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 have 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*

https://doi.org/10.1017/9781009023405.002 Published online by Cambridge University Press

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.

https://doi.org/10.1017/9781009023405.002 Published online by Cambridge University Press

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.

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].

https://doi.org/10.1017/9781009023405.002 Published online by Cambridge University Press

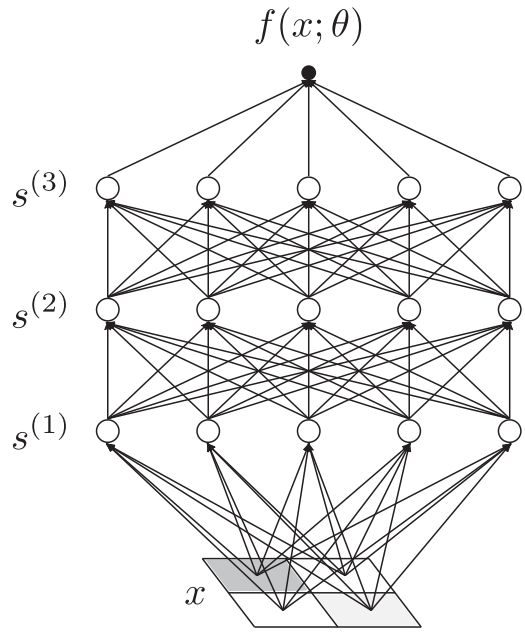

Figure 0.1 A graph of a simple multilayer neural network, depicting how the input x is transformed through a sequence of intermediate signals, $s^{(1)}$, $s^{(2)}$, and $s^{(3)}$, into the output $f(x;\theta)$. The white circles represent the neurons, the black dot at the top represents the network output, 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.

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 0.1.

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

https://doi.org/10.1017/9781009023405.002 Published online by Cambridge University Press

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 $(x, f(x))$ 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 $(x, f(x))$ 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

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

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

https://doi.org/10.1017/9781009023405.002 Published online by Cambridge University Press

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, $df/d\theta$, $d^2f/d\theta^2$, ... , 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, $df/d\theta$, $d^2f/d\theta^2$, ... – 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

https://doi.org/10.1017/9781009023405.002 Published online by Cambridge University Press

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

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 principal 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 a 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 provide some 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 0.1. 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.

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 natural 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,

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

https://doi.org/10.1017/9781009023405.002 Published online by Cambridge University Press

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

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

  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 compute.

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 detail 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, $s^{(1)}$, $s^{(2)}, \ldots$, will remain unchanged from initialization, leading to *random representations* and thus severely restricting the type 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

https://doi.org/10.1017/9781009023405.002 Published online by Cambridge University Press

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:

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

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 $1/n^2$ 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**.

https://doi.org/10.1017/9781009023405.002 Published online by Cambridge University Press

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 account 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]

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.

[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.

https://doi.org/10.1017/9781009023405.002 Published online by Cambridge University Press