{x\in\mathcal{X},y\in\mathcal{Y}} p(x)p(y)\left[\log p(x) + \log p(y)\right] \\
&= -\sum_{x\in\mathcal{X}} p(x)\log p(x) - \sum_{y\in\mathcal{Y}} p(y)\log p(y) \\
&= \mathcal{S}[p(x)] + \mathcal{S}[p(y)] .
\end{aligned} \tag{A.7}$$

energy as a principal assumption of microscopic statistical mechanics, and the associated entropy, (A.3), is sometimes called the *Boltzmann entropy*. In contrast, the fully-general formula (A.1) is sometimes called the *Gibbs entropy* in the context of statistical mechanics.

Finally, for a Bayesian, this principle of indifference offers a natural way to pick a set of prior beliefs, and a prior distribution that respects this indifference principle is sometimes called a *non-informative prior*. While motivated in part by the Occam's razor heuristic, a Bayesian's *subjective* adoption of such a prior is very different from the automatic and objective embodiment of Occam's razor by the Bayes' factor, cf. our discussion of Bayesian model comparison in §6.2.2.

Intuitively, if two observables are independent, then the expected amount of information learned from making an observation of each is just the sum of the information learned from making each individual observation. For macroscopic systems with small probabilities for individual outcomes, this makes the entropy a practically useful quantity to work with: while the probability of independent outcomes multiply and create smaller and smaller numbers, the surprisals, and thus the entropies, simply add.

The additivity property (A.7) has a very physical interpretation: the entropy is typically an **extensive** quantity, meaning that it scales with the number of **degrees of freedom** or microscopic size of the system. This should make sense given the counting-of-states interpretation of the entropy: if a variable x describes the potential contents of a 1 TB hard drive, and the variable y independently describes the potential contents of another 1 TB hard drive, then the total storage capacity of the hard drives together is additive and equal to 2 TB. As the number of states available to the combined "hard drive" system is the product of the states of the individual "hard drive" systems, their entropies are additive.

However, if the hard drives are not independent and *constrained* so that one hard drive mirrors the other, then their total storage capacity is *subadditive* and instead would be equal to 1 TB.[3] In this case, the system had only half as many degrees of freedom as we naively thought it had due to the strict constraint creating strong correlations between the contents of the hard drives.

Thus, more generally for two statistically *dependent* observables constrained by a nonzero interaction between them, the entropy obeys a property called *subadditivity*:

$$\mathcal{S}[p(x,y)] < \mathcal{S}[p(x)] + \mathcal{S}[p(y)] \ . \tag{A.8}$$

In words, this means that the entropy of a joint distribution $p(x,y)$ will be always less than or equal to the sum of the entropies of the marginal distributions $p(x)$ and $p(y)$. This is also physically intuitive: if two random variables are statistically dependent, then an observation of x conveys information about the likely outcome of making an observation of y, and so an observation of y doesn't convey as much information as it would have if we didn't already know x.[4]

[3]This configuration has a practical realization called *RAID 1*, where "RAID" stands for *Redundant Array of Inexpensive Disks*, and allows for fault tolerance by creating data redundancy between the two hard drives.

[4]For the mathematically curious, we can turn this intuition into a quick proof. The usual route is to first prove the *Jensen inequality* and then apply it to an auxiliary object called the **Kullback–Leibler (KL) divergence**.

First let's state and then prove the Jensen inequality. Consider a discrete probability distribution over N elements $a_{\mu=1,\ldots,N}$ with corresponding probabilities p_μ such that $\sum_{\mu=1}^{N} p_\mu = 1$. Next, consider a convex functions $f(a)$, i.e. a function that satisfies

$$f\big(\lambda a_1 + (1-\lambda)a_2\big) \geq \lambda f(a_1) + (1-\lambda) f(a_2)\,, \tag{A.9}$$

for any $\lambda \in [0,1]$ and for any numbers a_1 and a_2 in the domain of the function. The Jensen inequality states that the expectation of such a function is greater than or equal to the function applied to the mean:

$$\mathbb{E}\left[f(a)\right] \geq f\big(\mathbb{E}\left[a\right]\big) \ . \tag{A.10}$$

Turning this argument upside down and shuffling (A.8) leftside right, let us define the **mutual information** between two random variables as:

$$\begin{aligned}\mathcal{I}\left[p(x,y)\right] &\equiv \mathcal{S}[p(x)] + \mathcal{S}[p(y)] - \mathcal{S}[p(x,y)] \\ &= \sum_{x\in\mathcal{X},y\in\mathcal{Y}} p(x,y)\log\left[\frac{p(x,y)}{p(x)\,p(y)}\right] .\end{aligned} \tag{A.14}$$

This is a functional of a joint probability distribution and gives an average measure of how much information an observation of $x \in \mathcal{X}$ conveys about an observation of $y \in \mathcal{Y}$, and vice versa. Rearranged in this way, we see that the subadditivity of the entropy (A.8) implies the nonnegativity of the mutual information,

$$\mathcal{I}\left[p(x,y)\right] \geq 0\,, \tag{A.15}$$

with equality holding when and only when the sets of observable outcomes, $\mathcal{X}$ and $\mathcal{Y}$, are

---

This can be proved by induction on N as follows. First, note that (A.10) holds trivially when $N = 1$. Then, see that

$$\begin{aligned}\mathbb{E}\left[f(a)\right] \equiv \sum_{\mu=1}^{N+1} p_\mu f(a_\mu) =& p_{N+1}f(a_{N+1}) + (1-p_{N+1})\sum_{\mu=1}^{N}\frac{p_\mu}{1-p_{N+1}}f(a_\mu) \\ \geq& p_{N+1}f(a_{N+1}) + (1-p_{N+1})\,f\left(\sum_{\mu=1}^{N}\frac{p_\mu}{1-p_{N+1}}a_\mu\right) \\ \geq& f\left(p_{N+1}a_{N+1} + (1-p_{N+1})\sum_{\mu=1}^{N}\frac{p_\mu}{1-p_{N+1}}a_\mu\right) = f(\mathbb{E}\left[a\right])\,,\end{aligned} \tag{A.11}$$

where in going from the first line to the second line we used the Jensen inequality (A.10) for N elements, and in going from the second line to the third line we used the convexity of the function (A.9).

As the next step, let us introduce the KL divergence (sometimes known as the *relative entropy*) between two *different* probability distributions $p(x)$ and $q(x)$ for a discrete variable, $x \in \mathcal{X}$,

$$KL\left[p(x)\,||\,q(x)\right] \equiv \sum_{x\in\mathcal{X}} p(x)\log\left[\frac{p(x)}{q(x)}\right] = -\mathcal{S}[p(x)] + \mathcal{S}[p(x),q(x)]\,, \tag{A.12}$$

which is a multi-function functional of both distributions, $p(x)$ and $q(x)$, and importantly is *not* symmetric in its function arguments. Here $\mathcal{S}[p(x),q(x)] \equiv -\sum_{x\in\mathcal{X}} p(x)\log q(x)$ is an asymmetric quantity known as the *cross entropy*. As first mentioned in footnote 16 of §10.2.3, the KL divergence is a asymmetric measure of the closeness of two different distributions and is closely related to the *cross-entropy loss* (10.36).

Finally, let's show that the KL divergence is nonnegative by applying the Jensen inequality to the convex function $f(a) = -\log(a)$ with discrete elements $a_\mu = q(x)/p(x)$:

$$KL\left[p(x)\,||\,q(x)\right] \equiv \sum_{x\in\mathcal{X}} p(x)\log\left[\frac{p(x)}{q(x)}\right] \geq -\log\left[\sum_{x\in\mathcal{X}} p(x)\frac{q(x)}{p(x)}\right] = -\log(1) = 0\,. \tag{A.13}$$

To complete our proof, note that the positivity of the KL divergence (A.13) implies the subadditivty of the entropy (A.8) for a choice of distributions as $KL\left[p(x,y)\,||\,p(x)\,p(y)\right]$.

statistically independent.[5] Thus, the mutual information of a joint distribution is telling us something about the *interactions* that create the nontrivial correlations that are a signature of statistical dependence. We'll compute explicitly the mutual information of preactivations for infinite-width neural networks in §A.2 and for finite-width neural networks in §A.3: as you might imagine, the interaction of neurons at finite width will lead to a nonzero mutual information.

As the last preparation before such computations, we will need to extend the definition of the entropy (A.1) from discrete outcomes to continuous random variables as

$$\mathcal{S}[p(x)] \equiv -\int dx\; p(x)\; \log p(x) = \mathbb{E}\left[s(x)\right] \; . \tag{A.16}$$

While the entropy is still the expectation of the surprisal (A.4), to take that expectation we now have to evaluate an integral rather than compute a sum. As passing to the continuum actually involves a physically interesting subtlety, let's take a few paragraphs to unravel this definition (A.16).

First, let's understand this subtlety mathematically. Since our random variable is continuous, we can make a smooth and monotonic change of coordinates in our continuous *outcome space* $\mathcal{X}$ from x to $\widetilde{x} \equiv \widetilde{x}(x)$. Since such coordinates are arbitrary, consistency requires that the probability of observing x in the interval between $[x_1, x_2]$ be the same as the probability of observing $\widetilde{x}$ in $[\widetilde{x}(x_1), \widetilde{x}(x_2)]$. In equations, this means that

$$p(x_1 < x < x_2) \equiv \int_{x_1}^{x_2} dx\; p(x) \tag{A.17}$$

must equal

$$p(\widetilde{x}_1 < \widetilde{x} < \widetilde{x}_2) \equiv \int_{\widetilde{x}(x_1)}^{\widetilde{x}(x_2)} d\widetilde{x}\; p(\widetilde{x}) = \int_{x_1}^{x_2} dx\; \frac{d\widetilde{x}}{dx}\; p\Big(\widetilde{x}(x)\Big) \, , \tag{A.18}$$

where in the last step we used the standard transformation property of the integral measure under a change of coordinates. Thus, the two distributions $p(\widetilde{x})$ and $p(x)$ must be related as

$$p\Big(\widetilde{x}(x)\Big) \equiv \frac{dx}{d\widetilde{x}}\; p(x) \, , \tag{A.19}$$

which is the well-known coordinate transformation formula for a probability density of a single variable. What about the expected surprisal? Under this same coordinate transformation, the entropy (A.16) is given by

$$\begin{aligned} \mathcal{S}[p(\widetilde{x})] &= -\int d\widetilde{x}\; p(\widetilde{x})\; \log p(\widetilde{x}) \\ &= -\int dx\; \frac{d\widetilde{x}}{dx}\; \frac{dx}{d\widetilde{x}}\; p(x)\; \left[\log p(x) + \log\left(\frac{dx}{d\widetilde{x}}\right)\right] \\ &= \mathcal{S}[p(x)] + \int dx\; p(x)\; \log\left(\frac{d\widetilde{x}}{dx}\right) \, , \end{aligned} \tag{A.20}$$

[5] Note that the mutual information (A.14) can also be thought of as the KL divergence (A.12) between the joint distribution $p(x, y)$ and the product of the marginal distributions $p(x)\, p(y)$. That is, the mutual information $\mathcal{I}\left[p(x, y)\right]$ tells us how close a joint distribution is to being a product of independent distributions, with a nonzero mutual information signaling statistical dependence.

where in the second line we again used the standard transformation property of the integral measure under a change of coordinates and we also used (A.19) to express the probability distribution in the old coordinates. So with a general (monotonic) change of coordinates, we can make the entropy take any value we'd like by a judicious choice of the Jacobian of the transformation, $d\widetilde{x}/dx$, even a negative one!

Now, let's understand the physical meaning of this subtlety. Let's consider a coordinate change that's just a multiplicative factor, $\widetilde{x} = cx$, which is like changing the base unit that we use to measure the observable x from `inches` to `meters`. In this case, the surprisal of each particular outcome shifts by the same constant, $\Delta s(x) = \log c$, and thus so does the entropy. Thus, for continuous quantities the entropy is additively sensitive to the choice of units with which we measure our observable quantities, and we really should specify these units along with the definition of the entropy (A.16).[6]

Perhaps the most sensible choice is to set the measuring units according to the smallest possible measurements of x that we can physically distinguish – in other words, according to the precision limit set by the constraints of the physical world. This precision limit ϵ is sometimes called a *cutoff*, and in practice means that we only care about the discrete probability of finding x between $[x_0, x_0 + \epsilon]$.[7] Such a discretization of the outcome space will then ensure that the entropy is always positive, (A.6), since we're now dealing with discrete probabilities again.[8]

If this physical sensitivity of the entropy to the somewhat arbitrary cutoff bothers

[6]For any *dimensionful* observable, the probability *density* $p(x)$ must also be dimensionful so that the probability of observing x in the interval $[x_1, x_2]$, $p(x_1 < x < x_2) \equiv \int_{x_1}^{x_2} dx\ p(x)$, is properly dimensionless. Yet, putting such a dimensionful object $p(x)$ into an argument of the logarithm as $\log p(x)$ is illegitimate, as we discussed in footnote 15 of §1. This is another way of seeing that for continuous probability distributions, i.e. probability densities, we need to specify the measuring units to properly define the entropy.

For the curious and potentially worried reader, it should be noted here that the *Bayes' factor* (6.30) that contained the observation dependence of our Bayesian model comparison is invariant under a coordinate transformation of our observations $y_{\mathcal{A}}$. What ultimately mattered there was the relative magnitude of the *evidence* of different hypotheses, and not the absolute value of the individual evidences.

Also, please do not confuse this issue of the units for a probability density, which change the entropy by an additive constant, with the units of entropy that we discussed in footnote 1, which change the entropy by a multiplicative constant. Note that for discrete probability distributions, the probability distribution gives a simple probability and is already dimensionless, cf. our brief discussion in footnote 1.

[7]Please don't confuse our *measurement precision cutoff* here with the *perturbative cutoff* of the effective theory, the depth-to-width ratio $r \equiv L/n$. While in both cases we can think of them as important scales, they have very different physical meanings: in the former case, the precision cutoff gives the minimum distinguishable difference between measurements of two observables, $\epsilon \equiv \min(|z_2 - z_1|)$; in the latter case, the cutoff of the effective theory, L/n, sets the scale at which finite-width corrections need to be taken into account in the preactivation distribution $p(z)$.

[8]In the context of deep learning, a natural choice is the precision limit of the floating-point representation of the network's variables. Since type `float` eponymously has a precision that's relative to the value being stored, one could perhaps choose the minimum precision in the relevant range.

you, then perhaps you should consider the continuous analog of mutual information,

$$\begin{aligned}\mathcal{I}\left[p(x,y)\right] &\equiv \mathcal{S}[p(x)] + \mathcal{S}[p(y)] - \mathcal{S}[p(x,y)] \\ &= \int dxdy\ p(x,y)\ \log\left[\frac{p(x,y)}{p(x)\,p(y)}\right]\,,\end{aligned} \tag{A.21}$$

where the definition in terms of the entropy is the same as in the discrete case, (A.14). In particular, mutual information is insensitive to the choice of the measuring coordinates. To see why, let's consider two continuous random variables x and y and independent monotonic coordinate transformations,

$$p\Big(\widetilde{x}(x)\Big) = \frac{dx}{d\widetilde{x}}\ p(x)\,, \qquad p\Big(\widetilde{y}(y)\Big) = \frac{dy}{d\widetilde{y}}\ p(y)\,, \qquad p\Big(\widetilde{x}(x),\widetilde{y}(y)\Big) = \frac{dx}{d\widetilde{x}}\frac{dy}{d\widetilde{y}}\ p(x,y)\,, \tag{A.22}$$

where to get this we applied a similar consistency-of-probabilities argument to the one that we gave above. With this in mind, we can now show that the mutual information (A.21) stays invariant under these coordinate transformations:

$$\begin{aligned}\mathcal{I}\left[p(\widetilde{x},\widetilde{y})\right] &\equiv \mathcal{S}[p(\widetilde{x})] + \mathcal{S}[p(\widetilde{y})] - \mathcal{S}[p(\widetilde{x},\widetilde{y})] \\ &= \int d\widetilde{x}d\widetilde{y}\ p(\widetilde{x},\widetilde{y})\ \log\left[\frac{p(\widetilde{x},\widetilde{y})}{p(\widetilde{x})\,p(\widetilde{y})}\right] \\ &= \int dxdy\ p(x,y)\ \log\left[\frac{p(x,y)}{p(x)\,p(y)}\right] \equiv \mathcal{I}\left[p(x,y)\right]\,.\end{aligned} \tag{A.23}$$

In going from the second line to the third line, the coordinate transformation factors $dx/d\widetilde{x}$ and $dy/d\widetilde{y}$ completely cancelled inside the logarithm, and the transformation of the measure cancelled the coordinate transformation factors outside the logarithm. Thus, the mutual information is completely well defined for continuous random variables, independent of the cutoff ϵ. For this reason, with a consistent and fixed choice of units, it's completely valid to compute the entropy as an intermediate step in the computation of the mutual information, and we will make use of this fact in the following sections.

Finally, note that the notion of the mutual information can be extended to more than two random variables. For instance, in §A.3 we will consider the **tripartite information** among three random variables $x \in \mathcal{X}$, $y \in \mathcal{Y}$, and $z \in \mathcal{Z}$:

$$\begin{aligned}&\mathcal{I}_3\left[p(x,y,z)\right] \\ \equiv&\mathcal{I}\left[p(x,y)\right] - \mathcal{I}\left[p(x,y|z)\right] \\ =&\mathcal{S}[p(x)] + \mathcal{S}[p(y)] + \mathcal{S}[p(z)] - \mathcal{S}[p(x,y)] - \mathcal{S}[p(y,z)] - \mathcal{S}[p(z,x)] + \mathcal{S}[p(x,y,z)] \\ =&\sum_{x\in\mathcal{X},y\in\mathcal{Y},z\in\mathcal{Z}} p(x,y,z)\ \log\left[\frac{p(x,y)\,p(y,z)\,p(z,x)}{p(x)\,p(y)\,p(z)\,p(x,y,z)}\right]\,.\end{aligned} \tag{A.24}$$

Here, $\mathcal{I}\left[p(x,y|z)\right]$ is the mutual information of the joint distribution between x and y conditioned on z, $p(x,y|z)$, and the final expression for $\mathcal{I}_3\left[p(x,y,z)\right]$ makes it clear that *(a)* it is fully symmetric in its three arguments, and that *(b)* its continuous analog

that's in your imagination is cutoff independent and invariant under similar coordinate transformations as those in (A.22).

What is not immediately obvious is that tripartite information can be either positive or negative. From the first expression in (A.24), we can gain some intuition for the meaning of the tripartite information: it is a measure of whether knowledge of a third random variable, z in this way of writing the expression, increases or decreases the mutual information between the other two variables. When positive, it indicates that z contains information about x and y, and so knowing z decreases the amount of information you'd learn about x by measuring y; the information is stored *redundantly* between these three variables. When negative, it indicates that the information is distributed across x, y, and z in such a way that you'd learn less about x by measuring y than you would with first knowing z; in this case, the information is stored *synergistically* between these three variables.[9]

A.2 Information at Infinite Width: Criticality

With that informative introduction out of the way, let's us now make these abstract definitions concrete by using them to better understand the neural-network prior distribution.

To begin, let's focus on m preactivations $z_{i;\alpha}^{(\ell)}$ from the ℓ-th layer of an infinite-width neural network. Depending on when you're coming to this appendix from the main text, it's probably ingrained in your mind already that such preactivations are distributed according to a zero-mean Gaussian distribution

$$p\Big(z_1, \ldots, z_m \Big| \mathcal{D}\Big) = \frac{1}{\sqrt{|2\pi K|^m}} \exp\left(-\frac{1}{2} \sum_{i=1}^{m} \sum_{\alpha_1, \alpha_2 \in \mathcal{D}} K^{\alpha_1 \alpha_2} z_{i;\alpha_1} z_{i;\alpha_2} \right) , \tag{A.25}$$

where in this expression, and in this section, we will drop layer indices everywhere to

[9] An extreme example of positive tripartite information occurs when x, y, and z are completely dependent and exact copies of each other. In this redundant case, $\mathcal{I}\left[p(x,y)\right] > 0$, since knowledge of y tells you everything about x, but $\mathcal{I}\left[p(x,y|z)\right] = 0$, since conditioning on z means that there's nothing left for you to learn about x by also observing y. An example of such a situation would be three copies of the same book.

An extreme example of negative tripartite information occurs when the joint distribution between x and y factorizes, $p(x,y) = p(x)\,p(y)$, but joint distribution conditioned on z does not, $p(x,y|z) \neq p(x|z)\,p(y|z)$. In this synergistic case, $\mathcal{I}\left[p(x,y)\right] = 0$, since without z the variables are statistically independent, but $\mathcal{I}\left[p(x,y|z)\right] > 0$, since there are correlations that are mediated by z. An example of such a situation could be a code that distributes a key among three different parties: knowledge of any two parts would give absolutely no information about the key, but with all three parts together the key can be reconstructed.

declutter expressions. The entropy (A.16) of this distribution is then given by

$$\begin{aligned}\mathcal{S}\Big[p\Big(z_1,\ldots,z_m\Big|\mathcal{D}\Big)\Big] =& \left\langle\!\!\left\langle \frac{1}{2}\sum_{i=1}^{m}\sum_{\alpha_1,\alpha_2\in\mathcal{D}} K^{\alpha_1\alpha_2} z_{i;\alpha_1} z_{i;\alpha_2} \right\rangle\!\!\right\rangle_K + \log\left(\sqrt{|2\pi K|^m}\right) \\ =& \frac{m}{2}\left[N_{\mathcal{D}} + \log\left(|2\pi K|\right)\right] = \frac{m}{2}\log\left(|2e\pi K|\right)\,,\end{aligned} \tag{A.26}$$

where as a reminder $|2\pi eK|$ is the determinant of the $N_{\mathcal{D}}$-by-$N_{\mathcal{D}}$ matrix $2\pi eK_{\alpha_1\alpha_2}$.[10] From this we conclude that entropy is *extensive*, proportional to the number of neurons m in the joint distribution (A.25). This shows how the entropy can count the *degrees of freedom* in a deep learning context – in this case, by counting the neurons – and the exact additivity in the number of neurons signals to us that the individual neurons in an infinite-width layer are non-interacting and statistically independent.

To confirm this directly, let's work out the mutual information between two sets of neurons, $\mathcal{M}_1 = \{1,\ldots,m_1\}$ and $\mathcal{M}_2 = \{m_1+1,\ldots,m_1+m_2\}$, in the same layer ℓ. In excruciating detail for its triviality, we have

$$\begin{aligned}&\mathcal{I}\left[p\Big(\mathcal{M}_1,\mathcal{M}_2\Big|\mathcal{D}\Big)\right] \\ =&\mathcal{S}\Big[p\Big(z_1,\ldots,z_{m_1}\Big|\mathcal{D}\Big)\Big] + \mathcal{S}\Big[p\Big(z_{m_1+1},\ldots,z_{m_1+m_2}\Big|\mathcal{D}\Big)\Big] - \mathcal{S}\Big[p\Big(z_1,\ldots,z_{m_1+m_2}\Big|\mathcal{D}\Big)\Big] \\ =&\left[m_1+m_2-(m_1+m_2)\right]\frac{1}{2}\log\left(|2e\pi K|\right) = 0\,.\end{aligned} \tag{A.27}$$

where to go to the last line we used (A.26) three different ways. This zero mutual information confirms that at infinite width learning the activities of some neurons in a layer conveys no information about the activities of any of the other neurons. To find a finite result, we'll have to back off the infinite-width limit (§A.3).

That said, for a fixed neuron we do expect nontrivial correlations between different inputs and therefore also a finite mutual information. To investigate this, let's take two inputs $x_{i;+}$ and $x_{i;-}$ and compute the mutual information between the preactivations for a particular neuron, $z_{1;+}$ and $z_{1;-}$. Plugging the entropy (A.26) into the definition of the mutual information (A.14), we find

$$\begin{aligned}\mathcal{I}\left[p(z_{1;+},z_{1;-}|x_\pm)\right] =& \frac{1}{2}\Big[\log(K_{++}) + \log(K_{--}) - \log\Big(K_{++}K_{--} - K_{+-}^2\Big)\Big] \\ =& \frac{1}{2}\log\left(\frac{K_{++}K_{--}}{K_{++}K_{--}-K_{+-}^2}\right)\,.\end{aligned} \tag{A.28}$$

Focusing on inputs $x_{i;\pm}$ with equal norms such that $K_{++} = K_{--} = K_{[0]} + K_{[2]}$ and

[10] In this and the next section, for convenience we will ignore the ambiguity in the definition of the continuous entropy and use the formula (A.16) naively. This is permissible since we ultimately are interested in cutoff-independent quantities, such as the mutual information, and when interpreting an entropy such as (A.26) we will never care about its absolute value.

$K_{+-} = K_{[0]} - K_{[2]}$, cf. (5.17) and (5.19), we can rewrite this mutual information as

$$\mathcal{I}\left[p(z_{1;+}, z_{1;-})\right] = \frac{1}{2}\log\left[\frac{\left(1+\frac{K_{[2]}}{K_{[0]}}\right)^2}{4\frac{K_{[2]}}{K_{[0]}}}\right] , \tag{A.29}$$

which is parameterized entirely by the dimensionless ratio $K_{[2]}/K_{[0]}$ that for nearby inputs $x_{i;\pm} = x_{i;0} \pm \delta x_i/2$ characterizes their relative angle after passing through the network to the ℓ-th layer.

There are two interesting limits here. First, when $K_{[2]}/K_{[0]} \to 0$, the mutual information becomes large. This follows because as the relative angle between the preactivations vanishes, they become close to each other: knowing the preactivation for one input tells us about the value of the preactivation for the other input. In a classification problem, this is a great prior if the inputs are from the same class, but would make learning really difficult if they're from different classes. Second, when $K_{[0]} = K_{[2]}$, the mutual information vanishes. This follows because in this limit the off-diagonal components of the kernel, $K_{+-} = K_{[0]} - K_{[2]}$, vanishes: the preactivations become statistically independent. In a classification problem, this may be a good prior if the inputs are from different classes – so long as the details of the classes don't correlate in some way – but would make learning really difficult if the inputs are from the same class. Altogether, this suggests that as a prior we don't want $K_{[2]}/K_{[0]}$ to be too big or too small, which can be best ensured by setting both $\chi_\parallel = 1$ and $\chi_\perp = 1$, cf. (5.55). This gives an information-theoretic perspective on *criticality*.

Finally, while we have focused here on the prior distribution for networks at initialization, we could also study these same quantities after learning using the Bayesian posterior (6.66) or the fully-trained distribution of gradient-based learning (10.39). Since the entropy of a Gaussian distribution is independent of its mean – it can be eliminated by a change of dummy integration variables – the mutual information for either trained infinite-width network is given by this same expression, (A.28), but with the kernel replaced by the covariance of the Bayesian posterior distribution (6.57) or the covariance of the generalized posterior distribution (10.41). In the latter case, the mutual information will involve both the kernel and the frozen NTK, and its analysis would yield similar results to those that we found in §10.3.1 when we investigated the bias-variance tradeoff. This would give an information-theoretic perspective on generalization.[11]

A.3 Information at Finite Width: Optimal Aspect Ratio

In this section, we'll see how finite-width networks have a prior for nonzero mutual information between different neurons. Assuming that nonzero mutual information is desirable by intuition – and by analogy to an *unsupervised* learning objective – we can use this computation to optimize the depth-to-width ratio for finite-width MLPs at

[11] Analogously, studying the tripartite information generalization of (A.28) for either type of posterior distribution would give an alternative perspective on the $\star$-polation results of §10.3.2.

criticality. This *optimal aspect ratio*, $r^\star$, defines the scale that separates effectively-deep networks – describable by our effective theory approach – from overly-deep networks – not describable due to strong interactions and not trainable due to large fluctuations.[12]

In general, the entropy and mutual information of any interacting theory are really difficult to compute. However, when the interactions are *nearly-Gaussian*, we can use perturbation theory. To do so, there is actually a neat *variational principle* that we can use to organize our calculations, so let us explain that first.

As in the last section, we'll focus on the distribution of m preactivations in layer ℓ, drop (almost) all the layer indices, and focus exclusively on a single input x. As we have been doing since the beginning of time (§1.3), let's express the probability distribution in terms of an action as

$$p(z_1, \ldots, z_m|x) = \frac{e^{-S(z_1,\ldots,z_m)}}{Z}\,, \tag{A.30}$$

which must be normalized by the *partition function*

$$Z = \int \left[\prod_{i=1}^{m} dz_i\right] e^{-S(z_1,\ldots,z_m)}\,. \tag{A.31}$$

In terms of these quantities, the entropy (A.16) can be expressed as

$$\mathcal{S}[p(z_1, \ldots, z_m|x)] = \log(Z) + \frac{1}{Z}\int \left[\prod_{i=1}^{m} dz_i\right] e^{-S(z_1,\ldots,z_m)} S(z_1, \ldots, z_m)\,. \tag{A.32}$$

Just to make sure, please don't get confused between the action $S(z)$, which is a function of the preactivations, and the entropy $\mathcal{S}[p(z)]$, which is a functional of the probability distribution.[13]

To proceed, let's adopt a **variational ansatz**: we'll divide the action into two parts as

$$S(z_1, \ldots, z_m) = S_{\text{F}}(z_1, \ldots, z_m) + S_{\text{I}}(z_1, \ldots, z_m)\,, \tag{A.33}$$

with the idea being that the second term, S_{I}, encodes the part of the distribution that is perturbatively small. Specifically, the **variational principle** instructs us to choose the first term in (A.33), $S_{\text{F}}(z)$, that gives no variations of the entropy with respect to $S_{\text{I}}(z)$:

$$0 = \left.\frac{\delta \mathcal{S}[p(z)]}{\delta S_{\text{I}}(z)}\right|_{S_{\text{I}}(z)=0}\,. \tag{A.34}$$

[12] Note that in this section, we'll need the expressions for the running couplings that we derived in §4.4 when finding the effective distribution of m neurons in a wide-but-finite layer. As such, it may be helpful to reread that section before proceeding.

[13] For choices of units of the preactivations z for which the partition function is unity, $Z = 1$, the action is simply the surprisal (A.4), and thus the entropy is the expectation of the action: $\mathcal{S}[p(z)] = \mathbb{E}\left[S(z)\right]$. Note that in thermodynamics, the constant $-\log Z$ is sometimes called the *free energy*; this discussion should make clear that only its relative value for two different distributions is physical.

We'll satisfy this shortly. Additionally, $S_{\mathrm{I}}(z)$ should not be completely arbitrary, but instead be constructed to properly reflect the statistics of the preactivation distribution $p(z_1, \ldots, z_m|x)$. The first such constraint comes from demanding that the two-point correlator, when computed with the variational action, is determined by the *exact* single-input metric G:

$$\mathbb{E}\left[z_{i_1} z_{i_2}\right] \equiv \delta_{i_1 i_2} G \,. \tag{A.35}$$

The second constraint comes from demanding that the connected four-point correlator, when computed with the variational action, is determined by the *exact* single-input four-point vertex:

$$\mathbb{E}\left[z_{i_1} z_{i_2} z_{i_3} z_{i_4}\right]|_{\text{connected}} = \frac{1}{n}\left(\delta_{i_1 i_2}\delta_{i_3 i_4} + \delta_{i_1 i_3}\delta_{i_2 i_4} + \delta_{i_1 i_4}\delta_{i_2 i_3}\right) V \,. \tag{A.36}$$

Together, the constraints (A.35) and (A.36) will fix the couplings of the variation action S_{I} so that the full action (A.33) correctly specifies the m-neuron preactivation distribution (A.30).[14]

To understand why we're doing this, note that our variational principle is ultimately just a realization of the **principle of maximum entropy** for nearly-Gaussian distributions.[15] In particular, first note that the Gaussian distribution itself can be derived as a distribution that maximizes the entropy (A.16), subject to the constraints of fixed first and second moment.[16] As we will see in a moment, in (A.34) we are maximizing the entropy of the distribution with respect to the deformation of the action away from Gaussianity, S_{I}, subject to the constraint of fixing the higher-order cumulants order by order in perturbation theory. In general, the maximum entropy principle is an appropriate procedure when we have fixed observable information for which we want to find an underlying distribution. Here, we actually know the distribution that produces G

[14] In principle there are additional constraints coming from the statistics of higher-point correlators, but their contribution is subleading to both the leading and next-to-leading orders that we will work.

[15] For those readers that enjoy our historical asides, the maximum entropy principle was discovered by Jaynes and provides a link between *information theory* on the one hand and *statistical mechanics* on the other hand [78, 79]. As an example of this, consider a central problem in statistical mechanics: find the probability distribution p_i for the fundamental *microstates* i of a system that has a macroscopic average energy $\overline{E} \equiv \mathbb{E}\left[E_i\right] = \sum_i p_i E_i$. An application of the principle of maximum entropy then correctly picks out the *Boltzmann distribution* (or sometimes, the *Gibbs distribution*) of statistical mechanics

$$p_i \propto e^{-\beta E_i} \,, \tag{A.37}$$

if we optimize the entropy (A.1) subject to the observable constraint for the energy, $\sum_i p_i E_i = \overline{E}$, and the normalization condition for the distribution, $\sum_i p_i = 1$. Here, β is a Lagrange multiplier that depends on the energy $\overline{E}$ and has a physical interpretation as the inverse temperature, $\beta = 1/(k_{\mathrm{B}} T)$, with the aforementioned Boltzmann constant k_{B} and the familiar-to-everyone temperature T. This example also shows how statistical mechanics links the details of the fundamental microstates i to the macroscopic thermodynamic variables such as $\overline{E}$ and T.

[16] For those of you keeping track: the maximum entropy distribution with no information fixed is the uniform distribution, cf. (A.3); the maximum entropy distribution with a fixed first moment is the Boltzmann distribution, cf. (A.37); and the maximum entropy distribution with a fixed first and second moment is the Gaussian distribution, cf. (nearly-)everywhere.

and V, (A.30), but we can still use this principle as convenient tool for computing the entropy.[17]

Now, to satisfy the variational principle (A.34), let's choose

$$S_{\rm F}(z_1,\dots,z_m) = \frac{1}{2G}\sum_{i=1}^{m} z_i^2\,. \tag{A.38}$$

Importantly, G is not just the inverse of the quadratic coefficient in the action $S(z)$, but instead is the *exact* two-point correlator that we would actually measure, (A.35), incorporating the full series of corrections due to the interactions, cf. (4.104).[18] To see why such a choice satisfies the variational principle, let us start by rewriting expectations with respect to the full distribution (A.30) in terms of simpler *Gaussian* expectations taken with respect to a zero-mean Gaussian distribution with the same variance (A.35):

$$\langle\!\langle z_{i_1} z_{i_2} \rangle\!\rangle_G = \delta_{i_1 i_2} G\,. \tag{A.39}$$

Here, please recall our notation $\langle\!\langle \cdot \rangle\!\rangle_G$ for a Gaussian expectation of a multi-neuron function with variance $\delta_{i_1 i_2} G$,

$$\langle\!\langle f(z_1,\dots,z_m) \rangle\!\rangle_G \equiv \frac{1}{Z_G}\int \left[\prod_{i=1}^{m} dz_i\right] e^{-\frac{1}{2G}\sum_{i=1}^{m} z_i^2} f(z_1,\dots,z_m)\,, \tag{A.40}$$

and note also that such a Gaussian distribution will require a different partition function

$$Z_G \equiv \int \left[\prod_{i=1}^{m} dz_i\right] e^{-\frac{1}{2G}\sum_{i=1}^{m} z_i^2} = (2\pi G)^{\frac{m}{2}}\;; \tag{A.41}$$

importantly, $Z \neq Z_G$. Next, let us rewrite the entropy (A.32) in terms of these simpler expectations as

$$\begin{aligned}\mathcal{S}[p(z_1,\dots,z_m|x)] =& \log Z + \mathbb{E}\left[\frac{1}{2G}\sum_{i=1}^{m} z_i^2\right] + \frac{1}{Z}\int \left[\prod_{i=1}^{m} dz_i\right] e^{-\frac{1}{2G}\sum_{i=1}^{m} z_i^2} e^{-S_{\rm I}} S_{\rm I} \\ =& \frac{m}{2}\log(2\pi G) + \log\left(\frac{Z}{Z_G}\right) + \frac{m}{2} + \left(\frac{Z}{Z_G}\right)^{-1} \left\langle\!\!\left\langle e^{-S_{\rm I}} S_{\rm I} \right\rangle\!\!\right\rangle_G\,, \end{aligned} \tag{A.42}$$

where in the first equality we just plugged in (A.33), and in the second equality we rewrote all the full expectations in terms of the simpler Gaussian expectations using (A.40). Then, by Taylor-expanding in $S_{\rm I}$, we can evaluate the ratio of partition functions as

$$\begin{aligned}\frac{Z}{Z_G} =& \frac{1}{Z_G}\int \left[\prod_{i=1}^{m} dz_i\right] e^{-\frac{1}{2G}\sum_{i=1}^{m} z_i^2} \left(e^{-S_{\rm I}}\right) = \left\langle\!\!\left\langle e^{-S_{\rm I}} \right\rangle\!\!\right\rangle_G \\ =& 1 - \langle\!\langle S_{\rm I} \rangle\!\rangle_G + \frac{1}{2}\left\langle\!\!\left\langle S_{\rm I}^2 \right\rangle\!\!\right\rangle_G - \frac{1}{6}\left\langle\!\!\left\langle S_{\rm I}^3 \right\rangle\!\!\right\rangle_G + O\left(S_{\rm I}^4\right)\,, \end{aligned} \tag{A.43}$$

[17] In particular, this procedure will organize our perturbative computation of the entropy and ensure that we only need to compute Gaussian expectations of the form (A.40).

[18] This will let us express the entropy in terms of these measurable quantities and in no way will we need to actually compute any of the corrections in this series.

and similarly we can evaluate the needed Gaussian expectation as

$$\left\langle\!\!\left\langle e^{-S_{\rm I}} S_{\rm I} \right\rangle\!\!\right\rangle_G = \langle\!\langle S_{\rm I} \rangle\!\rangle_G - \left\langle\!\!\left\langle S_{\rm I}^2 \right\rangle\!\!\right\rangle_G + \frac{1}{2}\left\langle\!\!\left\langle S_{\rm I}^3 \right\rangle\!\!\right\rangle_G + O\Big(S_{\rm I}^4\Big) \ . \tag{A.44}$$

Plugging these back into the variational expression for the entropy (A.42) and organizing terms, for which you might find $\log(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} + \ldots$ and $1/(1+x) = 1 - x + x^2 - x^3 + \ldots$ helpful, we get

$$\begin{aligned} \mathcal{S}[p(z_1,\ldots,z_m|x)] =& \frac{m}{2}\log(2\pi e G) - \frac{1}{2}\left[\left\langle\!\!\left\langle S_{\rm I}^2 \right\rangle\!\!\right\rangle_G - \langle\!\langle S_{\rm I}\rangle\!\rangle_G^2\right] \\ &+ \frac{1}{3}\left[\left\langle\!\!\left\langle S_{\rm I}^3 \right\rangle\!\!\right\rangle_G - 3\left\langle\!\!\left\langle S_{\rm I}^2 \right\rangle\!\!\right\rangle_G \langle\!\langle S_{\rm I}\rangle\!\rangle_G + 2\langle\!\langle S_{\rm I}\rangle\!\rangle_G^3\right] + O\Big(S_{\rm I}^4\Big) \ . \end{aligned} \tag{A.45}$$

First, note that the first term is exactly the entropy for a multivariate Gaussian distribution with a covariance $\delta_{i_1 i_2} G$, cf. (A.26). Second, and most importantly, note that the would-be linear term proportional to $\langle\!\langle S_{\rm I}\rangle\!\rangle_G$ exactly cancelled out. In other words, our ansatz for the decomposition of the action (A.33) with the choice (A.38) automatically satisfies the variational principle (A.34).

Finally, let us note in passing that the leading correction coming from the quadratic term is definitively negative.[19] This negativity is actually necessary for mathematical consistency: as the Gaussian distribution maximizes the entropy of any set of random variables with known and fixed first and second moments, any deformation of a distribution away from Gaussianity, while also respecting such moment constraints, must necessarily have less entropy. In the current case, our variational ansatz (A.33) gives a nearly-Gaussian deformation of a zero-mean Gaussian distribution.[20]

Now that we're done passing notes, let's satisfy our constraints, (A.35) and (A.36), and then use our variational expression (A.45) to compute the entropy and mutual information of the preactivations in a finite-width network.

Leading-Order Correction: Nonzero Mutual Information

At leading order, we've already worked out how to relate the couplings of the variational action $S_{\rm I}$ to the single-input metric G and the single-input four-point correlator V. Recall from our discussion of the running couplings when integrating out neurons in §4.4 that the leading correction to the action was given by

$$S_{\rm I}(z_1,\ldots,z_m) = \frac{1}{2}\left(g_m - \frac{1}{G}\right)\sum_{i=1}^m z_i^2 - \frac{v_m}{8}\sum_{i,j=1}^m z_i^2 z_j^2 + O\left(\frac{1}{n^2}\right) \ , \tag{A.46}$$

where the running quadratic coupling was given by (4.102),

$$g_m = \frac{1}{G} + \left(\frac{m+2}{2}\right)\frac{V}{nG^3} + O\left(\frac{1}{n^2}\right) \ , \tag{A.47}$$

[19] To see this, notice that the expression in the square brackets of the quadratic term is the variance of the variational part of the action, and thus is positive: $\left\langle\!\!\left\langle S_{\rm I}^2 \right\rangle\!\!\right\rangle_G - \langle\!\langle S_{\rm I}\rangle\!\rangle_G^2 = \langle\!\langle \big(S_{\rm I} - \langle\!\langle S_{\rm I}\rangle\!\rangle_G\big)^2 \rangle\!\rangle_G \geq 0$.

[20] N.B. the terms in the second set of the square brackets in (A.45) are necessary for computing the next-to-leading-order correction.

and the running quartic coupling was given by (4.103),

$$v_m = \frac{V}{nG^4} + O\Big(\frac{1}{n^2}\Big) \,. \tag{A.48}$$

To get the expression (A.46), look at our expression for the distribution of m preactivations, (4.97), and then rearrange (A.33) with (A.38) to solve for $S_{\rm I}$. If you don't recall how to get (A.47) and (A.48), feel free to flip back and reread the last subsubsection of §4.4, or feel free to flip forward to the next subsection where we'll have to derive these expressions again to higher order in the $1/n$ expansion, cf. (A.59) and (A.63).

Given that this leading-order variational correction to the action $S_{\rm I}$ (A.46) now satisfies the constraints (A.35) and (A.36), we can now evaluate the leading correction to the entropy (A.45):

$$\begin{aligned}
&\Big\langle\!\Big\langle S_{\rm I}^2 \Big\rangle\!\Big\rangle_G - \langle\!\langle S_{\rm I} \rangle\!\rangle_G^2 \qquad\qquad (A.49)\\
=&\frac{1}{4}\left(g_m - \frac{1}{G}\right)^2 \sum_{i_1,i_2=1}^{m} \Big[\Big\langle\!\Big\langle z_{i_1}^2 z_{i_2}^2 \Big\rangle\!\Big\rangle_G - \Big\langle\!\Big\langle z_{i_1}^2 \Big\rangle\!\Big\rangle_G \Big\langle\!\Big\langle z_{i_2}^2 \Big\rangle\!\Big\rangle_G\Big]\\
&- \frac{1}{8}\left(g_m - \frac{1}{G}\right) v_m \sum_{i_1,i_2,i_3=1}^{m} \Big[\Big\langle\!\Big\langle z_{i_1}^2 z_{i_2}^2 z_{i_3}^2 \Big\rangle\!\Big\rangle_G - \Big\langle\!\Big\langle z_{i_1}^2 z_{i_2}^2 \Big\rangle\!\Big\rangle_G \Big\langle\!\Big\langle z_{i_3}^2 \Big\rangle\!\Big\rangle_G\Big]\\
&+ \frac{1}{64} v_m^2 \sum_{i_1,i_2,i_3,i_4=1}^{m} \Big[\Big\langle\!\Big\langle z_{i_1}^2 z_{i_2}^2 z_{i_3}^2 z_{i_4}^2 \Big\rangle\!\Big\rangle_G - \Big\langle\!\Big\langle z_{i_1}^2 z_{i_2}^2 \Big\rangle\!\Big\rangle_G \Big\langle\!\Big\langle z_{i_3}^2 z_{i_4}^2 \Big\rangle\!\Big\rangle_G\Big] + O\Big(\frac{1}{n^3}\Big)\\
=&\frac{m}{2}\left(g_m - \frac{1}{G}\right)^2 G^2 - \frac{m(m+2)}{2}\left(g_m - \frac{1}{G}\right) v_m G^3 + \frac{m(m+2)(m+3)}{8} v_m^2 G^4 + O\Big(\frac{1}{n^3}\Big) \,.
\end{aligned}$$

Here, in going from the second line to the last line, you may find this formula for these Gaussian expectations useful:

$$\sum_{i_1,\ldots,i_k=1}^{m} \Big\langle\!\Big\langle z_{i_1}^2 \cdots z_{i_k}^2 \Big\rangle\!\Big\rangle_G = m(m+2)\cdots[m+2(k-1)]\, G^k \,, \tag{A.50}$$

which is akin to (3.46) and will be used again in the next subsubsection repeatedly, up to $k = 6$. To complete the computation, plug in the quadratic coupling, (A.47), and the quartic coupling, (A.48), into (A.49), giving

$$\Big\langle\!\Big\langle S_{\rm I}^2 \Big\rangle\!\Big\rangle_G - \langle\!\langle S_{\rm I} \rangle\!\rangle_G^2 = \frac{m(m+2)}{8}\left(\frac{V}{nG^2}\right)^2 + O\Big(\frac{1}{n^3}\Big) \,, \tag{A.51}$$

for the variance of the variational action. Therefore, the entropy (A.45) is given by

$$\mathcal{S}[p(z_1,\ldots,z_m|x)] = \frac{m}{2}\log(2\pi e G) - \frac{(m^2+2m)}{16}\left(\frac{V}{nG^2}\right)^2 + O\Big(\frac{1}{n^3}\Big) \,, \tag{A.52}$$

exhibiting a nontrivial correction at finite width.

Let us reflect on this formula by making some comments. First, note that the correction is definitely negative, as we pointed out before: recall that the Gaussian distribution maximizes the entropy of a set of random variables with a fixed covariance, and since our first variational constraint (A.35) fixes the two-point correlator of the preactivations, the entropy for the nearly-Gaussian distribution (A.30) must be less than the entropy of a Gaussian distribution with the same variance. Second, note that unlike all our previous results, the leading correction here is *second order* in the inverse layer width as $\sim V^2/n^2$ and *not* $\sim V/n$. Indeed, since the quartic coupling v_m in a generic nearly-Gaussian action can take either sign – corresponding to a distribution with either fat tails or thin tails – the leading correction to the entropy must be proportional to the minus the *square* of the quartic coupling, $v_m^2 \propto V^2/n^2$, in order to guarantee that the entropy decreases.[21] Finally, we see that this correction breaks the perfect additivity for the neurons that we found in the infinite-width limit (A.26). In particular, although the decrease in the entropy is perturbatively small with an order $1/n^2$ scaling, it also depends *quadratically* on m. This nonlinearity in the number of neurons signals the presence of nontrivial interactions at finite width.

Accordingly, we can characterize this statistical dependence by computing mutual information between two non-overlapping sets of neurons, $\mathcal{M}_1 = \{1, \ldots, m_1\}$ and $\mathcal{M}_2 = \{m_1 + 1, \ldots, m_1 + m_2\}$,

$$\begin{aligned} \mathcal{I}\left[p(\mathcal{M}_1, \mathcal{M}_2|x)\right] \equiv & \mathcal{S}[p(\mathcal{M}_1|x)] + \mathcal{S}[p(\mathcal{M}_2|x)] - \mathcal{S}[p(\mathcal{M}_1, \mathcal{M}_2|x)] \\ = & \frac{m_1 m_2}{8} \left[\frac{V^{(\ell)}}{n_{\ell-1}\left(G^{(\ell)}\right)^2}\right]^2 + O\left(\frac{1}{n^3}\right) , \end{aligned} \tag{A.53}$$

where we used our entropy formula (A.52) in three different ways and also restored layer indices to better interpret this formula. This nonzero mutual information signifies that – at finite width, *only* – for a given layer ℓ, observing the activities of a group of neurons $\mathcal{M}_1$ will convey information about the activities of another group of neurons $\mathcal{M}_2$. This can be thought of as an information-theoretic reformulation and generalization of the *Hebbian learning* principle that we saw for the conditional variance (6.78) in §6.4.1: there we more simply saw that the variance of one neuron $z_2^{(\ell)}$, conditioned on an atypical observation of a second neuron, $z_1^{(\ell)} = \check{z}_1^{(\ell)}$, will itself be atypical; here, we can directly characterize how much one group of neurons can know about another non-overlapping group.[22]

Finally, remembering our *scaling law* for the normalized vertex (5.128), we see that the mutual information (A.53) scales with the depth ℓ of the hidden layer *squared*:

$$\mathcal{I}\left[p(\mathcal{M}_1, \mathcal{M}_2|x)\right] \propto \ell^2/n^2 \, . \tag{A.54}$$

[21] This same argument excludes a contribution of the form $O(u_m) = O\big(U/n^2\big)$ coming from the sextic coupling, cf. (A.55), and similarly excludes any linear contributions from other higher-order couplings.

[22] More generally, it would be interesting to work out the mutual information for multiple inputs in order to understand its data dependence. In that case, we expect it to be mediated by the multi-input four-point vertex and depend on the details of different groupings of four samples from the dataset $\mathcal{D}$.

In terms of *RG flow*, this means that the mutual information is *relevant*, suggesting that a growing mutual information is helpful when coarse-graining representations: as the fine-grained features are marginalized over, deeper hidden layers will have a growing set of correlations between groups of neurons. In other words, by increasing the size of the nonlinear subadditive term in the entropy, this reduces the number of independently available degrees of freedom in these deeper layers.[23]

NLO Correction: Optimal Aspect Ratio and Tripartite Information

Our leading-order result for the finite-width mutual information, (A.54), naively grows in depth without bounds, suggesting that deeper is always better. Of course, if the depth becomes too large, this naive answer breaks down as the higher-order terms in the perturbation series start to become important. To understand the mutual information at even greater depths, we'll need to compute the *next-to-leading-order* (NLO) correction.

To push our calculations to the next level, we need to ensure that our two variational constraints, (A.35) and (A.36), are satisfied to next-to-leading-order, $O(1/n^2)$. In principle, this means that we will need to also include an $O(1/n^2)$ *sextic* term in our variational action $S_{\rm I}$ as

$$S_{\rm I}(z_1,\ldots,z_m) = \frac{1}{2}\left(g_m - \frac{1}{G}\right)\sum_{i=1}^{m} z_i^2 - \frac{1}{8}v_m \sum_{i,j=1}^{m} z_i^2 z_j^2 + \frac{1}{24}u_m \sum_{i,j,k=1}^{m} z_i^2 z_j^2 z_k^2 + O\left(\frac{1}{n^3}\right) . \tag{A.55}$$

Such a sextic term was originally introduced in (ε.18); here, we've specialized it to focus on m preactivations in a layer ℓ, with u_m the running sextic coupling.

Now, let's explain how to explicitly satisfy the variational constraints. First, just as we did before for the entropy in (A.45), note that for a general observable we can express its full expectation in terms of simpler Gaussian expectations as

$$\begin{aligned}\mathbb{E}\left[\mathcal{O}\right] \equiv & \frac{1}{Z}\int\left[\prod_{i=1}^{m} dz_i\right] e^{-S}\mathcal{O} = \left(\frac{Z}{Z_G}\right)^{-1}\left\langle\!\!\left\langle e^{-S_{\rm I}}\mathcal{O}\right\rangle\!\!\right\rangle_G \\ & = \left\langle\!\!\left\langle \mathcal{O}\right\rangle\!\!\right\rangle_G - \left[\left\langle\!\!\left\langle \mathcal{O}S_{\rm I}\right\rangle\!\!\right\rangle_G - \left\langle\!\!\left\langle \mathcal{O}\right\rangle\!\!\right\rangle_G \left\langle\!\!\left\langle S_{\rm I}\right\rangle\!\!\right\rangle_G\right] \\ & \quad + \frac{1}{2}\left[\left\langle\!\!\left\langle \mathcal{O}S_{\rm I}^2\right\rangle\!\!\right\rangle_G - 2\left\langle\!\!\left\langle \mathcal{O}S_{\rm I}\right\rangle\!\!\right\rangle_G \left\langle\!\!\left\langle S_{\rm I}\right\rangle\!\!\right\rangle_G - \left\langle\!\!\left\langle \mathcal{O}\right\rangle\!\!\right\rangle_G \left\langle\!\!\left\langle S_{\rm I}^2\right\rangle\!\!\right\rangle_G + 2\left\langle\!\!\left\langle \mathcal{O}\right\rangle\!\!\right\rangle_G \left\langle\!\!\left\langle S_{\rm I}\right\rangle\!\!\right\rangle_G^2\right] + O\left(\frac{1}{n^3}\right) ,\end{aligned} \tag{A.56}$$

where in the last equality, we expanded $\left\langle\!\!\left\langle e^{-S_{\rm I}}\mathcal{O}\right\rangle\!\!\right\rangle_G$ in $S_{\rm I}$ and also used the formula that we already evaluated in (A.43) for the ratio of partition functions. Physically, the first square brackets says that in an interacting theory, the leading variational correction to an observable is given by the correlation of the observable with the variational part of the action.

[23] It would be interesting to try to interpret this in terms of the *optimal brain damage* of [80] or the *lottery ticket hypothesis* of [81].

To satisfy our first constraint for the metric, (A.35), we can use this general expression (A.56) with the observable

$$\mathcal{O} \equiv \frac{1}{m}\sum_{i=1}^{m} z_i^2\,, \tag{A.57}$$

for which we can use our constraint (A.35) to easily see that $\mathbb{E}\left[\mathcal{O}\right] = G$. Evaluating this same expectation using the variational sextic action, (A.55), we find

$$\begin{aligned} G =& \frac{1}{m}\sum_{i=1}^{m}\mathbb{E}\left[z_i^2\right] \\ =& G - \left(g_m - \frac{1}{G}\right)G^2 + \frac{(m+2)}{2}v_m G^3 + \left(g_m - \frac{1}{G}\right)^2 G^3 - \frac{(m+2)(m+4)}{4}u_m G^4 \\ & - \frac{3(m+2)}{2}\left(g_m - \frac{1}{G}\right)v_m G^4 + \frac{(m+2)(m+3)}{2}v_m^2 G^5 + O\left(\frac{1}{n^3}\right)\,. \end{aligned} \tag{A.58}$$

In evaluating this, you might again find the formula (A.50) helpful. This expression, (A.58), shows how the interacting theory modifies the two-point correlator. Rearranging terms to solve for g_m order by order, we find an NLO version of (A.47), which determines the variational quadratic coupling in terms of the metric and the other higher-order couplings:

$$g_m = \frac{1}{G} + \frac{(m+2)}{2}v_m G - \frac{(m+2)(m+4)}{4}u_m G^2 + \frac{(m+2)}{2}v_m^2 G^3 + O\left(\frac{1}{n^3}\right)\,. \tag{A.59}$$

Next, to satisfy the second constraint for the four-point vertex, (A.36), we can use our general expression (A.56) with the observable

$$\mathcal{O} \equiv \frac{1}{m(m+2)}\sum_{i,j=1}^{m} z_i^2 z_j^2\,, \tag{A.60}$$

for which we can use both our constraints (A.35) and (A.36) to show that $\mathbb{E}\left[\mathcal{O}\right] = G^2 + V/n$. Now evaluating $\mathcal{O}$ according to the variational sextic action, (A.55), we get

$$\begin{aligned} G^2 + \frac{V}{n} =& \frac{1}{m(m+2)}\sum_{i,j=1}^{m}\mathbb{E}\left[z_i^2 z_j^2\right] \\ =& G^2 - 2\left(g_m - \frac{1}{G}\right)G^3 + 3\left(g_m - \frac{1}{G}\right)^2 G^4 + (m+3)v_m G^4 - \frac{(m+4)^2}{2}u_m G^5 \\ & - 4(m+3)\left(g_m - \frac{1}{G}\right)v_m G^5 + \frac{(5m^2+34m+60)}{4}v_m^2 G^6 + O\left(\frac{1}{n^3}\right) \\ =& G^2 + v_m G^4 - (m+4)u_m G^5 + \frac{(m+8)}{2}v_m^2 G^6 + O\left(\frac{1}{n^3}\right)\,, \end{aligned} \tag{A.61, A.62}$$

where to get to the second line we again made heavy use of our formula (A.50), and to get to the final line we plugged in (A.59) for the quadratic coupling. Rearranging terms

to solve for v_m order by order, we find an NLO version of (A.48), which determines the variational quartic coupling in terms of the two constraints and the sextic coupling:

$$v_m = \frac{V}{nG^4} - \frac{(m+8)}{2}\left(\frac{V}{nG^3}\right)^2 + (m+4)u_m G + O\left(\frac{1}{n^3}\right) . \tag{A.63}$$

We now finally have all the pieces we need to evaluate the NLO correction to the entropy (A.45).[24] First, let's reevaluate the variance of the variational action to NLO by repeated use of the formula (A.50),

$$\begin{aligned}
&\left\langle\!\!\left\langle S_{\rm I}^2 \right\rangle\!\!\right\rangle_G - \langle\!\langle S_{\rm I} \rangle\!\rangle_G^2 \qquad\qquad (A.64)\\
=&\frac{m}{2}\left(g_m - \frac{1}{G}\right)^2 G^2 - \frac{m(m+2)}{2}\left(g_m - \frac{1}{G}\right) v_m G^3 + \frac{m(m+2)(m+3)}{8} v_m^2 G^4 \\
&+ \frac{m(m+2)(m+4)}{4}\left(g_m - \frac{1}{G}\right) u_m G^4 - \frac{m(m+2)(m+4)^2}{8} v_m u_m G^5 + O\left(\frac{1}{n^4}\right) \\
=&\frac{m(m+2)}{8}\left[\left(v_m G^2\right)^2 - 2(m+4)\left(v_m G^2\right)\left(u_m G^3\right)\right] + O\left(\frac{1}{n^4}\right) \\
=&\frac{m(m+2)}{8}\left[\left(\frac{V}{nG^2}\right)^2 - (m+8)\left(\frac{V}{nG^2}\right)^3\right] + O\left(\frac{1}{n^4}\right) .
\end{aligned}$$

Here, to get to the penultimate line we used our NLO expression for the quadratic coupling (A.59), and then to get to the final line we used our NLO expression for the quartic coupling (A.63). Second, let's similarly evaluate the subleading term in our expression (A.45) for the entropy:

$$\begin{aligned}
&\left\langle\!\!\left\langle S_{\rm I}^3 \right\rangle\!\!\right\rangle_G - 3\left\langle\!\!\left\langle S_{\rm I}^2 \right\rangle\!\!\right\rangle_G \langle\!\langle S_{\rm I} \rangle\!\rangle_G + 2\langle\!\langle S_{\rm I} \rangle\!\rangle_G^3 \qquad\qquad (A.65)\\
=&m\left(g_m - \frac{1}{G}\right)^3 G^3 - \frac{9m(m+2)}{4}\left(g_m - \frac{1}{G}\right)^2 v_m G^4 \\
&+ \frac{3m(m+2)(m+3)}{2}\left(g_m - \frac{1}{G}\right) v_m^2 G^5 - \frac{m(m+2)(5m^2+34m+60)}{16} v_m^3 G^6 + O\left(\frac{1}{n^4}\right) \\
=&-\frac{m(m+2)(m+8)}{8}\left(v_m G^2\right)^3 + O\left(\frac{1}{n^4}\right) \\
=&-\frac{m(m+2)(m+8)}{8}\left(\frac{V}{nG^2}\right)^3 + O\left(\frac{1}{n^4}\right) .
\end{aligned}$$

Putting (A.64) and (A.65) back into our variational expression for the entropy (A.45), we finally arrive at

$$\begin{aligned}
&\mathcal{S}[p(z_1, \ldots, z_m|x)] \qquad\qquad (A.66)\\
=&\frac{m}{2}\log(2\pi e G) - \frac{(m^2+2m)}{16}\left(\frac{V}{nG^2}\right)^2 + \frac{(m^3+10m^2+16m)}{48}\left(\frac{V}{nG^2}\right)^3 + O\left(\frac{1}{n^4}\right) .
\end{aligned}$$

[24] In principle, at this order we would need to continue and find an expression for u_m in terms of an additional six-point vertex constraint, but as we will soon see, u_m doesn't factor into any of our expressions for the mutual information. This is the reason why we are able to analyze these next-to-leading-order corrections without otherwise evaluating additional MLP recursions.

This result in turn lets us compute the NLO correction to the mutual information between two sets of neurons, $\mathcal{M}_1 = \{1, \ldots, m_1\}$ and $\mathcal{M}_2 = \{m_1 + 1, \ldots, m_1 + m_2\}$:

$$\begin{aligned} &\mathcal{I}\left[p(\mathcal{M}_1, \mathcal{M}_2|x)\right] \\ \equiv &\mathcal{S}[p(\mathcal{M}_1|x)] + \mathcal{S}[p(\mathcal{M}_2|x)] - \mathcal{S}[p(\mathcal{M}_1, \mathcal{M}_2|x)] \\ = &\frac{m_1 m_2}{8}\left[\frac{V^{(\ell)}}{n_{\ell-1}\left(G^{(\ell)}\right)^2}\right]^2 - \frac{m_1 m_2(20 + 3m_1 + 3m_2)}{48}\left[\frac{V^{(\ell)}}{n_{\ell-1}\left(G^{(\ell)}\right)^2}\right]^3 + O\Big(\frac{1}{n^4}\Big)\,, \end{aligned} \tag{A.67}$$

where once again we have restored layer indices on the metric, the four-point vertex, and the layer width. Note that as promised, the sextic coupling u_m dropped out in the end.[25]

Excitingly, these two terms have opposite signs. To explain our excitement, let us plug in our scaling solution for the normalized four-point vertex (5.128) evaluated at the final layer $\ell = L$:

$$\frac{V^{(L)}}{n_{L-1}\left(G^{(L)}\right)^2} \equiv \nu r\,. \tag{A.68}$$

Here, $r \equiv L/n$ is the overall depth-to-width aspect ratio of the network, and ν is an activation-function dependent constant: for the $K^\star = 0$ universality, we have (5.131)

$$\nu = \frac{2}{3}\,, \tag{A.69}$$

independent of the details of the activation function itself; for scale-invariant activation functions, we have (5.130)

$$\nu = \left(\frac{3A_4}{A_2^2} - 1\right)\,, \tag{A.70}$$

with $A_2 \equiv (a_+^2 + a_-^2)/2$ and $A_4 \equiv (a_+^4 + a_-^4)/2$.[26] Plugging this scaling solution (A.68) into our NLO expression for the mutual informtion (A.67), we get

$$\mathcal{I}\left[p(\mathcal{M}_1, \mathcal{M}_2|x)\right] = \frac{m_1 m_2}{8}\nu^2 r^2 - \frac{m_1 m_2(20 + 3m_1 + 3m_2)}{48}\nu^3 r^3 + O\Big(r^4\Big)\,. \tag{A.71}$$

Now, let us explain our excitement. At one extreme, in the infinite-width limit $r \to 0$, this mutual information vanishes, as we already knew from (A.27). Thus, for small aspect ratios $r \ll 1$, the first leading-order term dominates and the mutual information increases as depth increases. As the depth continues to increase, then the second term begins to dominate, *decreasing* the mutual information. But we know by the *subadditivity* of the

[25] This is another realization of the principle of maximum entropy for nearly-Gaussian distributions. In particular, the entropy for a zero-mean distribution that satisfies constraints fixing its two-point correlator and its connected four-point correlator – and otherwise leaves unfixed its connected six-point correlator – is given to order $1/n^3$ by our expression (A.66). Thus, if we did add a third constraint that fixed the connected six-point correlator, with $U = O\big(1/n^2\big)$, then any terms of the form $O(v_m u_m) = O\big(UV/n^3\big)$ cannot appear in (A.66), as they could *increase* the entropy depending on the sign of UV.

[26] This gives $\nu = 2$ for `linear` activations and $\nu = 5$ for the `ReLU`.

entropy, (A.15), that the mutual information is always bounded from below by zero, and so for large enough aspect ratios r, this decreasing will be balanced by additional higher-order corrections. Taken altogether, we expect that at some nonzero but not too large r, the mutual information will reach a local maximum. Indeed, maximizing (A.71), we find an **optimal aspect ratio** for the network:

$$r^\star = \left(\frac{4}{20 + 3m_1 + 3m_2}\right)\frac{1}{\nu}\,, \tag{A.72}$$

with ν containing all of the details of the activation function.[27]

Although it's not immediately obvious, maximizing a mutual information such as (A.68) is closely related to well-known **unsupervised learning** objectives.[28] In contrast to a *supervised* learning setting where the goal is to predict the true output $y \equiv f(x)$ for any input sample x, in an *unsupervised* learning setting the goal is to learn representations for a collection of input samples by observing patterns in the data. For human-generated datasets, this has the advantage of eliminating the tedious task of labeling all the samples. It should also be no surprise, given the benefit of representation learning (§11), that models (pre)trained with unsupervised learning algorithms can often be efficiently fine-tuned on subsequent supervised learning tasks.

In the current context, rather than doing any actual learning, we are understanding, a priori, which choice of the architecture and activation function can lead to a larger mutual information between neurons in deeper layers. In particular, comparing the values of ν in (A.69) and (A.70) for different choices of activation function, we see in principle that networks built from `tanh` activation function should be deeper than networks built from `ReLU` activation functions to have the same mutual information in a layer.

With this objective in mind, we should continue maximizing (A.71) across all the possible partitions (m_1, m_2). This picks out a very natural choice of a partition spanning the whole layer and of equal sizes: $m_1 = m_2 = n_L/2$. With this further maximization, we get

$$r^\star = \left(\frac{4}{20 + 3n_L}\right)\frac{1}{\nu}\,, \tag{A.73}$$

for the optimal aspect ratio and

$$\mathcal{I}\left[p(\mathcal{M}_1, \mathcal{M}_2|x)\right] = \frac{1}{6}\left(\frac{n_L}{20 + 3n_L}\right)^2\,, \tag{A.74}$$

[27] Don't worry about the higher-order contributions $O\big(r^4\big)$ that we neglected in obtaining the solution (A.72): for a particular choice of activation function, i.e. for a choice of ν, the estimate of the optimal value (A.72) is *a posteriori* justified so long as the product $\nu r^\star$ is perturbatively small for a particular ν and a particular grouping of neurons (m_1, m_2). FYI, this argument is analogous the one given to justify the two-loop *Banks-Zaks fixed point* of the renormalization group in non-Abelian gauge theory [82].

[28] In particular, the *InfoMax* principle [83] recommends maximizing the mutual information between the input x and a representation $z(x)$. A related notion involves maximizing the mutual information between different representations, $z_1(x)$ and $z_2(x)$, for the same input x [84]. This latter notion can be shown to lower bound the InfoMax objective and thus motivates our analysis here.

for the associated value of the maximized mutual information. In particular, this corresponds to a lower bound on the *InfoMax* objective that we discussed in footnote 28.[29]

As a final comment on (A.73), we can think of this optimal aspect ratio as defining a scale that separates the effectively-deep regime for which our effective theory is valid from the overly-deep regime where the theory is strongly-coupled and networks are no longer trainable. We will push this interpretation further in the next appendix when we discuss residual networks.

For a final computation, let's look at the *tripartite information* (A.24) for mutually-exclusive subsystems $\mathcal{M}_1$, $\mathcal{M}_2$, and $\mathcal{M}_3$ of sizes (m_1, m_2, m_3) neurons, respectively. Plugging in (A.66) for various different combinations of the sizes, we find

$$\mathcal{I}_3\left[p(\mathcal{M}_1,\mathcal{M}_2,\mathcal{M}_3|x)\right] = \frac{m_1 m_2 m_3}{8}\left[\frac{V^{(\ell)}}{n_{\ell-1}\left(G^{(\ell)}\right)^2}\right]^3 + O\left(\frac{1}{n^4}\right) , \tag{A.75}$$

which, as per (5.128), scales cubically with depth, $\mathcal{I}_3\left[p(\mathcal{M}_1,\mathcal{M}_2,\mathcal{M}_3|x)\right] \propto \ell^3/n_{\ell-1}^3$, and thus indicates that a nonzero result only first appears at $O(1/n^3)$.[30] Importantly, we see that the tripartite information is exclusively *positive*, meaning that any three groups of neurons in a layer will form *redundant* representations under RG flow: knowing the activities of any one group of neurons $\mathcal{M}_1$ means you would learn less information about a second group of neurons $\mathcal{M}_2$ from observing a third group $\mathcal{M}_3$ than you otherwise would have learned had you not already known $\mathcal{M}_1$. It would be interesting to try to understand further how this property relates to the coarse-graining mechanism of the representation group flow to the deeper layers.[31]

[29] While this value (A.74) may seem small, remember that our analysis is based entirely on the prior distribution before any learning has taken place.

[30] In hindsight, it's obvious that we needed the entropy to have at least a cubic dependence on the sizes m_i to find a nonzero answer for the tripartite information, just as we needed the entropy to have at least a quadratic dependence on the sizes to find a nonzero answer for the mutual information (A.53).

[31] It would also be interesting to try and interpret this in terms of the *optimal brain damage* of [80] or the *lottery ticket hypothesis* of [81].

Appendix B

Residual Learning

No, Doc. Not me, the other me, the one that's up on stage playing Johnny B. Goode.

Marty McFly [85].

In this final appendix, we'll analyze neural networks with residual connections, generally called **residual networks**. These networks were originally introduced in order to enable the training of deeper and deeper networks: traditionally deep networks suffer from the *exploding and vanishing gradient problem*, but even in networks where various tricks of the trade are used to ensure the propagation of forward and backward signals, overly-deep networks are empirically found to have *higher training and test errors* than their shallower-network counterparts.

From the microscopic perspective, the increase of the generalization error is intuitive for a deeper model with more model parameters, but the increase in the training error is not: the additional parameters should naively enable *better* fits to the training data. At the very least, one might hope that the additional layers could – in principle – approximate the identity map and do *no worse*. Yet the empirical evidence mentioned above suggests that it's difficult for optimization algorithms to tune the hidden layers of a deep network to such a nearly-identity map. This is called **degradation**, and in principle is a major limiting factor for developing larger scale deep-learning models.

From the macroscopic perspective of our effective theory, we can offer a *dual* explanation for this degradation problem. As a precursor to our explanation, first recall that, rather than using any heuristic approach to solve the exploding and vanishing gradient problem, in §9.4 we analytically solved its *exponential* manifestation by means of criticality and then solved its *polynomial* manifestation by means of the learning-rate equivalence principle. In §10.3 we further confirmed that the associated tunings of the initialization hyperparameters, $C_b^{(\ell)}$ and $C_W^{(\ell)}$, and of the training hyperparameters, $\lambda_b^{(\ell)}$ and $\lambda_W^{(\ell)}$, lead to the most robust generalization performance for MLPs.

Now, *even if* these hyperparameters are properly tuned, we would expect that overly-deep networks will suffer horribly from instantiation-to-instantiation fluctuations, lead-

ing to the breakdown of criticality for any *particular* network. This problem was first discussed in §3.4 for extremely deep (linear) networks, then more generally in §5.4, and finally in footnote 8 of Epilogue ε. Thus, combined with our discussion of the MLP's depth-to-width ratio $r \equiv L/n$ in §A.3, perhaps we can understand the training loss degradation problem in terms of the network leaving the regime of optimality and proceeding towards the regime of chaos.

If you'll please move the microscopic explanation back to the front of your mind, we can explain an ingenious solution to degradation by He *et al.* [86]. Rather than trying to find a better learning algorithm, we can instead modify the deep network architecture so that the hidden layers only have to learn a *residual function*: in place of a generic nonlinear $(\ell+1)$-th layer

$$z^{(\ell+1)} = \mathtt{L}\Big(z^{(\ell)}; \theta^{(\ell+1)}\Big) \,, \tag{B.1}$$

we design the layer as

$$z^{(\ell+1)} = \mathtt{R}\Big(z^{(\ell)}; \theta^{(\ell+1)}\Big) + z^{(\ell)} \,, \tag{B.2}$$

such that the **residual block**, $\mathtt{R}\Big(z^{(\ell)}; \theta^{(\ell+1)}\Big)$, is the residual of the function that we want our layer, (B.1), to implement. The basic structure of this generic residual layer is depicted in the left panel of Figure B.1 and will be further explained later on.

From a microscopic perspective, these residual connections make learning a nearly-identity map much easier. Indeed, it is far easier to set the residual block $\mathtt{R}\Big(z^{(\ell)}; \theta^{(\ell+1)}\Big)$ to near-zero than it is to coax a generic nonlinear function $\mathtt{L}\Big(z^{(\ell)}; \theta^{(\ell+1)}\Big)$ to approximate the identity. In particular, since standard building blocks of the neural network often have the property that they vanish when their parameters are set to zero – cf. (2.5) for the *MLP-layer block* – and since typical initialization distributions have zero mean – cf. (2.21) and (2.22) – residual networks make it fairly easy to find a solution with $\mathtt{R}\Big(z^{(\ell)}; \theta^{\star}\Big) \approx 0$ such that the addition of the hidden layer doesn't necessarily degrade the performance of the network.

More generally, we hope that the preactivation will actually play two roles, one of *coarse-graining* the input signal according to representation group flow (§4.6) and the other of propagating an *undegraded* copy of the input signal. This is plausibly quite helpful as it allows us to train deeper models with more parameters, and it has indeed been empirically demonstrated that such deeper residual networks lead to significant performance gains on the test set.

One of the goals of this appendix is to provide a dual macroscopic explanation for why the residual connections let us train overly-deep networks. Our macroscopic explanation above for the origin of the degradation problem – combined with the empirical success of very deep residual networks – suggests that the inclusion of residual connections shifts the optimal aspect ratio $r^{\star}$ from its MLP value, (A.72), to higher values. This would extend the range of effectively-deep networks and thus explain why residual connections let us train deeper networks. To test this hypothesis, we will need to carry out our effective theory analysis for residual networks.

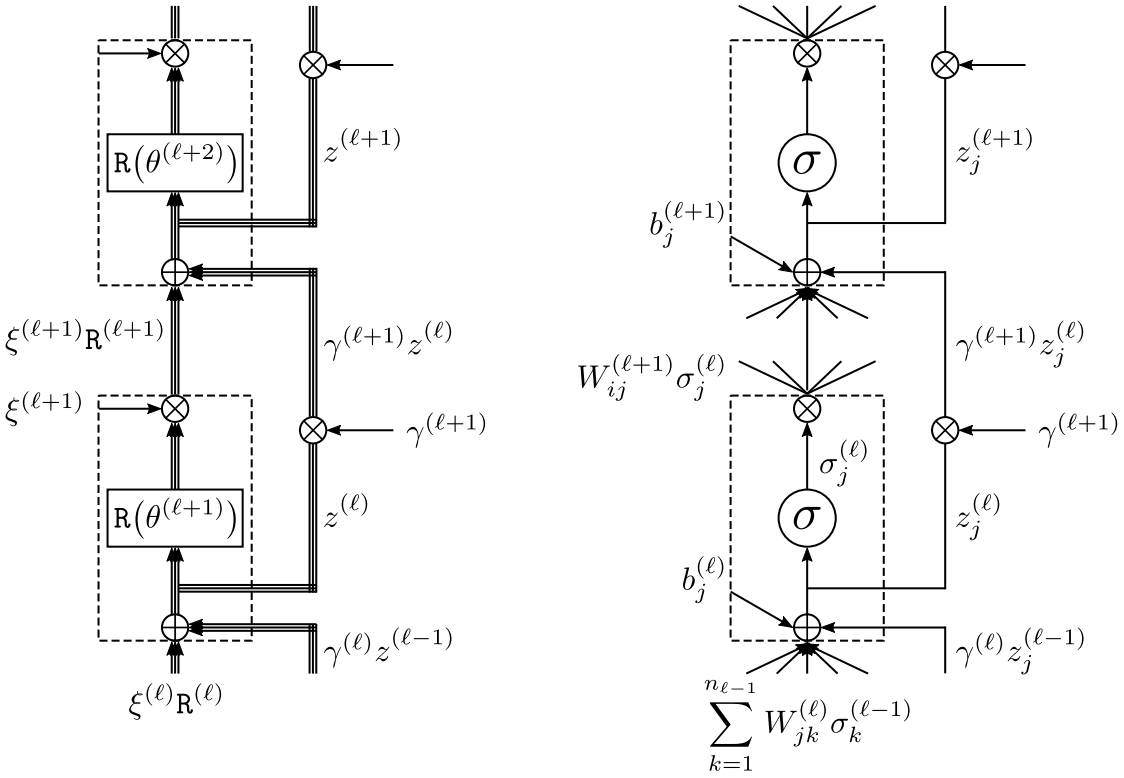

Figure B.1: **Left:** two residual blocks from adjacent layers for a very general *residual network*. This detailed structure depicts how each layer *(i)* adds a weighted block and the weighted preactivation to produce the next layer's preactivation, *(ii)* copies the preactivation to skip to the next layer, and *(iii)* generates the next layer's block. **Right:** two neurons from adjacent layers for a *residual MLP*. This detailed structure depicts how each layer *(i)* adds the bias, the weighted activation, and the weighted preactivation to produce the next layer's preactivation, *(ii)* copies the preactivation to skip to the next layer, *(iii)* generates the activation from the preactivation, and *(iv)* multiplies the activation by the next-layer weight.

Despite the long prelude, this appendix will be relatively brief and only involve some simple calculations. These exercises will serve two purposes. First, given the overwhelming ubiquity of **residual connections** – sometimes called *skip connections* or *shortcuts* – in modern deep learning architectures, it is practically useful to explain how the critical initialization hyperparameters *shift* when residual connections are included. Second, this will showcase how our effective theory formalism can easily be applied to neural network architectures other than vanilla MLPs.

To those ends – and to the end of the book – in §B.1 we'll begin by briefly introducing the perhaps simplest residual network, a *multilayer perceptron with residual connections*. In §B.2, we'll study the infinite-width limit of this model, performing a criticality analysis in order to understand how the *residual hyperparameters* interplay with the critical initialization hyperparameters.

Then, in §B.3 we'll study the residual MLP at finite width. Using our auxiliary unsupervised learning objective from the last appendix, we'll see how the inclusion of residual connections can shift the *optimal aspect ratio* of a network $r^\star$ to large values. We will also learn that for networks with aspect ratios below a certain threshold, residual connections are *always* harmful according to this criterion. Altogether, this provides a new effective-theory macroscopic perspective on how residual connections solve the degradation problem described above, and further lets us understand the tradeoff of propagating signals through the residual block – in this case, a nonlinear MLP-layer block – against propagation through identity block – skipping signals to deeper layers.

Finally, in §B.4 we'll give a hybrid theoretical-empirical recipe applying the analyses in the previous two sections to *general* residual networks with arbitrarily complicated residual blocks $\mathtt{R}\Big(z^{(\ell)};\theta^{(\ell+1)}\Big)$. We hope this may have broad application to the many deep learning architectures that implement residual connections.[1] After this, you will have finished all your residual learning from this book.

B.1 Residual Multilayer Perceptrons

A **multilayer perceptron with residual connections** can be defined by the forward equation,

$$z_{i;\delta}^{(\ell+1)} = \xi^{(\ell+1)}\left[b_i^{(\ell+1)} + \sum_{j=1}^{n_\ell} W_{ij}^{(\ell+1)}\sigma_{j;\delta}^{(\ell)}\right] + \gamma^{(\ell+1)} z_{i;\delta}^{(\ell)} \,. \tag{B.3}$$

Compared with our schematic description, (B.2), here we've picked just a standard MLP layer as our residual block, $\mathtt{R}\Big(z^{(\ell)};\theta^{(\ell+1)}\Big)$, and we've allowed for *scalar* **residual**

[1] Residual networks were first described by He *et al.* [86]. Typically, when referring to a residual network as a *ResNet*, the base block is composed of convolutional layers, cf. (2.8), that are further augmented with very popular heuristic for mitigating the exploding and vanishing gradient problem [87]. While original domain of ResNets was computer vision tasks, they have now been applied to other domains; more broadly, *residual connections* are components in a wide variety of modern deep learning architectures, including importantly the transformer-based language models that have been revolutionizing natural language processing [14].

hyperparameters, $\xi^{(\ell)}$ and $\gamma^{(\ell)}$, that control the relative magnitudes of the residual-block term vs. the identity term. Here we need to take

$$n_\ell = n_{\ell+1} \equiv n \tag{B.4}$$

so that we can add the weighted and biased activation back to the preactivation before the activation, and such residual connections are thus only included for the network's hidden layers.[2] A graph of two neurons in adjacent layers from a residual MLP is shown in the right panel of Figure B.1, and if you need to recall the overall global structure of an MLP, please refer back to Figure 2.1.

Although nice for the symmetry, one of the residual hyperparameters is redundant: you can show that adjusting $\xi^{(\ell)}$ has the same effect on the ensemble of residual MLPs as rescaling the initialization hyperparameters, $C_b^{(\ell)}$ and $C_W^{(\ell)}$, and the training hyperparameters, $\lambda_b^{(\ell)}$ and $\lambda_W^{(\ell)}$. As such, we'll henceforth set

$$\xi^{(\ell)} = 1\,, \tag{B.6}$$

without loss of generality, leaving us only one (potentially layer-dependent) residual hyperparameter: $\gamma^{(\ell)}$. Note importantly that in the limit $\gamma^{(\ell)} \to 0$ we recover a vanilla MLP without any residual connections. A customary choice for this hyperparameter is $\gamma^{(\ell)} = 1$, cf. (B.2), but let us now show that this is suboptimal because it *breaks* criticality.[3]

B.2 Residual Infinite Width: Criticality Analysis

To understand how to set the residual hyperparameter $\gamma^{(\ell)}$ and preserve criticality for the residual MLP, let's work out a recursion for the two-point correlator:

$$\mathbb{E}\left[z^{(\ell)}_{i_1;\delta_1} z^{(\ell)}_{i_2;\delta_2}\right] = \delta_{i_1 i_2} G^{(\ell)}_{\delta_1\delta_2}\,. \tag{B.7}$$

Using the forward equation (B.3), we find

$$G^{(\ell+1)}_{\delta_1\delta_2} = C_b + C_W \frac{1}{n}\sum_{j=1}^{n} \mathbb{E}\left[\sigma^{(\ell)}_{j;\delta_1}\sigma^{(\ell)}_{j;\delta_2}\right] + \gamma^2 G^{(\ell)}_{\delta_1\delta_2}\,. \tag{B.8}$$

[2] More generally, if we wanted $n_{\ell+1} \neq n_\ell$ we could instead use an iteration equation such as

$$z^{(\ell+1)}_{i;\delta} = \sum_{j=1}^{n_{\ell+1}} \xi^{(\ell+1)}_{ij}\left[b^{(\ell+1)}_j + \sum_{k=1}^{n_\ell} W^{(\ell+1)}_{jk}\sigma^{(\ell)}_{k;\delta}\right] + \sum_{j=1}^{n_\ell} \gamma^{(\ell+1)}_{ij} z^{(\ell)}_{j;\delta}\,, \tag{B.5}$$

where the residual hyperparameters are now matrices: $\xi^{(\ell+1)}_{ij}$ is an $n_{\ell+1}$-by-$n_{\ell+1}$ matrix and $\gamma^{(\ell+1)}_{ij}$ is an $n_{\ell+1}$-by-n_ℓ matrix, and in general both matrix could vary from layer to layer.

[3] In particular, this choice is problematic without any additional heuristics, e.g. [87], for otherwise mitigating the exploding and vanishing gradient problem. The following analysis thus offers an explanation for why deep $\gamma^{(\ell)} = 1$ residual networks without such heuristics will always perform poorly.

Here, mirroring all our previous discussions of criticality that are now too many to enumerate with references, throughout this section we'll set the bias variance, $C_b^{(\ell)}$, the rescaled weight variance, $C_W^{(\ell)}$, and the residual hyperparameter, $\gamma^{(\ell)}$, to be uniform across layers,

$$C_b^{(\ell)} = C_b \,, \qquad C_W^{(\ell)} = C_W \,, \qquad \gamma^{(\ell)} = \gamma \,. \tag{B.9}$$

Compared to the metric recursion for a vanilla MLP (4.72),

$$G^{(\ell+1)}_{\delta_1\delta_2} = C_b + C_W \frac{1}{n}\sum_{j=1}^{n} \mathbb{E}\left[\sigma^{(\ell)}_{j;\delta_1}\sigma^{(\ell)}_{j;\delta_2}\right] \,, \tag{B.10}$$

the $(\ell+1)$-th-layer metric for the residual MLP is shifted by the (rescaled) identity term from the forward equation, leading to an additional contribution of the ℓ-th-layer metric as the (rescaled) final term of (B.8).

In particular, the infinite-width limit of the residual MLP leads to the following recursion for its kernel,

$$K^{(\ell+1)}_{\delta_1\delta_2} = C_b + C_W \left\langle \sigma_{\delta_1}\sigma_{\delta_2} \right\rangle_{K^{(\ell)}} + \gamma^2 K^{(\ell)}_{\delta_1\delta_2} \,. \tag{B.11}$$

Then, writing the kernel in our $\gamma^{[a]}$ basis (5.15), using our δ expansion, (5.22)–(5.24), we can follow the two-input perturbative analysis of §5.1 to derive component recursions

$$K^{(\ell+1)}_{00} = C_b + C_W\, g\Big(K^{(\ell)}_{00}\Big) + \gamma^2 K^{(\ell)}_{00} \,, \tag{B.12}$$

$$\delta K^{(\ell+1)}_{[1]} = \chi_{\parallel}\Big(K^{(\ell)}_{00}\Big)\, \delta K^{(\ell)}_{[1]} \,, \tag{B.13}$$

$$\delta\delta K^{(\ell+1)}_{[2]} = \chi_{\perp}\Big(K^{(\ell)}_{00}\Big)\, \delta\delta K^{(\ell)}_{[2]} + h\Big(K^{(\ell)}_{00}\Big)\Big(\delta K^{(\ell)}_{[1]}\Big)^2 \,, \tag{B.14}$$

which are modified from their vanilla expressions (5.46)–(5.48).[4] Here, the *parallel susceptibility*,

$$\chi_{\parallel}(K) \equiv \gamma^2 + \frac{C_W}{2K^2}\left\langle \sigma(z)\,\sigma(z)\left(z^2 - K\right)\right\rangle_K \,, \tag{B.15}$$

and the *perpendicular susceptibility*,

$$\chi_{\perp}(K) \equiv \gamma^2 + C_W \left\langle \sigma'(z)\,\sigma'(z)\right\rangle_K \,, \tag{B.16}$$

are each shifted from their previous values, (5.50) and (5.51), by a constant γ^2 term, while the helper functions, $g(K)$ and $h(K)$, (5.49) and (5.52), are the same as before:

$$g(K) \equiv \left\langle \sigma(z)\,\sigma(z)\right\rangle_K \,, \tag{B.17}$$

$$h(K) \equiv \frac{1}{2}\frac{d}{dK}\chi_{\perp}(K) \,. \tag{B.18}$$

[4]Note that, as per this first recursion (B.12), a perturbation to a single-input, $K^{(\ell)}_{00} = K^{\star}_{00} + \Delta K^{(\ell)}_{00}$, is governed by the *shifted* parallel susceptibility (B.15), cf. (5.9).

As should be already clear from this simple analysis, inclusion of residual connections ($\gamma \neq 0$) will shift the critical initialization hyperparameters. Fixing the criticality conditions

$$\chi_{\parallel}(K^\star) = 1\,, \qquad \chi_{\perp}(K^\star) = 1\,, \tag{B.19}$$

we can find the new critical initializations for our universality classes: specifically, for the scale-invariant universality class, the critical initialization hyperparameters, (5.67), are modified as

$$C_b = 0\,, \qquad C_W = C_W(\gamma) \equiv \frac{1}{A_2}\left(1 - \gamma^2\right)\,, \tag{B.20}$$

where $A_2 \equiv (a_+^2 + a_-^2)/2$, cf. (5.59); analogously, for the the $K^\star = 0$ universality class, the critical initialization hyperparameters (5.90) are modified as

$$C_b = 0\,, \qquad C_W = C_W(\gamma) \equiv \frac{1}{\sigma_1^2}\left(1 - \gamma^2\right)\,, \tag{B.21}$$

where $\sigma_1 \equiv \sigma'(0)$.

In §2.3, we discussed that the *zero initialization*, $b_i^{(\ell)} = W_{ij}^{(\ell)} = 0$, fails to break the permutation symmetry among the n neurons in a hidden layer. In conjunction with this reasoning, we now see that the criticality conditions for either class, (B.20) and (B.21), are unsatisfiable for the customary choice $\gamma = 1$ of the residual hyperparameter. More broadly, for each universality class there is a *one-parameter family* of critical rescaled weight variances: for each $0 < \gamma^2 < 1$, there is an associated critical value of $C_W = C_W(\gamma)$. Thus, in order to directly mitigate the exploding and vanishing gradient problem, for a particular choice of $0 < \gamma^2 < 1$ and activation function, we must pick C_W according to the appropriate $C_W(\gamma)$.

B.3 Residual Finite Width: Optimal Aspect Ratio

From our infinite-width analysis, we just saw that residual networks have a one-parameter family of critical solutions: $C_W(\gamma)$. As per the kernel recursion (B.11), these solutions trade off the degree to which the identity branch vs. the MLP-layer branch contributes to the next layer's preactivations. In the strict infinite-width limit, criticality ensures that the kernel is preserved for any choice $C_W(\gamma)$ in the range $0 < \gamma^2 < 1$, and all fluctuations are suppressed; thus, we're in principle completely indifferent between any of these critical tunings.

However, as should now be a familiar point of discussion, networks in the infinite-width limit effectively have a depth-to-width ratio $r \equiv L/n \to 0$. This means that infinite-width analysis cannot really get at the question of **degradation**: why do extremely deep networks have *higher training errors* than otherwise equivalent shallower networks. To compare wide networks of different depths, we need to consider finite-width networks with different aspect ratios $r > 0$.[5]

[5]In particular, at finite width the *representation group flow* through the MLP-layer blocks leads to two competing finite-width effects: *(i)* the relevant and thus *growing* dNTK and ddNTKs lead to non-

Our main tool of analysis in this section will be a computation of the *mutual information at initialization* between non-overlapping representations in deep layers of a residual MLP. As per our auxiliary unsupervised learning criterion from the last appendix, cf. footnote 28 of §A.3, this mutual information gives a natural way to estimate the optimal aspect ratio $r^\star$ of a finite-width network. Given that residual networks without an exploding and vanishing gradient problem solve the degradation problem, a natural theoretical prediction for our residual MLPs is that the inclusion of residual connections, $\gamma > 0$, and then tuning to criticality as $C_W(\gamma)$ will together *shift their optimal aspect ratios to larger values.*

To evaluate the optimal aspect ratio via our formula (A.73), we just need to compute the coefficient ν of normalized, output-layer, single-input four-point vertex for the residual MLP:

$$\frac{V^{(L)}}{n\left(G^{(L)}\right)^2} \equiv \nu r \,. \tag{B.22}$$

Note that we've already derived the recursion for the leading-order single-input metric, i.e. the kernel $K^{(L)} \equiv G^{\{0\}(L)}$, in (B.12). To compute the four-point vertex, first recall that the multi-input four-point vertex defines the four-point connected correlator (4.77) as

$$\begin{aligned} &\mathbb{E}\left[z^{(\ell)}_{i_1;\delta_1} z^{(\ell)}_{i_2;\delta_2} z^{(\ell)}_{i_3;\delta_3} z^{(\ell)}_{i_4;\delta_4}\right]\Big|_{\text{connected}} \\ =&\frac{1}{n}\left[\delta_{i_1 i_2}\delta_{i_3 i_4} V^{(\ell)}_{(\delta_1\delta_2)(\delta_3\delta_4)} + \delta_{i_1 i_3}\delta_{i_2 i_4} V^{(\ell)}_{(\delta_1\delta_3)(\delta_2\delta_4)} + \delta_{i_1 i_4}\delta_{i_2 i_3} V^{(\ell)}_{(\delta_1\delta_4)(\delta_2\delta_3)}\right] . \end{aligned} \tag{B.23}$$

Using the forward equation (B.3) and following our analysis from §4.3, we see that the recursion for the residual-MLP's four-point vertex is shifted by two additional terms proportional to γ^2 and γ^4 as compared to the leading-order recursion for the vanilla

trivial representation learning during training, and *(ii)* the relevant and *growing* four-point vertex, NTK variance, and NTK-preactivation cross correlation lead to fluctuations in the ensemble from instantiation to instantiation. As the residual connections are supposed to mitigate this second harmful effect by xeroxing the undegraded input via the identity branch, we naturally expect that they will have a meaningful physical effect on this competition. In other words, we expect that residual networks of different residual hyperparameters γ and different aspect ratios r will lead to very different test-set generalization.

MLP (4.90),

$$
\begin{aligned}
&V^{(\ell+1)}_{(\delta_1\delta_2)(\delta_3\delta_4)} \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(B.24)}\\
=&C_W^2 \left[\langle\sigma_{\delta_1}\sigma_{\delta_2}\sigma_{\delta_3}\sigma_{\delta_4}\rangle_{K^{(\ell)}} - \langle\sigma_{\delta_1}\sigma_{\delta_2}\rangle_{K^{(\ell)}} \langle\sigma_{\delta_3}\sigma_{\delta_4}\rangle_{K^{(\ell)}}\right]\\
&+ \frac{C_W^2}{4} \sum_{\delta_5,\ldots,\delta_8\in\mathcal{D}} V_{(\ell)}^{(\delta_5\delta_6)(\delta_7\delta_8)} \langle\sigma_{\delta_1}\sigma_{\delta_2}\left(z_{\delta_5}z_{\delta_6}-K_{\delta_5\delta_6}\right)\rangle_{K^{(\ell)}} \langle\sigma_{\delta_3}\sigma_{\delta_4}\left(z_{\delta_7}z_{\delta_8}-K_{\delta_7\delta_8}\right)\rangle_{K^{(\ell)}}\\
&+ C_W\gamma^2\Big[\langle\sigma_{\delta_1}\sigma_{\delta_2}\left(z_{\delta_3}z_{\delta_4}-K_{\delta_3\delta_4}\right)\rangle_{K^{(\ell)}} + \langle\sigma_{\delta_3}\sigma_{\delta_4}\left(z_{\delta_1}z_{\delta_2}-K_{\delta_1\delta_2}\right)\rangle_{K^{(\ell)}}\\
&\qquad + \frac{1}{2}\sum_{\delta_5,\ldots,\delta_8\in\mathcal{D}} V_{(\ell)}^{(\delta_5\delta_6)(\delta_7\delta_8)} \langle\sigma_{\delta_1}\sigma_{\delta_2}\left(z_{\delta_5}z_{\delta_6}-K_{\delta_5\delta_6}\right)\rangle_{K^{(\ell)}} K^{(\ell)}_{\delta_7\delta_3}K^{(\ell)}_{\delta_8\delta_4}\\
&\qquad + \frac{1}{2}\sum_{\delta_5,\ldots,\delta_8\in\mathcal{D}} V_{(\ell)}^{(\delta_5\delta_6)(\delta_7\delta_8)} \langle\sigma_{\delta_3}\sigma_{\delta_4}\left(z_{\delta_5}z_{\delta_6}-K_{\delta_5\delta_6}\right)\rangle_{K^{(\ell)}} K^{(\ell)}_{\delta_7\delta_1}K^{(\ell)}_{\delta_8\delta_2}\Big]\\
&+ \gamma^4\, V^{(\ell)}_{(\delta_1\delta_2)(\delta_3\delta_4)}\,, \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(B.25)}
\end{aligned}
$$

where in parsing this expression, please remember that the raised indices of the four-point vertex are our shorthand for contraction with the ℓ-th-layer inverse kernel, cf. (4.83). Here, in working out the middle terms proportional to γ^2, you might find the *intra*layer formula (4.64) to be useful.

Specializing now to a single input, we get

$$
V^{(\ell+1)} =C_W^2\left[\left\langle\sigma^4(z)\right\rangle_{K^{(\ell)}} - \left\langle\sigma^2(z)\right\rangle_{K^{(\ell)}}^2\right] + \left(\chi_{\parallel}^{(\ell)}\right)^2 V^{(\ell)} + 4\gamma^2\left(\chi_{\parallel}^{(\ell)}-\gamma^2\right)\left(K^{(\ell)}\right)^2\,, \tag{B.26}
$$

where for convenience we have abbreviated $\chi_{\parallel}^{(\ell)} \equiv \chi_{\parallel}\left(K^{(\ell)}\right)$ as we often do. Here, this parallel susceptibility is the shifted one appropriate for residual MLPs as defined in (B.15) with the γ^2 term; otherwise, we notice the addition of the γ-dependent final term compared with the single-input recursion for vanilla MLPs (5.109). At criticality with $\chi_{\parallel}^{(\ell)}=1$, this final term will make a nontrivial difference on the deep asymptotic behavior of the four-point vertex. Now, let's solve this recursion at criticality for our two universality classes, with the usual initial condition $V^{(1)}=0$ (cf. the title of §4.1).

Scale-Invariant Universality Class

For the scale-invariant universality class, we tune the rescaled weight variance to criticality with (B.20) so that $\chi_{\parallel}^{(\ell)}=1$. Then, the single-input kernel is exactly fixed, $K^{(\ell)}=K^{\star}$, and we also need to remember (5.113),

$$
\left[\left\langle\sigma^4\right\rangle_{K^\star} - \left\langle\sigma^2\right\rangle_{K^\star}^2\right] = \left(3A_4 - A_2^2\right)\left(K^\star\right)^2\,, \tag{B.27}
$$

with $A_2 \equiv (a_+^2 + a_-^2)/2$ and $A_4 \equiv (a_+^4 + a_-^4)/2$. Substituting all of this into (B.26), the recursion becomes

$$
V^{(\ell+1)} = V^{(\ell)} + (1-\gamma^2)\left[(1-\gamma^2)\left(\frac{3A_4}{A_2^2}-1\right) + 4\gamma^2\right]\left(K^\star\right)^2\,, \tag{B.28}
$$

which gives a simple additive solution of the form (B.22), with

$$\nu = \nu(\gamma) \equiv (1-\gamma^2)\left[(1-\gamma^2)\left(\frac{3A_4}{A_2^2}-1\right)+4\gamma^2\right] . \tag{B.29}$$

As a quick check, we see that ν reduces to our previous result for vanilla MLPs, (A.70), as $\gamma \to 0$. More importantly, $\nu(\gamma)$ is strictly positive in the allowed range $0 < \gamma < 1$ and monotonically decreases to $\nu(1) = 0$ with increasing γ.[6]

$K^\star = 0$ Universality Class

For the $K^\star = 0$ universality class, we tune the rescaled weight variance to criticality with (B.21). In this case, the single-input asymptotic behavior of the kernel is modified. Evaluating the residual-MLP kernel recursion (B.12) at criticality and recalling our expansion (5.83),

$$g(K) = \sigma_1^2\left[K + a_1 K^2 + O\Big(K^3\Big)\right] , \tag{B.30}$$

the single-input kernel recursion becomes

$$K^{(\ell+1)} = K^{(\ell)} + a_1(1-\gamma^2)\left(K^{(\ell)}\right)^2 + \ldots , \tag{B.31}$$

which has deep asymptotic behavior of

$$K^{(\ell)} = \left[\frac{1}{(-a_1)(1-\gamma^2)}\right]\frac{1}{\ell} + \ldots . \tag{B.32}$$

To evaluate the four-point vertex recursion, we need

$$\chi_\parallel(K) = 1 + (1-\gamma^2)\left[2a_1 K + O\Big(K^2\Big)\right] , \tag{B.33}$$

$$\left[\Big\langle \sigma^4 \Big\rangle_{K^\star} - \Big\langle \sigma^2 \Big\rangle_{K^\star}^2\right] = \sigma_1^4\left[2K^2 + O\Big(K^3\Big)\right] , \tag{B.34}$$

where the first expression is shifted from (5.121), as per (B.15) and (B.21), but the second expression is the same as before (5.122). Plugging in these expressions and matching terms, we find

$$\nu = \nu(\gamma) \equiv \frac{2}{3}(1-\gamma^4) . \tag{B.35}$$

This also reduces to our previous result for vanilla $K^\star = 0$ MLPs, (A.69), as $\gamma \to 0$, and also is strictly positive in the allowed range $0 < \gamma < 1$, monotonically decreasing to $\nu(1) = 0$ with increasing γ. Thus, $\nu(\gamma)$ for $K^\star = 0$ universality class is qualitatively comparable to the scale-invariant universality class.

[6] To see this, you'll need to use the fact that $A_4 \geq A_2^2$ for all scale-invariant activation functions.

Physics of the Optimal Aspect Ratio

As we saw at the end of §A.3, the maximization of the $1/n^3$ mutual information (A.67) according to our unsupervised learning criterion led to a natural estimate of the optimal aspect ratio of a network (A.73). For residual MLPs, this gives

$$r^\star(\gamma) \equiv \left(\frac{4}{20+3n_L}\right)\frac{1}{\nu(\gamma)}\,, \tag{B.36}$$

with $\nu(\gamma)$ given by (B.29) for the scale-invariant universality class and (B.35) for the $K^\star = 0$ universality class. Thus, we see that the monotonic decrease of $\nu(\gamma)$ for either class leads to a monotonic increase of $r^\star(\gamma)$ as γ increases: beginning from vanilla MLPs at $\gamma = 0$, residual MLPs will prefer larger and larger aspect ratios. Thus, we have realized our theoretical prediction at the beginning of the section, finding support in our effective theory formalism for the hypothesis that residual connections can solve the degradation problem.

Let us offer a further interpretation of this mechanism. In [88], it was suggested that the benefit of residual connections in extremely deep networks is that they let the global network architecture behave as an *ensemble* of many shallower networks: in this interpretation, each *particular* "shallow network" is given by following the path of a signal from the input to the output of the single residual network. What was discovered is that gradients are dominated by paths skipping over most of the network and only passing through a small fraction of the hidden-layer residual blocks. This actually gives an interesting way to look at our result (B.36): if our maximization of mutual information (A.67) is weighing the helpful effect of depth as an inductive bias for neural association (§6.4.1) against the harmful effect of depth due to growing fluctuations (§5.4), then such ensembling would be a natural way to suppress the fluctuations while preserving the neural association. Then, using residual connections to extend the depth of *trainable* networks should, at least to a point, also lead to better test set performance.

Note finally that we can actually interpret (B.36) in another way: rather than estimating the optimal aspect ratio $r^\star(\gamma)$ for a fixed residual hyperparameter γ, *instead* we can think of it as estimating the optimal residual hyperparameter $\gamma^\star(r)$ for a fixed aspect aspect ratio r. Taking this latter perspective, we learn something interesting. On the one hand, for very shallow network with $r \ll 1$, our criterion (B.36) is unsatisfiable since $\nu(\gamma)$ monotonically decreases from its maximal value at $\gamma = 0$. Such networks are shallower than optimal according to our original criterion (A.73) for vanilla MLPs, $r < r^\star(\gamma = 0)$, and their mutual information will be greatest *without* the residual connections $\gamma^\star(r) = 0$. On the other hand, for very deep networks with $r > r^\star(\gamma = 0)$, the optimal residual hyperparameter $\gamma^\star(r)$ monotonically asymptotes to 1 with growing r.[7] Altogether, this suggests that we should only turn on the residual connections for

[7] When setting $\gamma^\star$ it's important to always remember to also adjust the rescaled weight variance $C_W(\gamma^\star)$ according to (B.20) or (B.21) in order to maintain criticality. In the limit of $r \gg r^\star(\gamma = 0)$, the optimal residual hyperparameter asymptotes to one as $1 - [\gamma^\star(r)]^2 \sim 1/r$ and the critical initialization gets smaller as $C_W\big(\gamma = \gamma^\star(r)\big) \sim 1/r$.

networks with aspect ratios r that are greater than the threshold $r^\star(\gamma = 0)$ set by the optimal aspect ratio of vanilla MLPs.

Incidentally, if you are worried about the validity of perturbation theory for large r, note that the real physical expansion parameter is given by the combination

$$\frac{V^{(L)}}{n\left(G^{(L)}\right)^2} = \nu\Big(\gamma = \gamma^\star(r)\Big) \times r\,, \tag{B.37}$$

which stays finite even as the ratio r asymptotes to infinity. In this sense, we can use $\gamma^\star(r)$ to arbitrarily extend the regime of effectively-deep networks that can be described by our effective theory.

B.4 Residual Building Blocks

In this final section, we will explain a hybrid theoretical-empirical method for tuning a very general residual network to criticality. This method may be implemented practically in order to tune the hyperparameters of residual networks beyond the multilayer perceptron architecture. We hope this discussion provides a *blueprint* for how a few simple measurements on small networks can then be scaled up to efficiently design much larger models according to our effective theory approach.

A *general* residual network can be defined by replacing a simple MLP-layer block (B.3) by a generic nonlinear **residual block**:

$$b_i^{(\ell+1)} + \sum_{j=1}^{n} W_{ij}^{(\ell+1)} \sigma_{j;\delta}^{(\ell)} \;\to\; \mathtt{R}_i\Big(z_\delta^{(\ell)}; \theta^{(\ell+1)}\Big) \equiv \mathtt{R}_{i;\delta}^{(\ell+1)}\,. \tag{B.38}$$

Here, the ℓ-th-layer residual block $\mathtt{R}_{i;\delta}^{(\ell)}$ is shaped by model parameters $\theta^{(\ell)}$, and we should pick $\mathtt{R}_{i;\delta}^{(\ell)}$ to be a square matrix in its neural indices, $n_\ell = n_{\ell+1} \equiv n$, so that we can add the output of the residual block back to the preactivation. This leads to the following forward equation,

$$z_{i;\delta}^{(\ell+1)} = \xi^{(\ell+1)} \mathtt{R}_i\Big(z_\delta^{(\ell)}; \theta^{(\ell+1)}\Big) + \gamma^{(\ell+1)} z_{i;\delta}^{(\ell)}\,, \tag{B.39}$$

where we've restored the second residual hyperparameter, $\xi^{(\ell)}$, in order to let us scale the overall magnitude of the residual-block term.This iteration equation (B.39) is sufficiently generic to schematically capture many popular deep learning architectures, including the computer vision workhorse architecture, the *residual convolution network* or **ResNet**, and the multi-headed self-attention-seeking language-modeling **transformer**. A graph of two residual blocks in adjacent layers from a general residual network described by (B.39) is shown in the left panel of Figure B.1.

In practice, certain heuristics and ad hoc methods are used in these general architectures to try and mitigate the exploding and vanishing gradient problem. However, as we know from §9.4, *criticality* is a much more motivated solution. In §B.2, we saw that for residual MLPs, the residual hyperparameters can be used to control criticality for

the network. Now, let's see how we can implement a form of criticality for our general residual network (B.39).

Broadly speaking, we now need to solve two problems of different nature and difficulties:

- First, we need to ensure that signals can easily propagate through the residual block, especially for blocks that are *deep* in some sense; for instance, if a block individually consists of $L_{\mathtt{R}}$ MLP layers, i.e. *an MLP-$L_{\mathtt{R}}$-layer block*, it will then have an internal version of exploding and vanishing gradient problem. Theorists should make every effort to critically analyze such blocks of practical interest, but if we have to treat the residual block $\mathtt{R}^{(\ell)}_{i;\delta}$ as a black box for one reason or another, then this will require some *engineering*: you need to measure the two-point block-block correlator at the output of a block

$$\mathbb{E}\left[\mathtt{R}^{(\ell+1)}_{i_1;\delta_1}\mathtt{R}^{(\ell+1)}_{i_2;\delta_2}\right] , \tag{B.40}$$

  and then compare it with the two-point correlator of the input to the block

$$\mathbb{E}\left[z^{(\ell)}_{i_1;\delta_1}z^{(\ell)}_{i_2;\delta_2}\right] . \tag{B.41}$$

  To be brief and concrete here, we'll focus on their diagonal components with $i_1 = i_2 \equiv i$ and $\delta_1 = \delta_2 \equiv \delta$. In particular, let us take an average over neural indices $i = 1, \ldots, n$ as well as over the sample indices $\delta \in \mathcal{D}$, so that we can compare two scalar quantities

$$\overline{G}^{(\ell+1)}_{\mathtt{RR}} \equiv \frac{1}{|\mathcal{D}|n}\sum_{i=1}^{n}\sum_{\delta\in\mathcal{D}}\mathbb{E}\left[\mathtt{R}^{(\ell+1)}_{i;\delta}\mathtt{R}^{(\ell+1)}_{i;\delta}\right] , \qquad \overline{G}^{(\ell)}_{zz} \equiv \frac{1}{|\mathcal{D}|n}\sum_{i=1}^{n}\sum_{\delta\in\mathcal{D}}\mathbb{E}\left[z^{(\ell)}_{i;\delta}z^{(\ell)}_{i;\delta}\right] \tag{B.42}$$

  rather than two matrices, (B.40) and (B.41).[8] After setting up these measurements, we need to adjust the *initialization hyperparameters* for the block such that these quantities have similar magnitudes. For instance, if internally the block is parameterized by many iterative layers – like our an MLP-$L_{\mathtt{R}}$-layer block – then we need to make sure that the difference in magnitudes is no worse than a *polynomial* in this depth hyperparameter $L_{\mathtt{R}}$; importantly, we want to ensure there's no exponential growth or decay in $\overline{G}^{(\ell+1)}_{\mathtt{RR}}/\overline{G}^{(\ell)}_{zz}$.[9] Note that while you're setting up measurements, it will also be helpful to measure the cross correlator

$$\overline{G}^{(\ell+0.5)}_{\mathtt{R}z} \equiv \frac{1}{|\mathcal{D}|n}\sum_{i=1}^{n}\mathbb{E}\left[\mathtt{R}^{(\ell+1)}_{i;\delta}z^{(\ell)}_{i;\delta}\right] , \tag{B.43}$$

[8] More generally, we should also account for off-diagonal components by averaging over pairs of inputs to estimate the appropriate analogs of $K^{(\ell)}_{[2]}$, cf. (5.19). We'd then also want to ensure that the analog of the recursion for this component, e.g. (B.14), is preserved. As our discussion in these bullets is somewhat schematic, we will leave these details to the PyTorch documentation.

[9] One reasonable heuristic solution to this problem, used by the *transformer* architecture, could be *layer normalization* [89]. However, more ideally, the block is not treated as a black box, and instead we use something about the structure of the block itself to find the criticality conditions.

which will be needed in the next bullet point.

- Now, using the forward equation (B.39), we can write a recursion for the diagonal component of the two-point correlator as

$$\overline{G}_{zz}^{(\ell+1)} = \gamma^2 \, \overline{G}_{zz}^{(\ell)} + 2\gamma\xi \, \overline{G}_{\mathtt{R}z}^{(\ell+0.5)} + \xi^2 \, \overline{G}_{\mathtt{RR}}^{(\ell+1)} \,, \tag{B.44}$$

  where we've suppressed the layer indices in the residual hyperparameters to declutter the expression.[10] To preserve this component of the two-point correlator of preactivations, we set the right-hand side of this equation equal to ℓ-th-layer correlator. Rearranging, we thus want

$$\left(1 - \gamma^2\right) \overline{G}_{zz}^{(\ell)} = \xi^2 \, \overline{G}_{\mathtt{RR}}^{(\ell+1)} + 2\gamma\xi \, \overline{G}_{\mathtt{R}z}^{(\ell+0.5)} \,. \tag{B.45}$$

  Since we've supposedly measured all these quantities, this equation should give simple *analytical* solutions for the residual hyperparameters.[11]

Overall, this hybrid approach realizes one of our goals of using experimentally measurable observables as input to an effective theory analysis. We hope that similar ways of thinking will lead to powerful ways of designing and tuning deep-learning models in the future.

[10] Note that unlike the MLP-single-layer block we discussed in §B.2, we cannot in general simply scale away ξ by a rescaling of C_W: for instance, if we have a deep MLP-$L_{\mathtt{R}}$-layer block with $L_{\mathtt{R}} \gg 1$ built with `ReLU` activations, then it is probably easier to set $C_W = 2$ and adjust ξ at the end. Thus, in general we should think of the initialization hyperparameters of the block as being fixed *first* – to ensure criticality of the block internally – and *then* for each γ, we tune $\xi(\gamma)$ according to (B.45) to ensure criticality of the whole network.

[11] Accordingly, each critical solution (γ, ξ) to the equation (B.45) will then yield architectures with different optimal aspect ratios $r^\star(\gamma, \xi)$. The optimal aspect ratio for a particular (γ, ξ) can be analogously estimated for general residual networks with a hybrid approach by combining a theoretical analysis as we did in §B.3 with measurements of an appropriate combination of four-point connected correlators.

References

[1] P. A. M. Dirac, *The Principles of Quantum Mechanics*. No. 27 in The International Series of Monographs on Physics. Oxford University Press, 1930.

[2] J. von Neumann, *Mathematical Foundations of Quantum Mechanics*. Princeton University Press, 1955.

[3] S. Carnot, *Reflections on the Motive Power of Heat and on Machines Fitted to Develop that Power*. J. Wiley, 1890. Trans. by R. H. Thurston from *Réflexions sur la puissance motrice du feu et sur les machines propres à développer cette puissance* (1824).

[4] M. Bessarab, *Landau*. M., Moscow worker, 1971. Trans. by B. Hanin from the original Russian source. `http://www.ega-math.narod.ru/Landau/Dau1971.htm`.

[5] J. Polchinski, "Memories of a Theoretical Physicist," `arXiv:1708.09093 [physics.hist-ph]`.

[6] F. Rosenblatt, "Principles of neurodynamics: Perceptrons and the theory of brain mechanism," tech. rep., Cornell Aeronautical Lab, Inc., 1961.

[7] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics* **5** no. 4, (1943) 115–133.

[8] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain.," *Psychological review* **65** no. 6, (1958) 386.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, pp. 1097–1105. 2012.

[10] K. Fukushima, "Neocognitron: a self organizing neural network model for a mechanism of pattern recognition unaffected by shift in position.," *Biological Cybernetics* **36** no. 4, (1980) 193–202.

[11] Y. LeCun, "Generalization and network design strategies," No. CRG-TR-89-4. 1989.

[12] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computation* **1** no. 4, (1989) 541–551.

[13] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE* **86** no. 11, (1998) 2278–2324.

[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30*, pp. 5998–6008. 2017. `arXiv:1706.03762 [cs.CL]`.

[15] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations*, D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, eds., ch. 8, pp. 318–362. MIT Press, Cambridge, MA, 1986.

[16] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural Networks: tricks of the trade*, pp. 9–48. Springer, 1998.

[17] Gallant and White, "There exists a neural network that does not make avoidable mistakes," in *IEEE 1988 International Conference on Neural Networks*, pp. 657–664 vol.1. 1988.

[18] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *International Conference on Machine Learning*. 2010.

[19] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 315–323, JMLR Workshop and Conference Proceedings. 2011.

[20] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *ICML Workshop on Deep Learning for Audio, Speech, and Language Processing*. 2013.

[21] C. Dugas, Y. Bengio, F. Bélisle, C. Nadeau, and R. Garcia, "Incorporating second-order functional knowledge for better option pricing," *Advances in Neural Information Processing Systems* **13** (2000) 472–478.

[22] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," `arXiv:1710.05941 [cs.NE]`.

[23] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," `arXiv:1606.08415 [cs.LG]`.

[24] A. M. Turing, "The chemical basis of morphogenesis," *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences* **237** no. 641, (1952) 37–72.

[25] A. M. Saxe, J. L. McClelland, and S. Ganguli, "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks," arXiv:1312.6120 [cs.NE].

[26] J. McGreevy, "Holographic duality with a view toward many-body physics," *Adv. High Energy Phys.* **2010** (2010) 723105, arXiv:0909.0518 [hep-th].

[27] R. M. Neal, "Priors for infinite networks," in *Bayesian Learning for Neural Networks*, pp. 29–53. Springer, 1996.

[28] J. Lee, Y. Bahri, R. Novak, S. S. Schoenholz, J. Pennington, and J. Sohl-Dickstein, "Deep neural networks as gaussian processes," in *International Conference on Learning Representations*. 2018. arXiv:1711.00165 [stat.ML].

[29] S. Yaida, "Non-Gaussian processes and neural networks at finite widths," in *Mathematical and Scientific Machine Learning Conference*. 2020. arXiv:1910.00019 [stat.ML].

[30] F. J. Dyson, "The s matrix in quantum electrodynamics," *Phys. Rev.* **75** (Jun, 1949) 1736–1755.

[31] J. Schwinger, "On the green's functions of quantized fields. i," *Proceedings of the National Academy of Sciences* **37** no. 7, (1951) 452–455.

[32] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Computer Vision – ECCV 2014*, pp. 818–833. 2014.

[33] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Advances in Neural Information Processing Systems 20*, pp. 1177–1184. 2008.

[34] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," arXiv:1810.04805 [cs.CL].

[35] M. Gell-Mann and F. E. Low, "Quantum electrodynamics at small distances," *Phys. Rev.* **95** (Sep, 1954) 1300–1312.

[36] K. G. Wilson, "Renormalization group and critical phenomena. i. renormalization group and the kadanoff scaling picture," *Phys. Rev. B* **4** (Nov, 1971) 3174–3183.

[37] K. G. Wilson, "Renormalization group and critical phenomena. ii. phase-space cell analysis of critical behavior," *Phys. Rev. B* **4** (Nov, 1971) 3184–3205.

[38] E. C. G. Stueckelberg de Breidenbach and A. Petermann, "Normalization of constants in the quanta theory," *Helv. Phys. Acta* **26** (1953) 499–520.

[39] N. Goldenfeld, *Lectures on phase transitions and the renormalization group*. CRC Press, 2018.

[40] J. Cardy, *Scaling and Renormalization in Statistical Physics*. Cambridge Lecture Notes in Physics. Cambridge University Press, 1996.

[41] M. Minsky and S. A. Papert, *Perceptrons: An Introduction to Computational Geometry*. MIT Press, 1988.

[42] S. Coleman, *Aspects of Symmetry: Selected Erice Lectures*. Cambridge University Press, Cambridge, U.K., 1985.

[43] B. Poole, S. Lahiri, M. Raghu, J. Sohl-Dickstein, and S. Ganguli, "Exponential expressivity in deep neural networks through transient chaos," in *Advances in Neural Information Processing Systems*, pp. 3360–3368. 2016. arXiv:1606.05340 [stat.ML].

[44] M. Raghu, B. Poole, J. Kleinberg, S. Ganguli, and J. Sohl-Dickstein, "On the expressive power of deep neural networks," in *International Conference on Machine Learning*, pp. 2847–2854. 2017. arXiv:1606.05336 [stat.ML].

[45] S. S. Schoenholz, J. Gilmer, S. Ganguli, and J. Sohl-Dickstein, "Deep information propagation," in *5th International Conference on Learning Representations*. 2017. arXiv:1611.01232 [stat.ML].

[46] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034. 2015. arXiv:1502.01852 [cs.CV].

[47] L. Kadanoff, "Critical Behavior. Universality and Scaling," in *Proceedings of the International School of Physics Enrico Fermi, Course LI (27 July – 8 August 1970)*. 1971.

[48] E. T. Jaynes, *Probability Theory: The Logic of Science*. Cambridge University Press, 2003.

[49] L. Froidmont, *On Christian Philosophy of the Soul*. 1649.

[50] D. J. MacKay, "Probable networks and plausible predictions—a review of practical bayesian methods for supervised neural networks," *Network: computation in neural systems* **6** no. 3, (1995) 469–505.

[51] C. K. I. Williams, "Computing with infinite networks," in *Advances in Neural Information Processing Systems 9*, p. 295–301. 1996. https://papers.nips.cc/paper/1197-computing-with-infinite-networks.

[52] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *NIPS Deep Learning and Representation Learning Workshop*. 2015. arXiv:1503.02531 [stat.ML].

[53] D. Hebb, *The Organization of Behavior: A Neuropsychological Theory*. Taylor & Francis, 2005.

[54] S. Coleman, "Sidney Coleman's Dirac Lecture 'Quantum Mechanics in Your Face'," arXiv:2011.12671 [physics.hist-ph].

[55] A. Jacot, F. Gabriel, and C. Hongler, "Neural tangent kernel: Convergence and generalization in neural networks," in *Advances in Neural Information Processing Systems*, pp. 8571–8580. 2018. arXiv:1806.07572 [cs.LG].

[56] S. Hochreiter, *Untersuchungen zu dynamischen neuronalen Netzen*. Diploma, Technische Universität München, 1991.

[57] Y. Bengio, P. Frasconi, and P. Simard, "The problem of learning long-term dependencies in recurrent networks," in *IEEE International Conference on Neural Networks*, vol. 3, pp. 1183–1188, IEEE. 1993.

[58] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International Conference on Machine Learning*, pp. 1310–1318, PMLR. 2013. arXiv:1211.5063 [cs.LG].

[59] M. Kline, *Mathematical Thought From Ancient to Modern Times: Volume 3*. Oxford University Press, 1990.

[60] J. Lee, L. Xiao, S. Schoenholz, Y. Bahri, R. Novak, J. Sohl-Dickstein, and J. Pennington, "Wide neural networks of any depth evolve as linear models under gradient descent," in *Advances in Neural Information Processing Systems 32*, pp. 8572–8583. 2019. arXiv:1902.06720 [stat.ML].

[61] E. Fix and J. Hodges, "Discriminatory analysis. nonparametric discrimination: Consistency properties," *USAF School of Aviation Medicine, Project Number: 21-49-004, Report Number: 4* (1951) .

[62] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theory* **13** (1967) 21–27.

[63] A. Einstein, "On the method of theoretical physics," *Philosophy of Science* **1** no. 2, (1934) 163–169.

[64] B. Hanin and M. Nica, "Finite depth and width corrections to the neural tangent kernel," in *International Conference on Learning Representations*. 2020. arXiv:1909.05989 [cs.LG].

[65] E. Dyer and G. Gur-Ari, "Asymptotics of wide networks from feynman diagrams," in *International Conference on Learning Representations*. 2020. arXiv:1909.11304 [cs.LG].

[66] G. F. Giudice, "Naturally Speaking: The Naturalness Criterion and Physics at the LHC," arXiv:0801.2562 [hep-ph].

[67] F. J. Dyson, "Forward," in *Classic Feynman: All the Adventures of a Curious Character*, R. Leighton, ed., pp. 5–9. W. W. Norton & Company Ltd., 2006.

[68] L. Chizat, E. Oyallon, and F. Bach, "On lazy training in differentiable programming," in *Advances in Neural Information Processing Systems*, vol. 32. 2019. arXiv:1812.07956 [math.OC].

[69] D. J. MacKay, *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 2003.

[70] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei, "Scaling laws for neural language models," arXiv:2001.08361 [cs.LG].

[71] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," arXiv:1611.03530 [cs.LG].

[72] D. H. Wolpert, "The lack of a priori distinctions between learning algorithms," *Neural computation* **8** no. 7, (1996) 1341–1390.

[73] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE transactions on evolutionary computation* **1** no. 1, (1997) 67–82.

[74] L. Boltzmann, "On certain questions of the theory of gases," *Nature* **51** no. 1322, (1895) 413–415.

[75] L. Boltzmann, *Lectures on Gas Theory*. Berkeley, University of California Press, 1964. Trans. by S. G. Brush from *Vorlesungen ueber Gastheorie* (2 vols., 1896 & 1898).

[76] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal* **27** no. 3, (1948) 379–423.

[77] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal* **27** no. 4, (1948) 623–656.

[78] E. T. Jaynes, "Information theory and statistical mechanics," *Phys. Rev.* **106** (May, 1957) 620–630.

[79] E. T. Jaynes, "Information theory and statistical mechanics. ii," *Phys. Rev.* **108** (Oct, 1957) 171–190.

[80] Y. LeCun, J. Denker, and S. Solla, "Optimal brain damage," in *Advances in Neural Information Processing Systems*, D. Touretzky, ed., vol. 2. Morgan-Kaufmann, 1990.

[81] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," in *International Conference on Learning Representations*. 2019. arXiv:1803.03635 [cs.LG].

[82] T. Banks and A. Zaks, "On the phase structure of vector-like gauge theories with massless fermions," *Nuclear Physics B* **196** no. 2, (1982) 189–204.

[83] R. Linsker, "Self-organization in a perceptual network," *Computer* **21** no. 3, (1988) 105–117.

[84] S. Becker and G. E. Hinton, "Self-organizing neural network that discovers surfaces in random-dot stereograms," *Nature* **355** no. 6356, (1992) 161–163.

[85] B. Gale, R. Zemeckis, M. J. Fox, and C. Lloyd, *Back to the Future Part II.* Universal Pictures, 1989.

[86] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778. 2016.

[87] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, pp. 448–456. 2015. `arXiv:1502.03167 [cs.LG]`.

[88] A. Veit, M. Wilber, and S. Belongie, "Residual networks behave like ensembles of relatively shallow networks," in *Advances in Neural Information Processing Systems*, vol. 30, pp. 550–558. 2016. `arXiv:1605.06431 [cs.CV]`.

[89] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," in *Deep Learning Symposium, Neural Information Processing Systems*. 2016. `arXiv:1607.06450 [stat.ML]`.