

contents

<i>preface</i>	<i>xv</i>
<i>acknowledgments</i>	<i>xvii</i>
<i>about this book</i>	<i>xviii</i>
<i>about the authors</i>	<i>xxi</i>
<i>about the cover illustration</i>	<i>xxii</i>

1 *What is deep learning?* **1**

- 1.1 Artificial intelligence, machine learning, and deep learning 2
- 1.2 Artificial intelligence 2
- 1.3 Machine learning 3
- 1.4 Learning rules and representations from data 4
- 1.5 The “deep” in “deep learning” 7
- 1.6 Understanding how deep learning works, in three figures 8
- 1.7 What makes deep learning different 10
- 1.8 The age of generative AI 11
- 1.9 What deep learning has achieved so far 11
- 1.10 Beware of the short-term hype 12
- 1.11 Summer can turn to winter 14
- 1.12 The promise of AI 14

2	<i>The mathematical building blocks of neural networks</i>	16
2.1	A first look at a neural network	17
2.2	Data representations for neural networks	21
	<i>Scalars (rank-0 tensors)</i>	22
	<i>Vectors (rank-1 tensors)</i>	22
	<i>Matrices (rank-2 tensors)</i>	22
	<i>Rank-3 tensors and higher-rank tensors</i>	23
	<i>Key attributes</i>	23
	<i>Manipulating tensors in NumPy</i>	25
	<i>The notion of data batches</i>	25
	<i>Real-world examples of data tensors</i>	26
2.3	The gears of neural networks: Tensor operations	28
	<i>Element-wise operations</i>	29
	<i>Broadcasting</i>	30
	<i>Tensor product</i>	32
	<i>Tensor reshaping</i>	34
	<i>Geometric interpretation of tensor operations</i>	35
	<i>A geometric interpretation of deep learning</i>	38
2.4	The engine of neural networks: Gradient-based optimization	39
	<i>What's a derivative?</i>	41
	<i>Derivative of a tensor operation: The gradient</i>	42
	<i>Stochastic gradient descent</i>	43
	<i>Chaining derivatives: The Backpropagation algorithm</i>	46
2.5	Looking back at our first example	51
	<i>Reimplementing our first example from scratch</i>	53
	<i>Running one training step</i>	55
	<i>The full training loop</i>	57
	<i>Evaluating the model</i>	58
3	<i>Introduction to TensorFlow, PyTorch, JAX, and Keras</i>	60
3.1	A brief history of deep learning frameworks	61
3.2	How these frameworks relate to each other	63
3.3	Introduction to TensorFlow	63
	<i>First steps with TensorFlow</i>	64
	<i>An end-to-end example: A linear classifier in pure TensorFlow</i>	69
	<i>What makes the TensorFlow approach unique</i>	74
3.4	Introduction to PyTorch	74
	<i>First steps with PyTorch</i>	75
	<i>An end-to-end example: A linear classifier in pure PyTorch</i>	78
	<i>What makes the PyTorch approach unique</i>	81
3.5	Introduction to JAX	82
	<i>First steps with JAX</i>	82
	<i>Tensors in JAX</i>	83
	<i>Random number generation in JAX</i>	83
	<i>An end-to-end example: A linear classifier in pure JAX</i>	88
	<i>What makes the JAX approach unique</i>	90

- 3.6 Introduction to Keras 90
First steps with Keras 91 ▪ *Layers: The building blocks of deep learning* 92 ▪ *From layers to models* 96 ▪ *The “compile” step: Configuring the learning process* 97 ▪ *Picking a loss function* 99 ▪ *Understanding the fit method* 100 ▪ *Monitoring loss and metrics on validation data* 101 ▪ *Inference: Using a model after training* 102

4 Classification and regression 104

- 4.1 Classifying movie reviews: A binary classification example 106
The IMDb dataset 106 ▪ *Preparing the data* 107 ▪ *Building your model* 108 ▪ *Validating your approach* 111 ▪ *Using a trained model to generate predictions on new data* 115 ▪ *Further experiments* 115 ▪ *Wrapping up* 116
- 4.2 Classifying newswires: A multiclass classification example 116
The Reuters dataset 116 ▪ *Preparing the data* 118 ▪ *Building your model* 118 ▪ *Validating your approach* 120 ▪ *Generating predictions on new data* 124 ▪ *A different way to handle the labels and the loss* 124 ▪ *The importance of having sufficiently large intermediate layers* 125 ▪ *Further experiments* 125 ▪ *Wrapping up* 126
- 4.3 Predicting house prices: A regression example 126
The California Housing Price dataset 126 ▪ *Preparing the data* 128 ▪ *Building your model* 128 ▪ *Validating your approach using K-fold validation* 129 ▪ *Generating predictions on new data* 134 ▪ *Wrapping up* 134

5 Fundamentals of machine learning 136

- 5.1 Generalization: The goal of machine learning 136
Underfitting and overfitting 137 ▪ *The nature of generalization in deep learning* 143
- 5.2 Evaluating machine-learning models 149
Training, validation, and test sets 149 ▪ *Beating a common-sense baseline* 152 ▪ *Things to keep in mind about model evaluation* 152
- 5.3 Improving model fit 153
Tuning key gradient descent parameters 153 ▪ *Using better architecture priors* 155 ▪ *Increasing model capacity* 155

- 5.4 Improving generalization 158
Dataset curation 159 • Feature engineering 159 • Using early stopping 161 • Regularizing your model 161
- 6 The universal workflow of machine learning 171**
- 6.1 Defining the task 172
*Framing the problem 172 • Collecting a dataset 174
Understanding your data 178 • Choosing a measure of success 178*
- 6.2 Developing a model 179
*Preparing the data 179 • Choosing an evaluation protocol 180
Beating a baseline 181 • Scaling up: Developing a model that overfits 182 • Regularizing and tuning your model 183*
- 6.3 Deploying your model 183
*Explaining your work to stakeholders and setting expectations 184
Shipping an inference model 184 • Monitoring your model in the wild 188 • Maintaining your model 188*
- 7 A deep dive on Keras 190**
- 7.1 A spectrum of workflows 191
- 7.2 Different ways to build Keras models 192
*The Sequential model 192 • The Functional API 195
Subclassing the Model class 202 • Mixing and matching different components 204 • Remember: Use the right tool for the job 205*
- 7.3 Using built-in training and evaluation loops 206
*Writing your own metrics 207 • Using callbacks 208
Writing your own callbacks 210 • Monitoring and visualization with TensorBoard 212*
- 7.4 Writing your own training and evaluation loops 214
Training vs. inference 215 • Writing custom training step functions 216 • Low-level usage of metrics 221 • Using fit() with a custom training loop 222 • Handling metrics in a custom train_step() 226
- 8 Image classification 231**
- 8.1 Introduction to ConvNets 232
The convolution operation 234 • The max-pooling operation 239

- 8.2 Training a ConvNet from scratch on a small dataset 241
 - The relevance of deep learning for small-data problems* 242
 - Downloading the data* 242 ▪ *Building your model* 245
 - Data preprocessing* 247 ▪ *Using data augmentation* 252
- 8.3 Using a pretrained model 256
 - Feature extraction with a pretrained model* 256 ▪ *Fine-tuning a pretrained model* 264

9 ConvNet architecture patterns 268

- 9.1 Modularity, hierarchy, and reuse 269
- 9.2 Residual connections 272
- 9.3 Batch normalization 276
- 9.4 Depthwise separable convolutions 278
- 9.5 Putting it together: A mini Xception-like model 280
- 9.6 Beyond convolution: Vision Transformers 282

10 Interpreting what ConvNets learn 284

- 10.1 Visualizing intermediate activations 285
- 10.2 Visualizing ConvNet filters 291
 - Gradient ascent in TensorFlow* 294 ▪ *Gradient ascent in PyTorch* 295 ▪ *Gradient ascent in JAX* 295 ▪ *The filter visualization loop* 296
- 10.3 Visualizing heatmaps of class activation 299
 - Getting the gradient of the top class: TensorFlow version* 302
 - Getting the gradient of the top class: PyTorch version* 302
 - Getting the gradient of the top class: JAX version* 303
 - Displaying the class activation heatmap* 304
- 10.4 Visualizing the latent space of a ConvNet 306

11 Image segmentation 308

- 11.1 Computer vision tasks 308
 - Types of image segmentation* 310
- 11.2 Training a segmentation model from scratch 311
 - Downloading a segmentation dataset* 311 ▪ *Building and training the segmentation model* 314

- 11.3 Using a pretrained segmentation model 318
Downloading the Segment Anything Model 319 ▪ *How Segment Anything works 319* ▪ *Preparing a test image 321* ▪ *Prompting the model with a target point 323* ▪ *Prompting the model with a target box 327*

12 Object detection 329

- 12.1 Single-stage vs. two-stage object detectors 330
Two-stage R-CNN detectors 330 ▪ *Single-stage detectors 332*
- 12.2 Training a YOLO model from scratch 332
Downloading the COCO dataset 332 ▪ *Creating a YOLO model 336* ▪ *Readying the COCO data for the YOLO model 339*
Training the YOLO model 342
- 12.3 Using a pretrained RetinaNet detector 346

13 Timeseries forecasting 351

- 13.1 Different kinds of timeseries tasks 351
- 13.2 A temperature forecasting example 352
Preparing the data 356 ▪ *A commonsense, non-machine-learning baseline 359* ▪ *Let's try a basic machine learning model 360*
Let's try a 1D convolutional model 362
- 13.3 Recurrent neural networks 364
Understanding recurrent neural networks 365 ▪ *A recurrent layer in Keras 368* ▪ *Getting the most out of recurrent neural networks 372* ▪ *Using recurrent dropout to fight overfitting 372*
Stacking recurrent layers 375 ▪ *Using bidirectional RNNs 377*
- 13.4 Going even further 379

14 Text classification 381

- 14.1 A brief history of natural language processing 381
- 14.2 Preparing text data 384
Character and word tokenization 387 ▪ *Subword tokenization 390*
- 14.3 Sets vs. sequences 395
Loading the IMDb classification dataset 396
- 14.4 Set models 398
Training a bag-of-words model 399 ▪ *Training a bigram model 403*

- 14.5 Sequence models 405
Training a recurrent model 406 ▪ *Understanding word embeddings* 409 ▪ *Using a word embedding* 410 ▪ *Pretraining a word embedding* 414 ▪ *Using the pretrained embedding for classification* 418

15 Language models and the Transformer 421

- 15.1 The language model 421
Training a Shakespeare language model 422 ▪ *Generating Shakespeare* 426
- 15.2 Sequence-to-sequence learning 428
English-to-Spanish translation 430 ▪ *Sequence-to-sequence learning with RNNs* 432
- 15.3 The Transformer architecture 437
Dot-product attention 439 ▪ *Transformer encoder block* 444
Transformer decoder block 446 ▪ *Sequence-to-sequence learning with a Transformer* 448 ▪ *Embedding positional information* 451
- 15.4 Classification with a pretrained Transformer 454
Pretraining a Transformer encoder 454 ▪ *Loading a pretrained Transformer* 455 ▪ *Preprocessing IMDb movie reviews* 458
Fine-tuning a pretrained Transformer 460
- 15.5 What makes the Transformer effective? 461

16 Text generation 466

- 16.1 A brief history of sequence generation 468
- 16.2 Training a mini-GPT 470
Building the model 473 ▪ *Pretraining the model* 476
Generative decoding 478 ▪ *Sampling strategies* 480
- 16.3 Using a pretrained LLM 484
Text generation with the Gemma model 485 ▪ *Instruction fine-tuning* 488 ▪ *Low-Rank Adaptation (LoRA)* 490
- 16.4 Going further with LLMs 495
Reinforcement Learning with Human Feedback (RLHF) 495
Multimodal LLMs 498 ▪ *Retrieval Augmented Generation (RAG)* 501 ▪ *“Reasoning” models* 502
- 16.5 Where are LLMs heading next? 504

17 Image generation 508

- 17.1 Deep learning for image generation 508
 - Sampling from latent spaces of images 509* ▪ *Variational autoencoders 510* ▪ *Implementing a VAE with Keras 513*
- 17.2 Diffusion models 518
 - The Oxford Flowers dataset 520* ▪ *A U-Net denoising autoencoder 521* ▪ *The concepts of diffusion time and diffusion schedule 523* ▪ *The training process 525* ▪ *The generation process 527* ▪ *Visualizing results with a custom callback 528*
 - It's go time! 529*
- 17.3 Text-to-image models 531
 - Exploring the latent space of a text-to-image model 533*

18 Best practices for the real world 538

- 18.1 Getting the most out of your models 539
 - Hyperparameter optimization 539* ▪ *Model ensembling 546*
- 18.2 Scaling up model training with multiple devices 548
 - Multi-GPU training 548* ▪ *Distributed training in practice 550*
 - TPU training 555*
- 18.3 Speeding up training and inference with lower-precision computation 556
 - Understanding floating-point precision 556* ▪ *Float16 inference 558* ▪ *Mixed-precision training 559* ▪ *Using loss scaling with mixed precision 559* ▪ *Beyond mixed precision: float8 training 560* ▪ *Faster inference with quantization 561*

19 The future of AI 564

- 19.1 The limitations of deep learning 564
 - Deep learning models struggle to adapt to novelty 565*
 - Deep learning models are highly sensitive to phrasing and other distractors 567* ▪ *Deep learning models struggle to learn generalizable programs 569* ▪ *The risk of anthropomorphizing machine-learning models 569*
- 19.2 Scale isn't all you need 570
 - Automatons vs. intelligent agents 571* ▪ *Local generalization vs. extreme generalization 573* ▪ *The purpose of intelligence 575*
 - Climbing the spectrum of generalization 575*

- 19.3 How to build intelligence 576
The kaleidoscope hypothesis 577 ▪ *The essence of intelligence: Abstraction acquisition and recombination* 578
The importance of setting the right target 578 ▪ *A new target: On-the-fly adaptation* 580 ▪ *ARC Prize* 581 ▪ *The test-time adaptation era* 582 ▪ *ARC-AGI 2* 583
- 19.4 The missing ingredients: Search and symbols 584
The two poles of abstraction 585 ▪ *Cognition as a combination of both kinds of abstraction* 587 ▪ *Why deep learning isn't a complete answer to abstraction generation* 588 ▪ *An alternative approach to AI: Program synthesis* 589 ▪ *Blending deep learning and program synthesis* 590 ▪ *Modular component recombination and lifelong learning* 592 ▪ *The long-term vision* 593

20 Conclusions 595

- 20.1 Key concepts in review 595
Various approaches to artificial intelligence 596 ▪ *What makes deep learning special within the field of machine learning* 596
How to think about deep learning 597 ▪ *Key enabling technologies* 598 ▪ *The universal machine learning workflow* 599 ▪ *Key network architectures* 600
- 20.2 Limitations of deep learning 605
- 20.3 What might lie ahead 606
- 20.4 Staying up to date in a fast-moving field 607
Practice on real-world problems using Kaggle 607 ▪ *Read about the latest developments on arXiv* 607 ▪ *Explore the Keras ecosystem* 608
- 20.5 Final words 608
- index* 609