

DEEP LEARNING

JOHN D. KELLEHER

The MIT Press Essential Knowledge Series

A complete list of the titles in this series appears at the back of this book.

The MIT Press | Cambridge, Massachusetts | London, England

CONTENTS

Series Foreword	vii
Preface	ix
Acknowledgments	xi

© 2019 The Massachusetts Institute of Technology

All rights reserved. No part of this book may be reproduced in any form by any electronic or mechanical means (including photocopying, recording, or information storage and retrieval) without permission in writing from the publisher.

This book was set in Chaparral Pro by Toppan Best-set Premedia Limited.
Printed and bound in the United States of America.

Library of Congress Cataloging-in-Publication Data

Names: Kelleher, John D., 1974- author.

Title: Deep learning / John D. Kelleher.

Description: Cambridge, MA : The MIT Press, [2019] | Series:

The MIT press essential knowledge series | Includes bibliographical references and index.

Identifiers: LCCN 2018059550 | ISBN 9780262537551 (pbk. : alk. paper)

Subjects: LCSH: Machine learning. | Artificial intelligence.

Classification: LCC Q325.5 .K454 2019 | DDC 006.3/1—dc23 LC record available at <https://lcn.loc.gov/2018059550>

1098765

1	Introduction to Deep Learning	1
2	Conceptual Foundations	39
3	Neural Networks: The Building Blocks of Deep Learning	65
4	A Brief History of Deep Learning	101
5	Convolutional and Recurrent Neural Networks	159
6	Learning Functions	185
7	The Future of Deep Learning	231

Glossary	251
Notes	257
References	261
Further Readings	267
Index	269

the network using large datasets. By contrast, hyperparameters are the parameters of a model (in these cases, the parameters of a neural network architecture) and/or training algorithm that cannot be directly estimated from the data but instead must be specified by the person creating the model, either through the use of heuristic rules, intuition, or trial and error. Often, much of the effort that goes into the creation of a deep learning network involves experimental work to answer the questions in relation to hyperparameters, and this process is known as hyperparameter tuning. The next chapter will review the history and evolution of deep learning, and the challenges posed by many of these questions are themes running through the review. Subsequent chapters in the book will explore how answering these questions in different ways can create networks with very different characteristics, each suited to different types of tasks. For example, recurrent neural networks are best suited to processing sequential/time-series data, whereas convolutional neural networks were originally developed to process images. Both of these network types are, however, built using the same fundamental processing unit, the artificial neuron; the differences in the behavior and abilities of these networks stems from how these neurons are arranged and composed.

A BRIEF HISTORY OF DEEP LEARNING

The history of deep learning can be described as three major periods of excitement and innovation, interspersed with periods of disillusionment. Figure 4.1 shows a timeline of this history, which highlights these periods of major research: on threshold logic units (early 1940s to the mid 1960s), connectionism (early 1980s to mid-1990s), and deep learning (mid 2000s to the present). Figure 4.1 distinguishes some of the primary characteristics of the networks developed in each of these three periods. The changes in these network characteristics highlight some of the major themes within the evolution of deep learning, including: the shift from binary to continuous values; the move from threshold activation functions, to logistic and tanh activation, and then onto ReLU activation; and the progressive deepening of the networks, from single layer,

by the value of the derivative of the logistic function at the appropriate activation for the neuron, the maximum value the gradient will have is a quarter of the gradient prior to the multiplication. Another problem with using the logistic function is that there are large portions of the domain of the function where the function is *saturated* (returning values that very close to 0 or 1), and the rate of change of the function in these regions is near zero; thus, the derivative of the function is near 0. This is an undesirable property when backpropagating error gradients because the error gradients will be forced to zero (or close to zero) when backpropagated through any neuron whose activation is within one of these saturated regions. In 2011 it was shown that switching to a rectified linear activation function, $g(x) = \max(0, x)$, improved training for deep feedforward neural networks (Glorot et al. 2011). Neurons that use a rectified linear activation function are known as rectified linear units (ReLU). One advantage of ReLUs is that the activation function is linear for the positive portion of its domain with a derivative equal to 1. This means that gradients can flow easily through ReLUs that have positive activation. However, the drawback of ReLUs is that the gradient of the function for the negative part of its domain is zero, so ReLUs do not train in this portion of the domain. Although undesirable, this is not necessarily a fatal flaw for learning because when backpropagating through a layer of ReLUs the gradients

can still flow through the ReLUs in the layers that have positive activation. Furthermore, there are a number of variants of the basic ReLU that introduce a gradient on the negative side of the domain, a commonly used variant being the leaky ReLU (Maas et al. 2013). Today, ReLUs (or variants of ReLUs) are the most frequently used neurons in deep learning research.

The Virtuous Cycle: Better Algorithms, Faster Hardware, Bigger Data

Although improved weight initialization methods and new activation functions have both contributed to the growth of deep learning, in recent years the two most important factors driving deep learning have been the speedup in computer power and the massive increase in dataset sizes. From a computational perspective, a major breakthrough for deep learning occurred in the late 2000s with the adoption of graphical processing units (GPUs) by the deep learning community to speed up training. A neural network can be understood as a sequence of matrix multiplications that are interspersed with the application of nonlinear activation functions, and GPUs are optimized for very fast matrix multiplication. Consequently, GPUs are ideal hardware to speed up neural network training, and their use has made a significant contribution to the development of the field. In 2004, Oh and Jung reported a twentyfold performance increase using a GPU

implementation of a neural network (Oh and Jung 2004), and the following year two further papers were published that demonstrated the potential of GPUs to speed up the training of neural networks: Steinkraus et al. (2005) used GPUs to train a two-layer neural network, and Chellapilla et al. (2006) used GPUs to train a CNN. However, at that time there were significant programming challenges to using GPUs for training networks (the training algorithm had to be implemented as a sequence of graphics operations), and so the initial adoption of GPUs by neural network researchers was relatively slow. These programming challenges were significantly reduced in 2007 when NVIDIA (a GPU manufacturer) released a C-like programming interface for GPUs called CUDA (compute unified device architecture).¹² CUDA was specifically designed to facilitate the use of GPUs for general computing tasks. In the years following the release of CUDA, the use of GPUs to speed up neural network training became standard.

However, even with these more powerful computer processors, deep learning would not have been possible unless massive datasets had also become available. The development of the internet and social media platforms, the proliferation of smartphones and “internet of things” sensors, has meant that the amount of data being captured has grown at an incredible rate over the last ten years. This has made it much easier for organizations to gather large datasets. This growth in data has been incredibly

important to deep learning because neural network models scale well with larger data (and in fact they can struggle with smaller datasets). It has also prompted organizations to consider how this data can be used to drive the development of new applications and innovations. This in turn has driven a need for new (more complex) computational models in order to deliver these new applications. And, the combination of large data and more complex algorithms requires faster hardware in order to make the necessary computational workload tractable. Figure 4.7 illustrates the virtuous cycle between big data, algorithmic breakthroughs (e.g., better weight initialization, ReLUs, etc.),

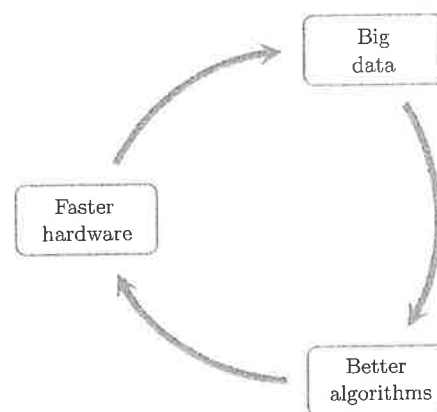


Figure 4.7 The virtuous cycle driving deep learning. Figure inspired by figure 1.2 in Reagen et al. 2017.

and improved hardware that is driving the deep learning revolution.

Summary

The history of deep learning reveals a number of underlying themes. There has been a shift from simple binary inputs to more complex continuous valued input. This trend toward more complex inputs is set to continue because deep learning models are most useful in high-dimensional domains, such as image processing and language. Images often have thousands of pixels in them, and language processing requires the ability represents and process hundreds of thousands of different words. This is why some of the best-known applications of deep learning are in these domains, for example, Facebook's face-recognition software, and Google's neural machine translation system. However, there are a growing number of new domains where large and complex digital datasets are being gathered. One area where deep learning has the potential to make a significant impact within the coming years is healthcare, and another complex domain is the sensor rich field of self-driving cars.

Somewhat surprisingly, at the core of these powerful models are simple information processing units: neurons. The connectionist idea that useful complex behavior can

emerge from the interactions between large numbers of simple processing units is still valid today. This emergent behavior arises through the sequences of layers in a network learning a hierarchical abstraction of increasingly complex features. This hierarchical abstraction is achieved by each neuron learning a simple transformation of the input it receives. The network as a whole then composes these sequences of smaller transformations in order to apply a complex (highly) nonlinear mapping to the input. The output from the model is then generated by the final output layers of neuron, based the learned representation generated through the hierarchical abstraction. This is why depth is such an important factor in neural networks: the deeper the network, the more powerful the model becomes in terms of its ability to learn complex nonlinear mappings. In many domains, the relationship between input data and desired outputs involves just such complex nonlinear mappings, and it is in these domains that deep learning models outdo other machine learning approaches.

An important design choice in creating a neural network is deciding which activation function to use within the neurons in a network. The activation function within each neuron in a network is how nonlinearity is introduced into the network, and as a result it is a necessary component if the network is to learn a nonlinear mapping from inputs to output. As networks have evolved, so too