

# Table of Contents

|   |           |
|---|-----------|
| <b>Preface.....</b>                                       | <b>ix</b> |
| <b>1. The Neural Network.....</b>                         | <b>1</b>  |
| Building Intelligent Machines                             | 1         |
| The Limits of Traditional Computer Programs               | 2         |
| The Mechanics of Machine Learning                         | 3         |
| The Neuron  | 7         |
| Expressing Linear Perceptrons as Neurons                  | 8         |
| Feed-Forward Neural Networks                              | 9         |
| Linear Neurons and Their Limitations                      | 12        |
| Sigmoid, Tanh, and ReLU Neurons                           | 13        |
| Softmax Output Layers                                     | 15        |
| Looking Forward   | 15        |
| <b>2. Training Feed-Forward Neural Networks.....</b>      | <b>17</b> |
| The Fast-Food Problem                                     | 17        |
| Gradient Descent  | 19        |
| The Delta Rule and Learning Rates                         | 21        |
| Gradient Descent with Sigmoidal Neurons                   | 22        |
| The Backpropagation Algorithm                             | 23        |
| Stochastic and Minibatch Gradient Descent                 | 25        |
| Test Sets, Validation Sets, and Overfitting               | 27        |
| Preventing Overfitting in Deep Neural Networks            | 34        |
| Summary   | 37        |
| <b>3. Implementing Neural Networks in TensorFlow.....</b> | <b>39</b> |
| What Is TensorFlow?                                       | 39        |
| How Does TensorFlow Compare to Alternatives?              | 40        |



|   |           |
|---|-----------|
| Installing TensorFlow   | 41        |
| Creating and Manipulating TensorFlow Variables                      | 43        |
| TensorFlow Operations   | 45        |
| Placeholder Tensors   | 45        |
| Sessions in TensorFlow  | 46        |
| Navigating Variable Scopes and Sharing Variables                    | 48        |
| Managing Models over the CPU and GPU                                | 51        |
| Specifying the Logistic Regression Model in TensorFlow              | 52        |
| Logging and Training the Logistic Regression Model                  | 55        |
| Leveraging TensorBoard to Visualize Computation Graphs and Learning | 58        |
| Building a Multilayer Model for MNIST in TensorFlow                 | 59        |
| Summary   | 62        |
| <b>4. Beyond Gradient Descent.....</b>                              | <b>63</b> |
| The Challenges with Gradient Descent                                | 63        |
| Local Minima in the Error Surfaces of Deep Networks                 | 64        |
| Model Identifiability   | 65        |
| How Pesky Are Spurious Local Minima in Deep Networks?               | 66        |
| Flat Regions in the Error Surface                                   | 69        |
| When the Gradient Points in the Wrong Direction                     | 71        |
| Momentum-Based Optimization   | 74        |
| A Brief View of Second-Order Methods                                | 77        |
| Learning Rate Adaptation  | 78        |
| AdaGrad—Accumulating Historical Gradients                           | 79        |
| RMSProp—Exponentially Weighted Moving Average of Gradients          | 80        |
| Adam—Combining Momentum and RMSProp                                 | 81        |
| The Philosophy Behind Optimizer Selection                           | 83        |
| Summary   | 83        |
| <b>5. Convolutional Neural Networks.....</b>                        | <b>85</b> |
| Neurons in Human Vision   | 85        |
| The Shortcomings of Feature Selection                               | 86        |
| Vanilla Deep Neural Networks Don't Scale                            | 89        |
| Filters and Feature Maps  | 90        |
| Full Description of the Convolutional Layer                         | 95        |
| Max Pooling   | 98        |
| Full Architectural Description of Convolution Networks              | 99        |
| Closing the Loop on MNIST with Convolutional Networks               | 101       |
| Image Preprocessing Pipelines Enable More Robust Models             | 103       |
| Accelerating Training with Batch Normalization                      | 104       |
| Building a Convolutional Network for CIFAR-10                       | 107       |
| Visualizing Learning in Convolutional Networks                      | 109       |



|   |            |
|---|------------|
| Leveraging Convolutional Filters to Replicate Artistic Styles | 113        |
| Learning Convolutional Filters for Other Problem Domains      | 114        |
| Summary   | 115        |
| <b>6. Embedding and Representation Learning.....</b>          | <b>117</b> |
| Learning Lower-Dimensional Representations                    | 117        |
| Principal Component Analysis                                  | 118        |
| Motivating the Autoencoder Architecture                       | 120        |
| Implementing an Autoencoder in TensorFlow                     | 121        |
| Denoising to Force Robust Representations                     | 134        |
| Sparsity in Autoencoders                                      | 137        |
| When Context Is More Informative than the Input Vector        | 140        |
| The Word2Vec Framework  | 143        |
| Implementing the Skip-Gram Architecture                       | 146        |
| Summary   | 152        |
| <b>7. Models for Sequence Analysis.....</b>                   | <b>153</b> |
| Analyzing Variable-Length Inputs                              | 153        |
| Tackling seq2seq with Neural N-Grams                          | 155        |
| Implementing a Part-of-Speech Tagger                          | 156        |
| Dependency Parsing and SyntaxNet                              | 164        |
| Beam Search and Global Normalization                          | 168        |
| A Case for Stateful Deep Learning Models                      | 172        |
| Recurrent Neural Networks                                     | 173        |
| The Challenges with Vanishing Gradients                       | 176        |
| Long Short-Term Memory (LSTM) Units                           | 178        |
| TensorFlow Primitives for RNN Models                          | 183        |
| Implementing a Sentiment Analysis Model                       | 185        |
| Solving seq2seq Tasks with Recurrent Neural Networks          | 189        |
| Augmenting Recurrent Networks with Attention                  | 191        |
| Dissecting a Neural Translation Network                       | 194        |
| Summary   | 217        |
| <b>8. Memory Augmented Neural Networks.....</b>               | <b>219</b> |
| Neural Turing Machines  | 219        |
| Attention-Based Memory Access                                 | 221        |
| NTM Memory Addressing Mechanisms                              | 223        |
| Differentiable Neural Computers                               | 226        |
| Interference-Free Writing in DNCs                             | 229        |
| DNC Memory Reuse  | 230        |
| Temporal Linking of DNC Writes                                | 231        |
| Understanding the DNC Read Head                               | 232        |



|   |            |
|---|------------|
| The DNC Controller Network                      | 232        |
| Visualizing the DNC in Action                   | 234        |
| Implementing the DNC in TensorFlow              | 237        |
| Teaching a DNC to Read and Comprehend           | 242        |
| Summary   | 244        |
| <b>9. Deep Reinforcement Learning</b>           | <b>245</b> |
| Deep Reinforcement Learning Masters Atari Games | 245        |
| What Is Reinforcement Learning?                 | 247        |
| Markov Decision Processes (MDP)                 | 248        |
| Policy  | 249        |
| Future Return                                   | 250        |
| Discounted Future Return                        | 251        |
| Explore Versus Exploit                          | 251        |
| Policy Versus Value Learning                    | 253        |
| Policy Learning via Policy Gradients            | 254        |
| Pole-Cart with Policy Gradients                 | 254        |
| OpenAI Gym                                      | 254        |
| Creating an Agent                               | 255        |
| Building the Model and Optimizer                | 257        |
| Sampling Actions                                | 257        |
| Keeping Track of History                        | 257        |
| Policy Gradient Main Function                   | 258        |
| PGAgent Performance on Pole-Cart                | 260        |
| Q-Learning and Deep Q-Networks                  | 261        |
| The Bellman Equation                            | 261        |
| Issues with Value Iteration                     | 262        |
| Approximating the Q-Function                    | 262        |
| Deep Q-Network (DQN)                            | 263        |
| Training DQN                                    | 263        |
| Learning Stability                              | 263        |
| Target Q-Network                                | 264        |
| Experience Replay                               | 264        |
| From Q-Function to Policy                       | 264        |
| DQN and the Markov Assumption                   | 265        |
| DQN's Solution to the Markov Assumption         | 265        |
| Playing Breakout with DQN                       | 265        |
| Building Our Architecture                       | 268        |
| Stacking Frames                                 | 268        |
| Setting Up Training Operations                  | 268        |
| Updating Our Target Q-Network                   | 269        |
| Implementing Experience Replay                  | 269        |



|  |            |
|--|------------|
| DQN Main Loop  | 270        |
| DQNAgent Results on Breakout                               | 272        |
| Improving and Moving Beyond DQN                            | 273        |
| Deep Recurrent Q-Networks (DRQN)                           | 273        |
| Asynchronous Advantage Actor-Critic Agent (A3C)            | 274        |
| UNsupervised REinforcement and Auxiliary Learning (UNREAL) | 275        |
| Summary  | 276        |
| <b>Index.....</b>  | <b>277</b> |

With the reinvigoration of neural networks in the 2000s, deep learning has become an extremely active area of research that is paving the way for modern machine learning. This book uses exposition and examples to help you understand major concepts in this complicated field. Large companies such as Google, Microsoft, and Facebook have taken notice and are actively growing in-house deep learning teams. For the rest of us, deep learning is still a pretty complex and difficult subject to grasp. Research papers are filled to the brim with jargon, and scattered online tutorials do little to help build a strong intuition for why and how deep learning practitioners approach problems. Our goal is to bridge this gap.

## Prerequisites and Objectives

This book is aimed at an audience with a basic operating understanding of calculus, matrices, and Python programming. Approaching this material without this background is possible, but likely to be more challenging. Background in linear algebra may also be helpful in navigating certain sections of mathematical exposition.

By the end of the book, we hope that our readers will be left with an intuition for how to approach problems using deep learning, the historical context for modern deep learning approaches, and a familiarity with implementing deep learning algorithms using the TensorFlow open source library.

## Conventions Used in This Book

The following typographical conventions are used in this book:

### *Italic*

Indicates new terms, URLs, email addresses, filenames, and file extensions.

### **Constant width**

Used for program listings, as well as within paragraphs to refer to program elements such as variable or function names, databases, data types, environment variables, statements, and keywords.