Deep Learning for Computer Architects

Synthesis Lectures on Computer Architecture

Synthesis Lectures on Computer Architecture publishes 50- to 100-page publications on topics pertaining to the science and art of designing, analyzing, selecting and interconnecting hardware components to create computers that meet functional, performance and cost goals. The scope will largely follow the purview of premier computer architecture conferences, such as ISCA, HPCA, MICRO, and ASPLOS.

The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines, Second edition
Luiz André Barroso, Jimmy Clidaras, and Urs Hölzle
2013

Shared-Memory Synchronization
Michael L. Scott
2013

Resilient Architecture Design for Voltage Variation
Vijay Janapa Reddi and Meeta Sharma Gupta
2013

Multithreading Architecture
Mario Nemirovsky and Dean M. Tullsen
2013

Performance Analysis and Tuning for General Purpose Graphics Processing Units (GPGPU)
Hyesoon Kim, Richard Vuduc, Sara Baghsorkhi, Jee Choi, and Wen-mei Hwu
2012

Automatic Parallelization: An Overview of Fundamental Compiler Techniques
Samuel P. Midkiff
2012

Phase Change Memory: From Devices to Systems
Moinuddin K. Qureshi, Sudhanva Gurumurthi, and Bipin Rajendran
2011

Multi-Core Cache Hierarchies
Rajeev Balasubramonian, Norman P. Jouppi, and Naveen Muralimanohar
2011

A Primer on Memory Consistency and Cache Coherence
Daniel J. Sorin, Mark D. Hill, and David A. Wood
2011

Dynamic Binary Modification: Tools, Techniques, and Applications
Kim Hazelwood
2011

Quantum Computing for Computer Architects, Second Edition
Tzvetan S. Metodi, Arvin I. Faruque, and Frederic T. Chong
2011

High Performance Datacenter Networks: Architectures, Algorithms, and Opportunities
Dennis Abts and John Kim
2011

Processor Microarchitecture: An Implementation Perspective
Antonio González, Fernando Latorre, and Grigorios Magklis
2010

Transactional Memory, 2nd edition
Tim Harris, James Larus, and Ravi Rajwar
2010

Computer Architecture Performance Evaluation Methods
Lieven Eeckhout
2010

Introduction to Reconfigurable Supercomputing
Marco Lanzagorta, Stephen Bique, and Robert Rosenberg
2009

On-Chip Networks
Natalie Enright Jerger and Li-Shiuan Peh
2009

The Memory System: You Can't Avoid It, You Can't Ignore It, You Can't Fake It
Bruce Jacob
2009

Fault Tolerant Computer Architecture
Daniel J. Sorin
2009

The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines
Luiz André Barroso and Urs Hölzle
2009

Computer Architecture Techniques for Power-Efficiency
Stefanos Kaxiras and Margaret Martonosi
2008

Chip Multiprocessor Architecture: Techniques to Improve Throughput and Latency
Kunle Olukotun, Lance Hammond, and James Laudon
2007

Transactional Memory
James R. Larus and Ravi Rajwar
2006

Quantum Computing for Computer Architects
Tzvetan S. Metodi and Frederic T. Chong
2006

Deep Learning for Computer Architects

Brandon Reagen, Robert Adolf, Paul Whatmough, Gu-Yeon Wei, and David Brooks

www.morganclaypool.com

ISBN: 9781627057288 paperback
ISBN: 9781627059855 ebook

DOI 10.2200/S00783ED1V01Y201706CAC041

A Publication in the Morgan & Claypool Publishers series
SYNTHESIS LECTURES ON COMPUTER ARCHITECTURE

Lecture #41
Series Editor: Margaret Martonosi, *Princeton University*
Founding Editor Emeritus: Mark D. Hill, *University of Wisconsin, Madison*
Series ISSN
Print 1935-3235 Electronic 1935-3243

Deep Learning for Computer Architects

Brandon Reagen
Harvard University

Robert Adolf
Harvard University

Paul Whatmough
ARM Research and Harvard University

Gu-Yeon Wei
Harvard University

David Brooks
Harvard University

SYNTHESIS LECTURES ON COMPUTER ARCHITECTURE #41

MORGAN & CLAYPOOL PUBLISHERS

ABSTRACT

Machine learning, and specifically deep learning, has been hugely disruptive in many fields of computer science. The success of deep learning techniques in solving notoriously difficult classification and regression problems has resulted in their rapid adoption in solving real-world problems. The emergence of deep learning is widely attributed to a virtuous cycle whereby fundamental advancements in training deeper models were enabled by the availability of massive datasets and high-performance computer hardware.

This text serves as a primer for computer architects in a new and rapidly evolving field. We review how machine learning has evolved since its inception in the 1960s and track the key developments leading up to the emergence of the powerful deep learning techniques that emerged in the last decade. Next we review representative workloads, including the most commonly used datasets and seminal networks across a variety of domains. In addition to discussing the workloads themselves, we also detail the most popular deep learning tools and show how aspiring practitioners can use the tools with the workloads to characterize and optimize DNNs.

The remainder of the book is dedicated to the design and optimization of hardware and architectures for machine learning. As high-performance hardware was so instrumental in the success of machine learning becoming a practical solution, this chapter recounts a variety of optimizations proposed recently to further improve future designs. Finally, we present a review of recent research published in the area as well as a taxonomy to help readers understand how various contributions fall in context.

KEYWORDS

deep learning, neural network accelerators, hardware software co-design, DNN benchmarking and characterization, hardware support for machine learning

Contents

Preface

This book is intended to be a general introduction to neural networks for those with a computer architecture, circuits, or systems background. In the introduction (Chapter 1), we define key vocabulary, recap the history and evolution of the techniques, and for make the case for additional hardware support in the field.

We then review the basics of neural networks from linear regression to perceptrons and up to today's state-of-the-art deep neural networks (Chapter 2). The scope and language is presented such that anyone should be able to follow along, and the goal is to get the community on the same page. While there has been an explosion of interest in the field, evidence suggests many terms are being conflated and that there are gaps in understandings in the area. We hope that what is presented here dispels rumors and provides common ground for nonexperts.

Following the review, we dive into tools, workloads, and characterization. For the practitioner, this may be the most useful chapter. We begin with an overview of modern neural network and machine learning software packages (namely TensorFlow, Torch, Keras, and Theano) and explain their design choices and differences to guide the reader to choosing the right tool for their work. In the second half of Chapter 3, we present a collection of commonly used, seminal workloads that we have assembled in a benchmark suite named Fathom [2]. The workloads are broken down into two categories: dataset and model, with explanation of why the workload and/or dataset is seminal as well as how it should be used. This section should also help reviewers of neural network papers better judge contributions. By having a better understanding of each of the workloads, we feel that more thoughtful interpretations of ideas and contributions are possible. Included with the benchmark is a characterization of the workloads on both a CPU and GPU.

Chapter 4 builds off of Chapter 3 and is likely of greatest interest to the architect looking to investigate accelerating neural networks with custom hardware. In this chapter, we review the Minerva accelerator design and optimization framework [114] and include details of how high-level neural network software libraries can be used in conglomeration with hardware CAD and simulation flows to codesign the algorithms and hardware. We specifically focus on the Minerva methodology and how to experiment with neural network accuracy and power, performance, and area hardware trade-offs. After reading this chapter, a graduate student should feel confident in evaluating their own accelerator/custom hardware optimizations.

In Chapter 5, we present a thorough survey of relevant hardware for neural network papers, and develop a taxonomy to help the reader understand and contrast different projects. We primarily focus on the past decade and group papers based on the level in the compute stack they address (algorithmic, software, architecture, or circuits) and by optimization type (sparsity,

quantization, arithmetic approximation, and fault tolerance). The survey primarily focuses on the top machine learning, architecture, and circuit conferences; this survey attempts to capture the most relevant works for architects in the area at the time of this book's publication. The truth is there are just too many publications to possibly include them all in one place. Our hope is that the survey acts instead as a starting point; that the taxonomy provides order such that interested readers know where to look to learn more about a specific topic; and that the casual participant in hardware support for neural networks finds here a means of comparing and contrasting related work.

Finally, we conclude by dispelling any myths that hardware for deep learning research has reached its saturation point by suggesting what more remains to be done. Despite the numerous papers on the subject, we are far from done, even within supervised learning. This chapter sheds light on areas that need attention and briefly outlines other areas of machine learning. Moreover, while hardware has largely been a service industry for the machine learning community, we should really begin to think about how we can leverage modern machine learning to improve hardware design. This is a tough undertaking as it requires a true understanding of the methods rather than implementing existing designs, but if the past decade of machine learning has taught us anything, it is that these models work well. Computer architecture is among the least formal fields in computer science (being almost completely empirical and intuition based). Machine learning may have the most to offer in terms of rethinking how we design hardware, including Bayesian optimization, and shows how beneficial these techniques can be in hardware design.

Brandon Reagen, Robert Adolf, Paul Whatmough, Gu-Yeon Wei, and David Brooks
July 2017

CHAPTER 1

Introduction

Machine learning has been capturing headlines for its successes in notoriously difficult artificial intelligence problems. From the decisive victories of DeepMind's AlphaGo system against top-ranked human Go players to the wonder of self-driving cars navigating city streets, the impact of these methods are being felt far and wide. But the mathematical and computational foundations of machine learning are not magic: these are methods that have been developed gradually over the better part of a century, and it is a part of computer science and mathematics just like any other.

So what is machine learning? One way to think about it is as a way of programming with data. Instead of a human expert crafting an explicit solution to some problem, a machine learning approach is implicit: a human provides a set of rules and data, and a computer uses both to arrive at a solution automatically. This shifts the research and engineering burden from identifying specific one-off solutions to developing indirect methods that can be applied to a variety of problems. While this approach comes with a fair number its own challenges, it has the potential to solve problems for which we have no known heuristics and to be applied broadly.

The focus of this book is on a specific type of machine learning: neural networks. Neural networks can loosely be considered the computational analog of a brain. They consist of a myriad of tiny elements linked together to produce complex behaviors. Constructing a practical neural network piece by piece is beyond human capability, so, as with other machine learning approaches, we rely on indirect methods to build them. A neural network might be given pictures and taught to recognize objects or given recordings and taught to transcribe their contents. But perhaps the most interesting feature of neural networks is just how long they have been around. The harvest being reaped today was sown over the course of many decades. So to put current events into context, we begin with a historical perspective.

1.1 THE RISES AND FALLS OF NEURAL NETWORKS

Neural networks have been around since nearly the beginning of computing, but they have something of a checkered past. Early work, such as that of McCulloch and Pitts [104], was focused on creating mathematical models similar to biological neurons. Attempts at recreating brain-like behavior in hardware started in the 1950s, best exemplified by the work of Rosenblatt on his "perceptron" [117]. But interest in neural networks (both by researchers and the general public) has waxed and waned over the years. Years of optimistic enthusiasm gave way to disillusionment, which in turn was overcome again by dogged persistence. The waves of prevailing opinion are

shown in Figure 1.1 overlaid on a timeline of major events that have influenced where neural networks are today. The hype generated by Rosenblatt in 1957 was quashed by Minsky and Papert in their seminal book *Perceptrons* [106]. In the book, the authors dismissed what they saw as overpromises and highlighted the technical limits of perceptrons themselves. It was famously shown that a single perceptron was incapable of learning some simple types of functions such as XOR. There were other rumblings at the time that perceptrons were not as significant as they were made out to be, mainly from the artificial intelligence community, which felt perceptrons oversimplified the difficulty of the problems the field was attempting to solve. These events precipitated the first *AI winter*, where interest and funding for machine learning (both in neural networks and in artificial intelligence more broadly) dissolved almost entirely.

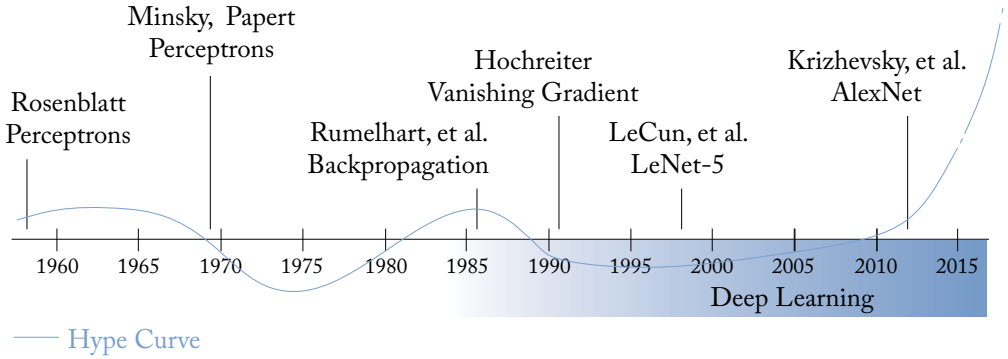


Figure 1.1: A review of the rise and fall of neural networks over years. Highlights the three major peaks: early 1960s (Rosenblatt), mid-1980s (Rumelhart), and now (deep learning).

After a decade in the cold, interest in neural networks began to pick up again as researchers started to realize that the critiques of the past may have been too harsh. A new flourishing of research introduced larger networks and new techniques for tuning them, especially on a type of computing called *parallel distributed processing*—large numbers of neurons working simultaneously to achieve some end. The defining paper of the decade came when Rumelhart, Hinton, and Williams proposed their method for backpropagation in 1986 [119]. While others have rightly been credited for inventing the technique earlier (most notably Paul Werbos who had proposed the technique twelve years before [140]), this brought backpropagation into the mainstream and changed attitudes toward neural networks. Backpropagation leveraged a simple calculus to allow for networks of arbitrary structure to be trained efficiently. Perhaps most important, this allowed for more complicated, hierarchical neural nets. This, in turn, expanded the set of problems that could be attacked and sparked interest in practical applications.

Despite this remarkable progress, overenthusiasm and hype once again contributed to looming trouble. In fact, Minsky (who partially instigated and weathered the first winter) was one of the first to warn that the second winter would come if the hype did not die down. To

keep research money flowing, researchers began to promise more and more, and when they fell short of delivering on their promises, many funding agencies became disillusioned with the field as a whole. Notable examples of this include the Lighthill report from England [103] and the cancellation of the Speech Understanding Research program from DARPA in favor of more traditional systems. This downturn was accompanied by a newfound appreciation for the complexity of neural networks. As especially pointed out by Lighthill, for these models to be useful in solving real-world problems would require incredible amounts of computational power that simply was not available at the time.

While the hype died down and the money dried up, progress still moved on in the background. During the second AI winter, which lasted from the late 1980s until the mid-2000s, many substantial advances were still made. For example, the development of convolutional neural networks [89] (see Section 3.1.1) in the 1990s quietly grew out of similar models introduced earlier (e.g., the Neocognitron [48]). However, two decades would pass before widespread interest in neural networks would arise again.

1.2 THE THIRD WAVE

In the late 2000s, the second AI winter began to thaw. While many advances had been made on algorithms and theory for neural networks, what made this time around different was the setting to which neural networks awoke. As a whole, since the late 1980s, the computing landscape had changed. From the Internet and ubiquitous connectivity to smart phones and social media, the sheer volume of data being generated was overwhelming. At the same time, computing hardware continued to follow Moore's law, growing exponentially through the AI winter. The world's most powerful computer at the end of the 1980s was literally equivalent to a smart phone by 2010. Problems that used to be completely infeasible suddenly looked very realistic.

1.2.1 A VIRTUOUS CYCLE

This dramatic shift in circumstances began to drive a self-reinforcing cycle of progress and opportunity (Figure 1.2). It was the interplay between these three areas—data, algorithms, and computation—that was directly responsible for the third revival of neural networks. Each was significant on its own, but the combined benefits were more profound still.

These key factors form a virtuous cycle. As more complex, larger datasets become available, new neural network techniques are invented. These techniques typically involve larger models with mechanisms that also require more computations per model parameter. Thus, the limits of even today's most powerful, commercially available devices are being tested. As more powerful hardware is made available, models will quickly expand to consume and use every available device. The relationship between large datasets, algorithmic training advances, and high-performance hardware forms a virtuous cycle. Whenever advances are made in one, it fuels the other two to advance.

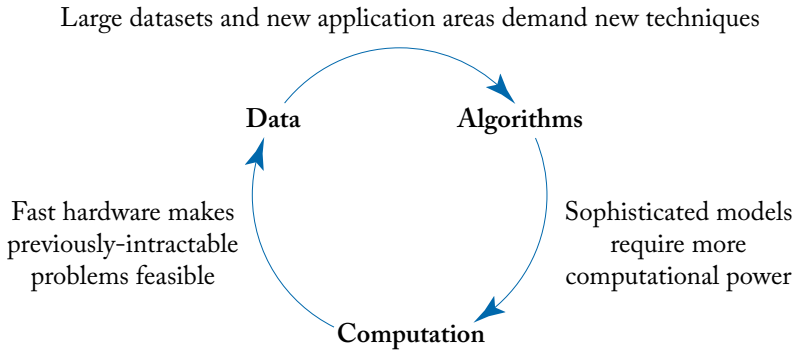


Figure 1.2: The virtuous cycle underpinning the modern deep learning revival.

Big Data With the rise of the Internet came a deluge of data. By the early 2000s, the problem was rarely obtaining data but instead, trying to make sense of it all. Rising demand for algorithms to extract patterns from the noise neatly fit with many machine learning methods, which rely heavily on having a surfeit of data to operate. Neural networks in particular stood out: compared with simpler techniques, neural nets tend to scale better with increasing data. Of course, with more data and increasingly complicated algorithms, ever more powerful computing resources were needed.

Big Ideas Many of the thorny issues from the late 1980s and early 1990s were still ongoing at the turn of the millennium, but progress had been made. New types of neural networks took advantage of domain-specific characteristics in areas such as image and speech processing, and new algorithms for optimizing neural nets began to whittle away at the issues that had stymied researchers in the prior decades. These ideas, collected and built over many years, were dusted off and put back to work. But instead of megabytes of data and megaflops of computational power, these techniques now had millions of times more resources to draw upon. Moreover, as these methods began to improve, they moved out of the lab and into the wider world. Success begot success, and demand increased for still more innovation.

Big Iron Underneath it all was the relentless juggernaut of Moore's law. To be fair, computing hardware was a trend unto itself; with or without a revival of neural networks, the demand for more capability was as strong as ever. But the improvements in computing resonated with machine learning somewhat differently. As frequency scaling tapered off in the early 2000s, many application domains struggled to adjust to the new realities of parallelism. In contrast, neural networks excel on parallel hardware by their very nature. As the third wave was taking off, computer processors were shifting toward architectures that looked almost tailor-made for the new algorithms and massive datasets. As machine learning continues to grow in prevalence, the influence it has on hardware design is increasing.

This should be especially encouraging to the computer architect looking to get involved in the field of machine learning. While there has been an abundance of work in architectural support for neural networks already, the virtuous cycle suggests that there will be demand for new solutions as the field evolves. Moreover, as we point out in Chapter 3, the hardware community has only just begun to scratch the surface of what is possible with neural networks and machine learning in general.

1.3 THE ROLE OF HARDWARE IN DEEP LEARNING

Neural networks are inveterate computational hogs, and practitioners are constantly looking for new devices that might offer more capability. In fact, as a community, we are already in the middle of the second transition. The first transition, in the late 2000s, was when researchers discovered that commodity GPUs could provide significantly more throughput than desktop CPUs. Unlike many other application domains, neural networks had no trouble making the leap to a massively data-parallel programming model—many of the original algorithms from the 1980s were already formulated this way anyway. As a result, GPUs have been the dominant platform for neural networks for several years, and many of the successes in the field were the result of clusters of graphics cards churning away for weeks on end [135].

Recently, however, interest has been growing in dedicated hardware approaches. The reasoning is simple: with sufficient demand, specialized solutions can offer better performance, lower latency, lower power, or whatever else an application might need compared to a generic system. With a constant demand for more cycles from the virtuous cycle mentioned above, opportunities are growing.

1.3.1 STATE OF THE PRACTICE

With neural networks' popularity and amenability to hardware acceleration, it should come as no surprise that countless publications, prototypes, and commercial processors exist. And while it may seem overwhelming, it is in fact just the tip of the iceberg. In this section, we give a brief look at the state of the art to highlight the advances that have been made and what more needs to be done. Chapter 5 provides a more comprehensive look at the field.

To reason quantitatively about the field, we look at a commonly used research dataset called *MNIST*. MNIST is a widely studied problem used by the machine learning community as a sort of lowest common denominator. While it is no longer representative of real-world applications, it serves a useful role. MNIST is small, which makes it easier to dissect and understand, and the wealth of prior experiments on the problem means comparison is more straightforward. (An excellent testimonial as to why datasets like MNIST are imperative to the field of machine learning is given by Roger Grosse [56].)

Figure 1.3 shows the results of a literature survey on hardware and software implementations of neural networks that can run with the MNIST dataset. The details of the problem are not important, but the basic idea is that these algorithms are attempting to differentiate be-

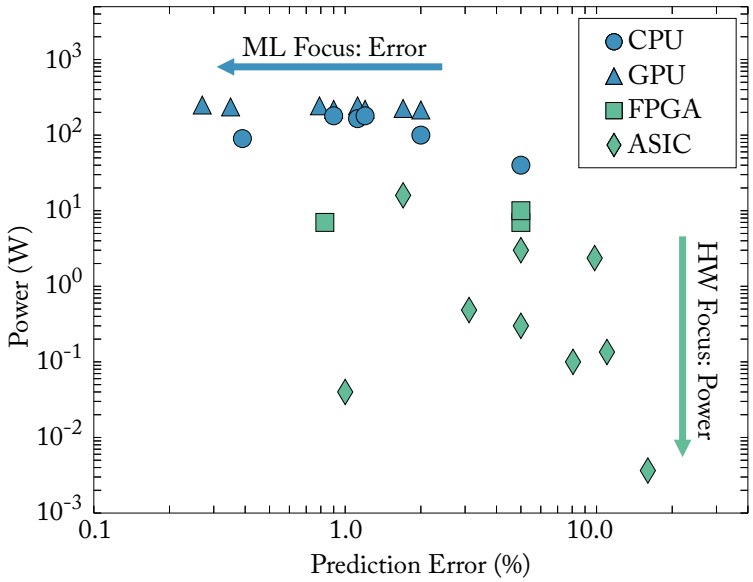


Figure 1.3: A literature survey of neural networks reveals the disconnect between machine learning research and computer architecture communities when designing neural network systems. References: CPUs [47, 63, 112, 130, 139], GPUs [24, 30, 32, 63, 128, 130, 139], FPGAs [47, 57], and ASICs [7, 18, 24, 46, 47, 83, 88, 132].

tween images of single handwritten digits (0–9). On the x-axis is the prediction error of a given network: a 1% error rate means the net correctly matched a 99% of the test images with their corresponding digit. On the y-axis is the power consumed by the neural network on the platform it was tested on. The most interesting trend here is that the machine learning community (blue) has historically focused on minimizing prediction error and favors the computational power of GPUs over CPUs, with steady progress toward the top left of the plot. In contrast, solutions from the hardware community (green) trend toward the bottom right of the figure, emphasizing practical efforts to constrain silicon area and reduce power consumption at the expense of nonnegligible reductions in prediction accuracy.

The divergent trends observed for published machine learning vs. hardware results reveal a notable gap for implementations that achieve competitive prediction accuracy with power budgets within the grasp of mobile and IoT platforms. A theme throughout this book is how to approach this problem and reason about accuracy-performance trade-offs rigorously and scientifically.

This book serves as a review of the state of the art in deep learning, in a manner that is suitable for architects. In addition, it presents workloads, metrics of interest, infrastructures, and future directions to guide readers through the field and understand how different works compare.

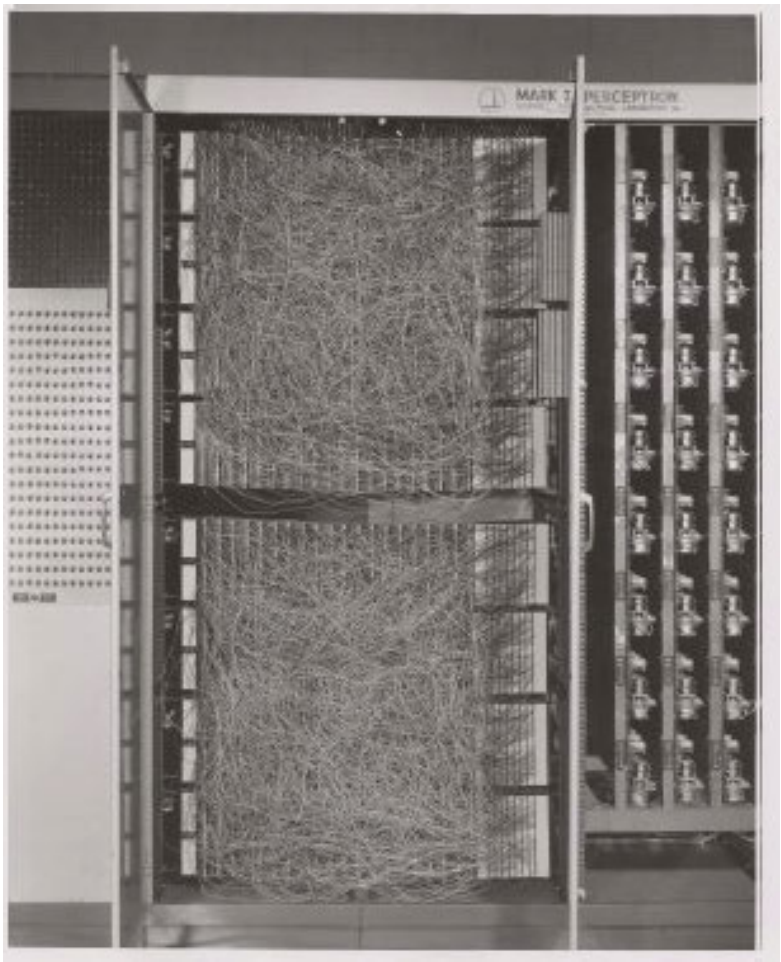


Figure 1.4: A photo of one of the first neural network prototypes, built by Frank Rosenblatt at Cornell. Many early neural networks were designed with specialized hardware at their core: Rosenblatt developed the Mark I Perceptron machine in the 1960s, and Minsky built SNARC in 1951. It seems fitting now that computer architecture is once again playing a large role in the lively revival of neural networks.

CHAPTER 2

Foundations of Deep Learning

2.1 NEURAL NETWORKS

The seat of human thought has long inspired scientists and philosophers, but it wasn't until the 20th century that the mechanisms behind the brain began to be unlocked. Anatomically, the brain is a massive network of simple, interconnected components: a neural network. Thought, memory, and perception are products of the interactions between the elements of this network, the result of thousands of signals being exchanged. Modern deep learning methods can trace their origins to computational analogs of biological neural networks, so it helps to have a basic understanding of their functionality.

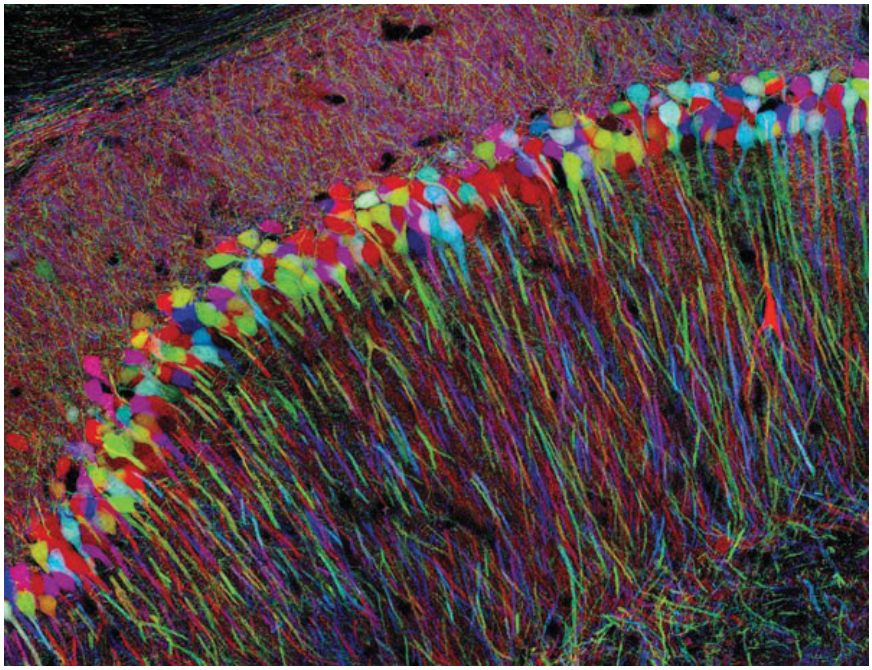
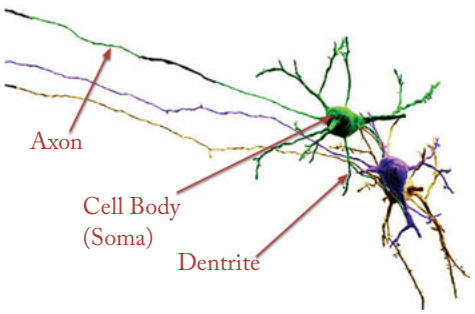


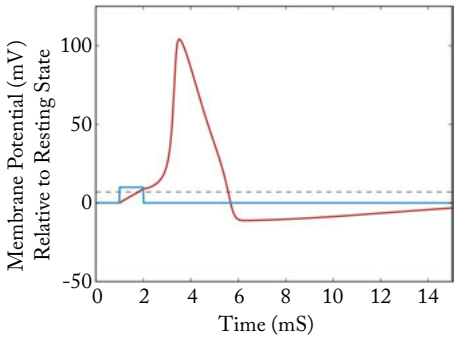
Figure 2.1: Local neuron connectivity in a mouse brain. True color image produced by induced expression of fluorescent protein [99]. Each colored patch is a single neuron cell.

2.1.1 BIOLOGICAL NEURAL NETWORKS

The fundamental unit of the brain is a neuron (Figure 2.2a), a specialized cell that is designed to pass electrochemical signals to other parts of the nervous system, including other neurons. A neuron is comprised of three basic pieces: the soma, or body, which contains the nucleus, metabolic machinery, and other organelles common to all cells; dendrites, which act as a neuron's inputs, receiving signals from other neurons and from sensory pathways; and an axon, an elongated extension of the cell responsible for transmitting outgoing signals. Connections between neurons are formed where an axon of one neuron adjoins the dendrite of another at an intercellular boundary region called a *synapse*. The long, spindly structure of neurons enables a dense web, with some neurons having up to 200,000 different direct connections [102].



(a) Annotated visualization of the structure of a biological neuron, reconstructed from electron microscope images of 30 nm slices of a mouse brain [138].



(b) The shape of an action potential. A small external voltage stimulus (blue) triggers a cascade of charge buildup inside a neuron (red) via voltage-gated ion channels. The activation threshold is shown as a dotted line. Simulated using a Hodgkin-Huxley model of a neuron [73] (Nobel prize in Medicine, 1963).

Figure 2.2: The structure and operation of biological neurons.

Unlike most biological signaling mechanisms, communication between two neurons is an electrical phenomenon. The signal itself, called an *action potential*, is a temporary spike in local electrical charge difference across the cell membrane (see Figure 2.2b). This spike is the result of specialized membrane proteins called *voltage-gated channels*, which act as voltage-controlled ion pumps. At rest, these ion channels are closed, but when a small charge difference builds up (either through the result of a chemical stimulus from a sensory organ or from an induced charge from another neuron's synapse), one type of protein changes shape and locks into an open position, allowing sodium ions to flow into the cell, amplifying the voltage gradient. When the voltage reaches a certain threshold, the protein changes conformation again, locking itself into a closed state. Simultaneously, a second transport protein switches itself open, allowing potassium