



Charu C. Aggarwal

# Neural Networks and Deep Learning

A Textbook

*Second Edition*

MOREMEDIA



Springer

# Neural Networks and Deep Learning

Charu C. Aggarwal

# Neural Networks and Deep Learning

A Textbook

Second Edition

 Springer

Charu C. Aggarwal  
IBM T. J. Watson Research Center  
International Business Machines  
Yorktown Heights, NY, USA

ISBN 978-3-031-29641-3      ISBN 978-3-031-29642-0 (eBook)  
<https://doi.org/10.1007/978-3-031-29642-0>

© Springer Nature Switzerland AG 2018, 2023

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors, and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG  
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

To my wife Lata, my daughter Sayani,  
and my late parents Dr. Prem Sarup and Mrs. Pushplata Aggarwal.

---

---

# Preface

---

---

“Any A.I. smart enough to pass a Turing test is smart enough to know to fail it.”\_\*\*\*Ian McDonald

Neural networks were developed to simulate the human nervous system for machine learning tasks by treating the computational units in a learning model in a manner similar to human neurons. The grand vision of neural networks is to create artificial intelligence by building machines whose architecture simulates the computations in the human nervous system. Although the biological model of neural networks is an exciting one and evokes comparisons with science fiction, neural networks have a much simpler and mundane mathematical basis than a complex biological system. The neural network abstraction can be viewed as a modular approach of enabling learning algorithms that are based on continuous optimization on a computational graph of mathematical dependencies between the input and output. These ideas are strikingly similar to classical optimization methods in control theory, which historically preceded the development of neural network algorithms.

Neural networks were developed soon after the advent of computers in the fifties and sixties. Rosenblatt’s perceptron algorithm was seen as a fundamental cornerstone of neural networks, which caused an initial period of euphoria — it was soon followed by disappointment as the initial successes were somewhat limited. Eventually, at the turn of the century, greater data availability and increasing computational power lead to increased successes of neural networks, and this area was reborn under the new label of “deep learning.” Although we are still far from the day that artificial intelligence (AI) is close to human performance, there are specific domains like image recognition, self-driving cars, and game playing, where AI has matched or exceeded human performance. It is also hard to predict what AI might be able to do in the future. For example, few computer vision experts would have thought two decades ago that any automated system could ever perform an intuitive task like categorizing an image more accurately than a human. The large amounts of data available in recent years together with increased computational power have enabled experimentation with more sophisticated and deep neural architectures than was previously possible. The resulting success has changed the broader perception of the potential of deep learning. This book discusses neural networks from this modern perspective. The chapters of the book are organized as follows:

1. *The basics of neural networks*: Chapters 1, 2, and 3 discuss the basics of neural network design and the backpropagation algorithm. Many traditional machine learning models

can be understood as special cases of neural learning. Understanding the relationship between traditional machine learning and neural networks is the first step to understanding the latter. The simulation of various machine learning models with neural networks is provided in Chapter 3. This will give the analyst a feel of how neural networks push the envelope of traditional machine learning algorithms.

2. *Fundamentals of neural networks*: Although Chapters 1, 2, and 3 provide an overview of the training methods for neural networks, a more detailed understanding of the training challenges is provided in Chapters 4 and 5. Chapters 6 and 7 present radial-basis function (RBF) networks and restricted Boltzmann machines.
3. *Advanced topics in neural networks*: A lot of the recent success of deep learning is a result of the specialized architectures for various domains, such as recurrent neural networks and convolutional neural networks. Chapters 8 and 9 discuss recurrent and convolutional neural networks. Graph neural networks are discussed in Chapter 10. Several advanced topics like deep reinforcement learning, attention mechanisms, neural Turing mechanisms, and generative adversarial networks are discussed in Chapters 11 and 12.

We have included some “forgotten” architectures like RBF networks and Kohonen self-organizing maps because of their potential in many applications. An application-centric view is highlighted throughout the book in order to give the reader a feel for the technology.

## What Is New in The Second Edition

The second edition has focused on improving the presentations in the first edition and also on adding new material. Significant changes have been made to almost all the chapters to improve presentation and add new material where needed. In some cases, the material in different chapters has been reorganized and reordered in order to improve exposition. The discussion of training challenges with depth has been separated from the backpropagation chapter, so that both topics could be discussed in greater detail. Second-order methods have been explained with greater clarity and examples. Chapter 10 on graph neural networks is entirely new. The discussion on GRUs in Chapter 8 has been greatly enhanced. New convolutional architectures using attention, such as Squeeze-and-Excitation Networks, are discussed in Chapter 9. The chapter on reinforcement learning has been significantly reorganized in order to provide a clearer exposition. The Monte Carlo sampling approach for reinforcement learning is discussed in greater detail in its own dedicated section. Chapter 12 contains expanded discussions on attention mechanisms for graphs and computer vision, transformers, pre-trained language models, and adversarial learning.

## Notations

Throughout this book, a vector or a multidimensional data point is annotated with a bar, such as  $\bar{X}$  or  $\bar{y}$ . A vector or multidimensional point may be denoted by either small letters or capital letters, as long as it has a bar. Vector dot products are denoted by centered dots, such as  $\bar{X} \cdot \bar{Y}$ . A matrix is denoted in capital letters without a bar, such as  $R$ . Throughout the book, the  $n \times d$  matrix corresponding to the training data set is denoted by  $D$ , with  $n$  documents and  $d$  dimensions. The individual data points in  $D$  are therefore  $d$ -dimensional

row vectors. On the other hand, vectors with one component for each data point are usually  $n$ -dimensional column vectors. An example is the  $n$ -dimensional column vector  $\bar{y}$  of class variables. An observed value  $y_i$  is distinguished from a predicted value  $\hat{y}_i$  by a circumflex at the top of the variable.

Yorktown Heights, NY, USA

Charu C. Aggarwal



---

---

# Acknowledgments

---

---

## Acknowledgements for the First Edition

---

I would like to thank my family for their love and support during the busy time spent in writing this book. I would also like to thank my manager Nagui Halim for his support during the writing of this book.

Several figures in this book have been provided by the courtesy of various individuals and institutions. The Smithsonian Institution made the image of the Mark I perceptron (cf. Figure 1.5) available at no cost. Saket Sathe provided the outputs in Chapter 8 for the tiny Shakespeare data set, based on code available/described in [243, 604]. Andrew Zisserman provided Figures 9.13 and 9.17 in the section on convolutional visualizations. Another visualization of the feature maps in the convolution network (cf. Figure 9.16) was provided by Matthew Zeiler. NVIDIA provided Figure 11.10 on the convolutional neural network for self-driving cars in Chapter 11, and Sergey Levine provided the image on self-learning robots (cf. Figure 11.9) in the same chapter. Alec Radford provided Figure 12.11, which appears in Chapter 12. Alex Krizhevsky provided Figure 9.9(b) containing *AlexNet*.

This book has benefitted from significant feedback and several collaborations that I have had with numerous colleagues over the years. I would like to thank Quoc Le, Saket Sathe, Karthik Subbian, Jiliang Tang, and Suhang Wang for their feedback on various portions of this book. Shuai Zheng provided feedback on the section on regularized autoencoders in Chapter 5. I received feedback on the sections on autoencoders from Lei Cai and Hao Yuan. Feedback on the chapter on convolutional neural networks was provided by Hongyang Gao, Shuiwang Ji, and Zhengyang Wang. Shuiwang Ji, Lei Cai, Zhengyang Wang and Hao Yuan also reviewed the Chapters 4 and 8, and suggested several edits. They also suggested the ideas of using Figures 9.6 and 9.7 for elucidating the convolution/deconvolution operations.

For their collaborations, I would like to thank Tarek F. Abdelzaher, Jinghui Chen, Jing Gao, Quanquan Gu, Manish Gupta, Jiawei Han, Alexander Hinneburg, Thomas Huang, Nan Li, Huan Liu, Ruoming Jin, Daniel Keim, Arijit Khan, Latifur Khan, Mohammad M. Masud, Jian Pei, Magda Procopiuc, Guojun Qi, Chandan Reddy, Saket Sathe, Jaideep Srivastava, Karthik Subbian, Yizhou Sun, Jiliang Tang, Min-Hsuan Tsai, Haixun Wang, Jianyong Wang, Min Wang, Suhang Wang, Joel Wolf, Xifeng Yan, Mohammed Zaki, ChengXiang Zhai, and Peixiang Zhao. I would also like to thank my advisor James B. Orlin for his guidance during my early years as a researcher.

I would like to thank Lata Aggarwal for helping me with some of the figures created using PowerPoint graphics in this book. My daughter, Sayani, was helpful in incorporating special effects (e.g., image color, contrast, and blurring) in several JPEG images used at various places in this book.

## **Acknowledgements for the Second Edition**

---

The chapter on graph neural networks benefited from collaborations with Jiliang Tang, Lingfei Wu, and Suhang Wang. Shuiwang Ji, Meng Liu, and Hongyang Gao provided detailed comments on the chapter on graph neural networks as well as other parts of the book. Zhengyang Wang and Shuiwang Ji provided feedback on the new section on transformer networks. Yao Ma and Jiliang Tang provided the permission to use Figure 10.3. Yao Ma also provided detailed comments on the chapter on graph neural networks. Sharmishtha Dutta, a PhD student at RPI, proofread Chapter 10, and also pointed out the sections that needed further clarity. My manager, Horst Samulowitz, provided support during the writing of the second edition of this book.

---

---

# Contents

---

---

- 1 An Introduction to Neural Networks** **1**
- 1.1 Introduction 1
- 1.2 Single Computational Layer: The Perceptron 5
- 1.2.1 Use of Bias 8
- 1.2.2 What Objective Function Is the Perceptron Optimizing? 8
- 1.3 The Base Components of Neural Architectures 10
- 1.3.1 Choice of Activation Function 10
- 1.3.2 Softmax Activation Function 12
- 1.3.3 Common Loss Functions 13
- 1.4 Multilayer Neural Networks 13
- 1.4.1 The Multilayer Network as a Computational Graph 15
- 1.5 The Importance of Nonlinearity 17
- 1.5.1 Nonlinear Activations in Action 18
- 1.6 Advanced Architectures and Structured Data 20
- 1.7 Two Notable Benchmarks 21
- 1.7.1 The MNIST Database of Handwritten Digits 21
- 1.7.2 The ImageNet Database 22
- 1.8 Summary 23
- 1.9 Bibliographic Notes and Software Resources 23
- 1.10 Exercises 25
  
- 2 The Backpropagation Algorithm** **29**
- 2.1 Introduction 29
- 2.2 The Computational Graph Abstraction 30
- 2.2.1 Computational Graphs Create Complex Functions 31
- 2.3 Backpropagation in Computational Graphs 33
- 2.3.1 Computing Node-to-Node Derivatives with the Chain Rule 34
- 2.3.2 Dynamic Programming for Computing Node-to-Node Derivatives 38
- 2.3.3 Converting Node-to-Node Derivatives into Loss-to-Weight Derivatives 42

2.4	Backpropagation in Neural Networks . . . . .	44
2.4.1	Some Useful Derivatives of Activation Functions . . . . .	46
2.4.2	Examples of Updates for Various Activations . . . . .	48
2.5	The Vector-Centric View of Backpropagation . . . . .	50
2.5.1	Derivatives with Respect to Vectors . . . . .	51
2.5.2	Vector-Centric Chain Rule . . . . .	51
2.5.3	A Decoupled View of Vector-Centric Backpropagation . . . . .	52
2.5.4	Vector-Centric Backpropagation with Non-Layered Architectures . . . . .	57
2.6	The Not-So-Unimportant Details . . . . .	58
2.6.1	Mini-Batch Stochastic Gradient Descent . . . . .	58
2.6.2	Learning Rate Decay . . . . .	60
2.6.3	Checking the Correctness of Gradient Computation . . . . .	60
2.6.4	Regularization . . . . .	61
2.6.5	Loss Functions on Hidden Nodes . . . . .	61
2.6.6	Backpropagation Tricks for Handling Shared Weights . . . . .	62
2.7	Tuning and Preprocessing . . . . .	62
2.7.1	Tuning Hyperparameters . . . . .	63
2.7.2	Feature Preprocessing . . . . .	64
2.7.3	Initialization . . . . .	66
2.8	Backpropagation Is Interpretable . . . . .	67
2.9	Summary . . . . .	67
2.10	Bibliographic Notes and Software Resources . . . . .	68
2.11	Exercises . . . . .	68
<b>3</b>	<b>Machine Learning with Shallow Neural Networks</b>	<b>73</b>
3.1	Introduction . . . . .	73
3.2	Neural Architectures for Binary Classification Models . . . . .	75
3.2.1	Revisiting the Perceptron . . . . .	75
3.2.2	Least-Squares Regression . . . . .	76
3.2.2.1	Widrow-Hoff Learning . . . . .	78
3.2.2.2	Closed Form Solutions . . . . .	79
3.2.3	Support Vector Machines . . . . .	79
3.2.4	Logistic Regression . . . . .	81
3.2.5	Comparison of Different Models . . . . .	82
3.3	Neural Architectures for Multiclass Models . . . . .	84
3.3.1	Multiclass Perceptron . . . . .	84
3.3.2	Weston-Watkins SVM . . . . .	85
3.3.3	Multinomial Logistic Regression (Softmax Classifier) . . . . .	86
3.4	Unsupervised Learning with Autoencoders . . . . .	88
3.4.1	Linear Autoencoder with a Single Hidden Layer . . . . .	89
3.4.1.1	Connections with Singular Value Decomposition . . . . .	91
3.4.1.2	Sharing Weights in the Encoder and Decoder . . . . .	91
3.4.2	Nonlinear Activation Functions and Depth . . . . .	92
3.4.3	Application to Visualization . . . . .	93
3.4.4	Application to Outlier Detection . . . . .	95
3.4.5	Application to Multimodal Embeddings . . . . .	95
3.4.6	Benefits of Autoencoders . . . . .	96
3.5	Recommender Systems . . . . .	96

- 3.6 Text Embedding with Word2vec . . . . . 99
  - 3.6.1 Neural Embedding with Continuous Bag of Words . . . . . 100
  - 3.6.2 Neural Embedding with Skip-Gram Model . . . . . 103
  - 3.6.3 Word2vec (SGNS) is Logistic Matrix Factorization . . . . . 107
- 3.7 Simple Neural Architectures for Graph Embeddings . . . . . 110
  - 3.7.1 Handling Arbitrary Edge Counts . . . . . 111
  - 3.7.2 Beyond One-Hop Structural Models . . . . . 112
  - 3.7.3 Multinomial Model . . . . . 112
- 3.8 Summary . . . . . 113
- 3.9 Bibliographic Notes and Software Resources . . . . . 113
- 3.10 Exercises . . . . . 114
- 4 Deep Learning: Principles and Training Algorithms . . . . . 119**
  - 4.1 Introduction . . . . . 119
  - 4.2 Why Is Depth Beneficial? . . . . . 120
    - 4.2.1 Hierarchical Feature Engineering: How Depth Reveals Rich Structure . . . . . 120
  - 4.3 Why Is Training Deep Networks Hard? . . . . . 122
    - 4.3.1 Geometric Understanding of the Effect of Gradient Ratios . . . . . 122
    - 4.3.2 The Vanishing and Exploding Gradient Problems . . . . . 124
    - 4.3.3 Cliffs and Valleys . . . . . 126
    - 4.3.4 Convergence Problems with Depth . . . . . 127
    - 4.3.5 Local Minima . . . . . 127
  - 4.4 Depth-Friendly Neural Architectures . . . . . 129
    - 4.4.1 Activation Function Choice . . . . . 129
    - 4.4.2 Dying Neurons and “Brain Damage” . . . . . 130
      - 4.4.2.1 Leaky ReLU . . . . . 130
      - 4.4.2.2 Maxout Networks . . . . . 131
    - 4.4.3 Using Skip Connections . . . . . 131
  - 4.5 Depth-Friendly Gradient-Descent Strategies . . . . . 132
    - 4.5.1 Importance of Preprocessing and Initialization . . . . . 132
    - 4.5.2 Momentum-Based Learning . . . . . 133
    - 4.5.3 Nesterov Momentum . . . . . 134
    - 4.5.4 Parameter-Specific Learning Rates . . . . . 135
      - 4.5.4.1 AdaGrad . . . . . 136
      - 4.5.4.2 RMSProp . . . . . 136
      - 4.5.4.3 AdaDelta . . . . . 137
    - 4.5.5 Combining Parameter-Specific Learning and Momentum . . . . . 138
      - 4.5.5.1 RMSProp with Nesterov Momentum . . . . . 138
      - 4.5.5.2 Adam . . . . . 138
    - 4.5.6 Gradient Clipping . . . . . 139
    - 4.5.7 Polyak Averaging . . . . . 139
  - 4.6 Second-Order Derivatives: The Newton Method . . . . . 140
    - 4.6.1 Example: Newton Method in the Quadratic Bowl . . . . . 142
    - 4.6.2 Example: Newton Method in a Non-Quadratic Function . . . . . 142
    - 4.6.3 The Saddle-Point Problem with Second-Order Methods . . . . . 143
  - 4.7 Fast Approximations of Newton Method . . . . . 145
    - 4.7.1 Conjugate Gradient Method . . . . . 145
    - 4.7.2 Quasi-Newton Methods and BFGS . . . . . 148

4.8	Batch Normalization . . . . .	150
4.9	Practical Tricks for Acceleration and Compression . . . . .	153
4.9.1	GPU Acceleration . . . . .	154
4.9.2	Parallel and Distributed Implementations . . . . .	156
4.9.3	Algorithmic Tricks for Model Compression . . . . .	157
4.10	Summary . . . . .	160
4.11	Bibliographic Notes and Software Resources . . . . .	160
4.12	Exercises . . . . .	162
<b>5</b>	<b>Teaching Deep Learners to Generalize</b>	<b>165</b>
5.1	Introduction . . . . .	165
5.1.1	Example: Linear Regression . . . . .	166
5.1.2	Example: Polynomial Regression . . . . .	167
5.2	The Bias-Variance Trade-Off . . . . .	171
5.3	Generalization Issues in Model Tuning and Evaluation . . . . .	174
5.3.1	Evaluating with Hold-Out and Cross-Validation . . . . .	176
5.3.2	Issues with Training at Scale . . . . .	177
5.3.3	How to Detect Need to Collect More Data . . . . .	178
5.4	Penalty-Based Regularization . . . . .	178
5.4.1	Connections with Noise Injection . . . . .	179
5.4.2	$L_1$ -Regularization . . . . .	180
5.4.3	$L_1$ - or $L_2$ -Regularization? . . . . .	181
5.4.4	Penalizing Hidden Units: Learning Sparse Representations . . . . .	181
5.5	Ensemble Methods . . . . .	182
5.5.1	Bagging and Subsampling . . . . .	182
5.5.2	Parametric Model Selection and Averaging . . . . .	184
5.5.3	Randomized Connection Dropping . . . . .	184
5.5.4	Dropout . . . . .	185
5.5.5	Data Perturbation Ensembles . . . . .	187
5.6	Early Stopping . . . . .	188
5.6.1	Understanding Early Stopping from the Variance Perspective . . . . .	189
5.7	Unsupervised Pretraining . . . . .	189
5.7.1	Variations of Unsupervised Pretraining . . . . .	192
5.7.2	What About Supervised Pretraining? . . . . .	193
5.8	Continuation and Curriculum Learning . . . . .	194
5.9	Parameter Sharing . . . . .	196
5.10	Regularization in Unsupervised Applications . . . . .	197
5.10.1	When the Hidden Layer is Broader than the Input Layer . . . . .	197
5.10.1.1	Sparse Feature Learning . . . . .	198
5.10.2	Noise Injection: De-noising Autoencoders . . . . .	198
5.10.3	Gradient-Based Penalization: Contractive Autoencoders . . . . .	199
5.10.4	Hidden Probabilistic Structure: Variational Autoencoders . . . . .	203
5.10.4.1	Reconstruction and Generative Sampling . . . . .	206
5.10.4.2	Conditional Variational Autoencoders . . . . .	208
5.10.4.3	Relationship with Generative Adversarial Networks . . . . .	208
5.11	Summary . . . . .	209
5.12	Bibliographic Notes and Software Resources . . . . .	210
5.13	Exercises . . . . .	211

<b>6</b>	<b>Radial Basis Function Networks</b>	<b>215</b>
6.1	Introduction	215
6.2	Training an RBF Network	218
6.2.1	Training the Hidden Layer	218
6.2.2	Training the Output Layer	220
6.2.3	Iterative Construction of Hidden Layer	221
6.2.4	Fully Supervised Learning of Hidden Layer	222
6.3	Variations and Special Cases of RBF Networks	223
6.3.1	Classification with Perceptron Criterion	224
6.3.2	Classification with Hinge Loss	224
6.3.3	Example of Linear Separability Promoted by RBF	224
6.3.4	Application to Interpolation	226
6.4	Relationship with Kernel Methods	227
6.4.1	Kernel Regression Is a Special Case of RBF Networks	227
6.4.2	Kernel SVM Is a Special Case of RBF Networks	228
6.5	Summary	229
6.6	Bibliographic Notes and Software Resources	229
6.7	Exercises	229
<b>7</b>	<b>Restricted Boltzmann Machines</b>	<b>231</b>
7.1	Introduction	231
7.2	Hopfield Networks	232
7.2.1	Training a Hopfield Network	235
7.2.2	Building a Toy Recommender and Its Limitations	236
7.2.3	Increasing the Expressive Power of the Hopfield Network	237
7.3	The Boltzmann Machine	238
7.3.1	How a Boltzmann Machine Generates Data	240
7.3.2	Learning the Weights of a Boltzmann Machine	240
7.4	Restricted Boltzmann Machines	242
7.4.1	Training the RBM	244
7.4.2	Contrastive Divergence Algorithm	245
7.5	Applications of Restricted Boltzmann Machines	247
7.5.1	Dimensionality Reduction and Data Reconstruction	247
7.5.2	RBM for Collaborative Filtering	249
7.5.3	Using RBMs for Classification	252
7.5.4	Topic Models with RBMs	254
7.5.5	RBM for Machine Learning with Multimodal Data	256
7.6	Using RBMs beyond Binary Data Types	258
7.7	Stacking Restricted Boltzmann Machines	258
7.7.1	Unsupervised Learning	261
7.7.2	Supervised Learning	261
7.7.3	Deep Boltzmann Machines and Deep Belief Networks	261
7.8	Summary	262
7.9	Bibliographic Notes and Software Resources	262
7.10	Exercises	264

<b>8</b>	<b>Recurrent Neural Networks</b>	<b>265</b>
8.1	Introduction	265
8.2	The Architecture of Recurrent Neural Networks	267
8.2.1	Language Modeling Example of RNN	270
8.2.2	Backpropagation Through Time	273
8.2.3	Bidirectional Recurrent Networks	275
8.2.4	Multilayer Recurrent Networks	277
8.3	The Challenges of Training Recurrent Networks	278
8.3.1	Layer Normalization	281
8.4	Echo-State Networks	282
8.5	Long Short-Term Memory (LSTM)	285
8.6	Gated Recurrent Units (GRUs)	287
8.7	Applications of Recurrent Neural Networks	289
8.7.1	Contextualized Word Embeddings with ELMo	290
8.7.2	Application to Automatic Image Captioning	291
8.7.3	Sequence-to-Sequence Learning and Machine Translation	292
8.7.4	Application to Sentence-Level Classification	295
8.7.5	Token-Level Classification with Linguistic Features	296
8.7.6	Time-Series Forecasting and Prediction	297
8.7.7	Temporal Recommender Systems	299
8.7.8	Secondary Protein Structure Prediction	301
8.7.9	End-to-End Speech Recognition	301
8.7.10	Handwriting Recognition	301
8.8	Summary	302
8.9	Bibliographic Notes and Software Resources	302
8.10	Exercises	303
<b>9</b>	<b>Convolutional Neural Networks</b>	<b>305</b>
9.1	Introduction	305
9.1.1	Historical Perspective and Biological Inspiration	305
9.1.2	Broader Observations about Convolutional Neural Networks	306
9.2	The Basic Structure of a Convolutional Network	307
9.2.1	Padding	312
9.2.2	Strides	313
9.2.3	The ReLU Layer	315
9.2.4	Pooling	315
9.2.5	Fully Connected Layers	317
9.2.6	The Interleaving between Layers	317
9.2.7	Hierarchical Feature Engineering	320
9.3	Training a Convolutional Network	321
9.3.1	Backpropagating Through Convolutions	321
9.3.2	Backpropagation as Convolution with Inverted/Transposed Filter	322
9.3.3	Convolution/Backpropagation as Matrix Multiplications	324
9.3.4	Data Augmentation	326
9.4	Case Studies of Convolutional Architectures	326
9.4.1	AlexNet	327
9.4.2	ZFNet	329
9.4.3	VGG	330



9.4.4	GoogLeNet . . . . .	333
9.4.5	ResNet . . . . .	335
9.4.6	Squeeze-and-Excitation Networks (SENet) . . . . .	338
9.4.7	The Effects of Depth . . . . .	339
9.4.8	Pretrained Models . . . . .	340
9.5	Visualization and Unsupervised Learning . . . . .	341
9.5.1	Visualizing the Features of a Trained Network . . . . .	341
9.5.2	Convolutional Autoencoders . . . . .	347
9.6	Applications of Convolutional Networks . . . . .	351
9.6.1	Content-Based Image Retrieval . . . . .	352
9.6.2	Object Localization . . . . .	352
9.6.3	Object Detection . . . . .	354
9.6.4	Natural Language and Sequence Learning with TextCNN . . . . .	355
9.6.5	Video Classification . . . . .	355
9.7	Summary . . . . .	356
9.8	Bibliographic Notes and Software Resources . . . . .	356
9.9	Exercises . . . . .	359
<b>10</b>	<b>Graph Neural Networks</b> . . . . .	<b>361</b>
10.1	Introduction . . . . .	361
10.2	Node Embeddings with Conventional Architectures . . . . .	362
10.2.1	Adjacency Matrix Representation and Feature Engineering . . . . .	364
10.3	Graph Neural Networks: The General Framework . . . . .	364
10.3.1	The Neighborhood Function . . . . .	368
10.3.2	Graph Convolution Function . . . . .	368
10.3.3	GraphSAGE . . . . .	369
10.3.4	Handling Edge Weights . . . . .	371
10.3.5	Handling New Vertices . . . . .	371
10.3.6	Handling Relational Networks . . . . .	372
10.3.7	Directed Graphs . . . . .	373
10.3.8	Gated Graph Neural Networks . . . . .	373
10.3.9	Comparison with Image Convolutional Networks . . . . .	374
10.4	Backpropagation in Graph Neural Networks . . . . .	375
10.5	Beyond Nodes: Generating Graph-Level Models . . . . .	377
10.6	Applications of Graph Neural Networks . . . . .	382
10.7	Summary . . . . .	384
10.8	Bibliographic Notes and Software Resources . . . . .	384
10.9	Exercises . . . . .	385
<b>11</b>	<b>Deep Reinforcement Learning</b> . . . . .	<b>389</b>
11.1	Introduction . . . . .	389
11.2	Stateless Algorithms: Multi-Armed Bandits . . . . .	391
11.3	The Basic Framework of Reinforcement Learning . . . . .	393
11.4	Monte Carlo Sampling . . . . .	395
11.4.1	Monte Carlo Sampling Algorithm . . . . .	395
11.4.2	Monte Carlo Rollouts with Function Approximators . . . . .	396

11.5	Bootstrapping for Value Function Learning . . . . .	398
11.5.1	Q-Learning . . . . .	399
11.5.2	Deep Learning Models as Function Approximators . . . . .	400
11.5.3	Example: Neural Network Specifics for Video Game Setting . . . . .	403
11.5.4	On-Policy versus Off-Policy Methods: SARSA . . . . .	404
11.5.5	Modeling States versus State-Action Pairs . . . . .	405
11.6	Policy Gradient Methods . . . . .	407
11.6.1	Finite Difference Methods . . . . .	408
11.6.2	Likelihood Ratio Methods . . . . .	409
11.6.3	Actor-Critic Methods . . . . .	411
11.6.4	Continuous Action Spaces . . . . .	413
11.7	Monte Carlo Tree Search . . . . .	413
11.8	Case Studies . . . . .	415
11.8.1	AlphaGo and AlphaZero for Go and Chess . . . . .	415
11.8.2	Self-Learning Robots . . . . .	420
11.8.2.1	Deep Learning of Locomotion Skills . . . . .	420
11.8.2.2	Deep Learning of Visuomotor Skills . . . . .	422
11.8.3	Building Conversational Systems: Deep Learning for Chatbots . . . . .	423
11.8.4	Self-Driving Cars . . . . .	425
11.8.5	Neural Architecture Search with Reinforcement Learning . . . . .	428
11.9	Practical Challenges Associated with Safety . . . . .	429
11.10	Summary . . . . .	429
11.11	Bibliographic Notes and Software Resources . . . . .	430
11.12	Exercises . . . . .	432
<b>12</b>	<b>Advanced Topics in Deep Learning</b> . . . . .	<b>435</b>
12.1	Introduction . . . . .	435
12.2	Attention Mechanisms . . . . .	436
12.2.1	Recurrent Models of Visual Attention . . . . .	437
12.2.2	Attention Mechanisms for Image Captioning . . . . .	439
12.2.3	Soft Image Attention with Spatial Transformer . . . . .	440
12.2.4	Attention Mechanisms for Machine Translation . . . . .	442
12.2.5	Transformer Networks . . . . .	446
12.2.5.1	How Self Attention Helps . . . . .	446
12.2.5.2	The Self-Attention Module . . . . .	447
12.2.5.3	Incorporating Positional Information . . . . .	449
12.2.5.4	The Sequence-to-Sequence Transformer . . . . .	450
12.2.5.5	Multihead Attention . . . . .	450
12.2.6	Transformer-Based Pre-trained Language Models . . . . .	451
12.2.6.1	GPT-n . . . . .	452
12.2.6.2	BERT . . . . .	454
12.2.6.3	T5 . . . . .	455
12.2.7	Vision Transformer (ViT) . . . . .	457
12.2.8	Attention Mechanisms in Graphs . . . . .	458
12.3	Neural Turing Machines . . . . .	459
12.4	Adversarial Deep Learning . . . . .	463
12.5	Generative Adversarial Networks (GANs) . . . . .	467
12.5.1	Training a Generative Adversarial Network . . . . .	468
12.5.2	Comparison with Variational Autoencoder . . . . .	470

- 12.5.3 Using GANs for Generating Image Data . . . . . 470
- 12.5.4 Conditional Generative Adversarial Networks . . . . . 471
- 12.6 Competitive Learning . . . . . 476
  - 12.6.1 Vector Quantization . . . . . 477
  - 12.6.2 Kohonen Self-Organizing Map . . . . . 478
- 12.7 Limitations of Neural Networks . . . . . 480
  - 12.7.1 An Aspirational Goal: Few Shot Learning . . . . . 481
  - 12.7.2 An Aspirational Goal: Energy-Efficient Learning . . . . . 482
- 12.8 Summary . . . . . 483
- 12.9 Bibliographic Notes and Software Resources . . . . . 483
- 12.10 Exercises . . . . . 485
  
- Bibliography** . . . . . **487**
  
- Index** . . . . . **525**

---

---

# Author Biography

---

---

**Charu C. Aggarwal** is a Distinguished Research Staff Member (DRSM) at the IBM T. J. Watson Research Center in Yorktown Heights, New York. He completed his undergraduate degree in Computer Science from the Indian Institute of Technology at Kanpur in 1993 and his Ph.D. from the Massachusetts Institute of Technology in 1996.



He has worked extensively in the field of data mining. He has published more than 400 papers in refereed conferences and journals and authored over 80 patents. He is the author or editor of 20 books, including textbooks on data mining, recommender systems, and outlier analysis. Because of the commercial value of his patents, he has thrice been designated a Master Inventor at IBM. He is a recipient of an IBM Corporate Award (2003) for his work on bio-terrorist threat detection in data streams, a recipient of the IBM Outstanding Innovation Award (2008) for his scientific contributions to privacy technology, and a recipient of two IBM Outstanding Technical Achievement Awards (2009,

2015) for his work on data streams/high-dimensional data. He received the EDBT 2014 Test of Time Award for his work on condensation-based privacy-preserving data mining. He is a recipient of the IEEE ICDM Research Contributions Award (2015) and ACM SIGKDD Innovation Award, which are the two most prestigious awards for influential research contributions in the field of data mining. He is also a recipient of the W. Wallace McDowell Award, which is the highest award given solely by the IEEE Computer Society across the field of Computer Science.

He has served as the general co-chair of the IEEE Big Data Conference (2014) and as the program co-chair of the ACM CIKM Conference (2015), the IEEE ICDM Conference (2015), and the ACM KDD Conference (2016). He served as an associate editor of the IEEE Transactions on Knowledge and Data Engineering from 2004 to 2008. He is an associate editor of the IEEE Transactions on Big Data, an action editor of the Data Mining and Knowledge Discovery Journal, and an associate editor of the Knowledge and Information Systems Journal. He has served or currently serves as the editor-in-chief of the ACM Transactions on Knowledge Discovery from Data as well as the ACM SIGKDD Explorations. He is also an editor-in-chief of ACM Books. He serves on the advisory board of the Lecture Notes on Social Networks, a publication by Springer. He has served as the vice-president of

the SIAM Activity Group on Data Mining and is a member of the SIAM industry committee. He received the ACM SIGKDD Service Award for the aforementioned contributions to running conferences and journals— this honor is the most prestigious award for services to the field of data mining. He is a fellow of the SIAM, ACM, and the IEEE, for “contributions to knowledge discovery and data mining algorithms.”