
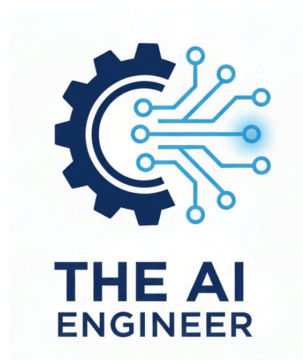


Deep Learning Basics with PyTorch

An approachable, code-first introduction to deep learning

Dr. Yves J. Hilpisch 

February 18, 2026



¹Get in touch: <https://linktr.ee/dyjh>. Web page: <https://theaiengineer.dev>. Research, structuring, drafting, and visualizations were assisted by GPT 5.x as a co-writing tool under human direction. Comments and feedback are welcome.

Preface

Deep learning has moved from research labs into everyday products. From recommendation systems to generative models like GPTs, diffusion image generators, and multimodal assistants, modern AI is powered by deep neural networks. This book gives you a clear, code-first path into the field using PyTorch — a dominant framework in research and widely used in industry for NLP, LLMs, and beyond.

Why Deep Learning, Why Now

Classical machine learning remains essential, but many real-world problems demand flexible models that learn rich representations directly from data. Deep learning delivers:

- End-to-end learning: fewer hand-crafted features, more learning from raw inputs.
- Scale: models improve with more data, parameters, and compute.
- Transfer: pretrained models (e.g., BERT, GPT) can be adapted efficiently to new tasks.

The “transformer era” accelerated this trend. Attention mechanisms and large-scale training unlocked breakthroughs in language (GPT-style LLMs), vision (ViTs, diffusion), audio, and cross-modal reasoning. Understanding these foundations prepares you to contribute and to apply them responsibly.

Why PyTorch

PyTorch combines a clean, Pythonic API with eager execution and mature tooling. It is ubiquitous in:

- NLP and LLMs: most open-source transformer stacks and training recipes target PyTorch first.
- Research: dynamic graphs and strong ecosystem support rapid iteration and reproducibility.
- Production: TorchScript, ONNX export, and optimized runtimes make deployment practical.

We use PyTorch throughout to keep code concise, readable, and close to the math.

What You’ll Learn (At a Glance)

This book is organized into five parts, each building on the last, plus practical appendices.

- **Part I** — data → features → models → metrics; limits of classical approaches.
- **Part II** — tensors, autograd, layers, activations, clean training loops with `nn.Module`.

- [Part III](#) — input pipelines, regularization, LR schedules, CNNs, and robust training at scale.
- [Part IV](#) — sequences and embeddings, attention and transformers, and reliable large-model training (DDP, AMP, checkpoints).
- [Part V](#) — ethics, risk, documentation, governance, and a learning path to keep growing.

Appendices provide quick references and runnable resources:

- [Appendix A](#) — Python & NumPy.
- [Appendix B](#) — Probability and Statistics.
- [Appendix C](#) — Linear Algebra.
- [Appendix D](#) — Calculus.
- [Appendix E](#) — Installation & Environment.
- [Appendix F](#) — Full Scripts.
- [Appendix G](#) — Notebooks Index.
- [Appendix H](#) — Glossary.

How to Use This Book

Every chapter is paired with runnable code and figures that you can reproduce.

- Code-first: every concept is paired with a minimal, runnable example. Figures are reproducible and generated from scripts in `code/figures/`.
- Compact math: just enough notation to reason about shapes, derivatives, and objectives — always tied back to code.
- Visual feedback: decision boundaries, loss curves, and feature maps build intuition at each step.

Who This Book Is For

Python programmers who want a clear path from ML fundamentals to modern deep learning with PyTorch. If you know basic Python and NumPy, you're ready. A refresher is provided in [Appendix A](#).

Setup and Resources

This book supports both local and cloud workflows.

- Local or Colab: [Appendix E](#) walks through both. A GPU is optional for most early chapters, but helpful later for CNNs and transformer demos.
- Companion repository: provides the runnable code and notebooks with a simple layout — `code/` (scripts), `figures/` (outputs), `notebooks/` (experiments).
- Notebooks and full scripts: see [Appendix F](#) and [Appendix G](#) for complete listings and links.

Companion Repository

Browse the code and notebooks online or clone locally:

- Code and notebooks: github.com/yhilpisch/dlcode
- Structure: `code/` (chapter scripts and figure generators), `notebooks/` (interactive notebooks), `figures/` (generated images)

The rendered book (HTML/PDF) stands alone. The companion repository mirrors the structure so you can run, modify, and extend the examples at your own pace.

A Note on Scope

We emphasize supervised learning, practical training workflows, and the transformer foundations behind today’s LLMs. The field is broad — reinforcement learning, self-supervised pretraining, and retrieval-augmented generation are noted where relevant and included in the “Next Steps” chapter for further study.

Thanks

This project stands on the shoulders of the PyTorch community and open-source contributors who share code, papers, and tools. Special thanks to readers who report issues and suggest improvements — the book is better because of you. Enjoy the journey, and build something you can show.

Technical & Legal Note

This book and its code are provided for educational purposes only. Examples are simplified and may omit edge cases; verify results and assumptions for your context, especially in safety-, medical-, or financial-critical settings. The author and publisher provide the material “as is” without warranties and are not liable for losses arising from use; external dependencies and links can change over time.

Contents

Preface	i
I Foundations of Machine Learning	1
1 Introduction to Machine Learning	2
1.1 What Is Machine Learning?	2
1.2 Types of ML	3
1.3 The ML Workflow	3
1.4 A Minimal Example: Linear Regression	3
1.5 First Run (Environment Check)	5
1.6 Sanity Box	5
1.7 Common Pitfalls	5
1.8 Exercises	5
1.9 Where We're Heading Next	6
2 Data, Features, and Splits	7
2.1 What Is Data in ML?	7
2.2 Splits: Train, Validation, Test	8
2.3 Visualizing Features (Iris)	8
2.4 Example: Iris Classification (Pipeline)	8
2.5 Decision Boundaries in 2D	10
2.6 Sanity Box	11
2.7 Common Pitfalls	12
2.8 Exercises	12
2.9 Where We're Heading Next	12
3 Basic Models	13
3.1 A Tiny Playground Dataset	13
3.1.1 Noisy Line for Regression	13
3.1.2 Two Moons for Classification	14
3.2 Linear Regression (scikit-learn)	14
3.3 Logistic Regression (Classification)	15
3.4 Decision Trees (A Taste)	18
3.5 SVMs (Margins and Kernels)	18
3.6 Model Evaluation: Metrics	19
3.7 Sanity Box	19
3.8 Common Pitfalls	19
3.9 Exercises	20

4	Limits of Classical ML	21
4.1	The Curse of Dimensionality (A Taste)	21
4.2	The Cost of Feature Engineering	22
4.3	When Capacity Becomes the Limiter	22
4.4	Why Neural Networks? (Representation Learning)	23
4.5	Transition: From Handcrafted to Learned Features	24
4.6	Visual: Under/Overfitting and Learning Curves	24
4.6.1	Learning Curves via <code>sklearn.model_selection.learning_curve</code>	26
4.7	Preparing for PyTorch	27
4.8	Sanity Box	27
4.9	Common Pitfalls	27
4.10	Exercises	27
4.11	Where We're Heading Next	28
II	Neural Networks and PyTorch Basics	29
5	First Steps with PyTorch	30
5.1	Setup	30
5.2	Tensors: Creation, Shapes, Dtypes, Devices	31
5.3	NumPy Interop (Zero-Copy When Possible)	31
5.4	Views vs Copies (Reshape, View, Clone)	31
5.5	Broadcasting Rules (Write Less, Compute More)	31
5.6	Autograd: Gradients for Learning	32
5.7	Sanity Box	33
5.8	Common Pitfalls	33
5.9	Exercises	34
5.10	Where We're Heading Next	34
6	Building Blocks of Neural Networks	36
6.1	The Perceptron (Neuron)	36
6.2	Activation Functions (Shapes and Intuition)	36
6.3	From Neuron to Layer to Network	38
6.4	Why Nonlinearity? XOR as a Counterexample	38
6.5	Sanity Box	38
6.6	Common Pitfalls	39
6.7	Exercises	39
6.8	Where We're Heading Next	39
7	Training Neural Networks	40
7.1	A Minimal MLP for Toy Classification	40
7.2	Losses, Backward, and the Training Step	41
7.3	SGD vs Adam (Optimizer Comparison)	41
7.4	Sanity Box	43
7.5	Common Pitfalls	44
7.6	Exercises	44
7.7	Where We're Heading Next	44

8 Organizing with <code>nn.Module</code>	45
8.1 Refactor the Tiny MLP with <code>nn.Module</code>	45
8.2 A Clean Training/Eval Loop	46
8.3 Save and Load with <code>state_dict</code>	47
8.4 Reusable Training Loop Template	47
8.5 DataLoader-Ready Loop (Preview)	48
8.6 Optional: Dataclass Config	48
8.7 Putting It Together (Mini Demo)	49
8.8 Sanity Box	49
8.9 Common Pitfalls	50
8.10 Exercises	50
8.11 Where We're Heading Next	50
 III Supervised Deep Learning in Practice	 51
9 Data Pipelines	52
9.1 Dataset and DataLoader Basics (Moons)	52
9.2 Transforms (Standardization and Composition)	53
9.3 Custom Dataset and Collate (Variable Length)	54
9.4 Shuffle vs No Shuffle (Why Shuffling Matters)	55
9.5 Optional: Handling Imbalance with Weighted Sampling	57
9.6 Optional: Caching Heavy Transforms	57
9.7 Sanity Box	58
9.8 Common Pitfalls	58
9.9 Exercises	58
9.10 Where We're Heading Next	59
10 Improving Training	60
10.1 A Running Setup (Moons with Validation Split)	60
10.2 Diagnosing Over/Underfitting	61
10.3 Smoother Boundaries with Weight Decay	61
10.4 Dropout (When and How Much?)	63
10.5 Early Stopping (Keep the Best Validation Model)	63
10.6 Learning-Rate Schedules (Keep It Simple)	64
10.7 Sanity Box	64
10.8 Common Pitfalls	64
10.9 Exercises	65
10.10 Where We're Heading Next	65
11 Deeper Architectures	66
11.1 Convolution by Example (2D, Single Channel)	66
11.2 Stride and Padding (Shape and Information)	67
11.3 Pooling (Downsampling and Invariance)	67
11.4 A Tiny CNN in PyTorch	68
11.5 Sanity Box	68
11.6 Common Pitfalls	69
11.7 Exercises	69
11.8 Where We're Heading Next	69

12 Training at Scale	71
12.1 Throughput vs Batch Size	71
12.2 Mixed Precision (AMP)	71
12.3 Gradient Accumulation (Memory-Bound Batches)	72
12.4 Learning-Rate Schedules (Cosine)	74
12.5 Checkpoints (Robust Resumes)	74
12.6 Sanity Box	75
12.7 Common Pitfalls	75
12.8 Exercises	75
12.9 Where We're Heading Next	75
 IV Toward Large Language Models	 77
13 Sequences and Language	78
13.1 Tokenization and Padding	78
13.2 Embeddings (Dense Representations)	79
13.3 A Tiny Sentiment Toy	79
13.4 Common Pitfalls	80
13.5 Exercises	81
13.6 Where We're Heading Next	81
14 RNNs, Attention, and Transformers	82
14.1 From RNNs to Attention (Motivation)	82
14.2 Scaled Dot-Product Attention (Single Head)	83
14.3 Masks: Padding vs Causal	84
14.4 Multi-Head Attention (Shapes Matter)	84
14.5 Positional Information	85
14.6 Transformer Block (Pre-Norm)	86
14.7 Common Pitfalls	86
14.8 Exercises	87
14.9 Where We're Heading Next	87
15 Training Large Models	88
15.1 Why Distributed? Throughput and Wall-Clock	88
15.2 Effective Batch Size and Gradient Accumulation	89
15.3 AMP and Loss Scaling	89
15.4 Launching DDP (Single Node)	90
15.5 Throughput: When to Scale Out	91
15.6 Checkpointing Across Ranks	91
15.7 Multi-Node Sketch	91
15.8 Memory-Saving Techniques (Preview)	91
15.9 Common Pitfalls	92
15.10 Exercises	92
15.11 Where We're Heading Next	93
 V Broader Context and Next Steps	 94
16 Ethics, Risks, and Applications	95
16.1 Why Responsibility Is a Systems Concern	95
16.2 Data Documentation and Consent	95

16.3 Measuring Fairness	96
16.4 Interpretability and Monitoring	97
16.5 Privacy and Security — Baselines First	97
16.6 Shipping With Guardrails	97
16.7 Common Pitfalls	97
16.8 Exercises	98
16.9 References	98
16.10 Where We're Heading Next	98
17 Next Steps	99
17.1 Projects That Matter (and Fit)	99
17.2 A Short, Biased Reading List	100
17.3 Efficient Fine-Tuning: LoRA/QLoRA in Practice	100
17.4 Reproducible Research Habits	101
17.5 Example Project Scopes	101
17.6 Specialization Tracks	103
17.7 Exercises	103
17.8 References	103
17.9 Where We're Heading Next	104
A Python & NumPy	105
A.1 Quick Python Essentials	105
A.2 NumPy: Arrays and dtypes	105
A.3 Shapes, Indexing, Slicing	106
A.4 Broadcasting and Vectorization	106
A.5 Random Numbers and Reproducibility	106
A.6 Linear Algebra in a Nutshell	106
A.7 Quick Visualizations	107
A.8 Cheat Sheet	107
A.9 Figure Generators (Reference)	109
B Probability and Statistics	112
B.1 Random Variables and Events	112
B.2 Expectation, Variance, Covariance	112
B.3 Core Distributions We'll Use	112
B.4 Law of Large Numbers (LLN)	113
C Linear Algebra	116
C.1 Vectors, Norms, and Angles	116
C.2 Matrices as Linear Maps	117
C.3 Matrix Multiplication and Shapes	117
C.4 Transpose, Inverse, Determinant	117
C.5 Solving Linear Systems	118
C.6 Eigenvectors (Quick View)	118
C.7 Figure Generators (Reference)	119
D Calculus	123
D.1 From Differences to Derivatives	123
D.2 Basic Rules You'll Actually Use	123
D.3 Partial, Gradients, and Notation	124
D.4 Tiny Gradient Descent (2D Quadratic)	125
D.5 Figure Generators (Reference)	126

E	Installation & Environment	130
E.1	Option A — Use Google Colab (Easiest)	130
E.2	Option B — Local Install (Recommended for Iteration)	130
E.3	Option C — Miniconda/Conda (Alternative)	132
E.4	GPU Acceleration (Optional)	132
E.5	Troubleshooting	132
E.6	Quick Start (Copy/Paste)	133
F	Full Scripts	134
F.1	Chapter 1 — Introduction to Machine Learning	134
F.2	Chapter 2 — Data, Features, and Representations	137
F.3	Chapter 3 — Basic Models in scikit-learn	144
F.4	Chapter 4 — The Limits of Classical ML	152
F.5	Chapter 5 — First Steps with PyTorch	158
F.6	Chapter 6 — Building Blocks of Neural Networks	165
F.7	Chapter 7 — Training Neural Networks	170
F.8	Chapter 8 — Organizing Code with <code>torch.nn</code>	176
F.9	Chapter 9 — Working with Data in PyTorch	181
F.10	Chapter 10 — Improving Training	192
F.11	Chapter 11 — Deeper Architectures	197
F.12	Chapter 12 — Training at Scale	201
F.13	Chapter 13 — Sequences and Language Data	207
F.14	Chapter 14 — Recurrent and Attention-based Models	210
F.15	Chapter 15 — Training Large Models (DDP, AMP, Checkpointing)	218
F.16	Chapter 16 — Ethics, Risks, and Applications	226
F.17	Chapter 17 — Next Steps for the Reader	228
G	Notebooks Index	232
G.1	Index by Chapter	232
G.2	Appendix Notebooks	233
G.3	Reproducibility Notes	233
H	Glossary	234

List of Figures

1.1	Linear regression on a toy dataset.	4
2.1	Iris petal length/width by species (well separated in 2D).	9
2.2	Iris sepal length/width by species (less separable in 2D).	9
2.3	Iris logistic regression confusion matrix (petal features).	10
2.4	Iris decision regions with petal features.	11
3.1	Linear regression on noisy data (fit and residual pattern).	14
3.2	Two-moons dataset (before modeling).	15
3.3	Residuals vs. x (check for curvature or structure).	16
3.4	Residual histogram (rough symmetry around zero).	16
3.5	Confusion matrix — logistic regression on moons.	17
3.6	Decision boundaries: logistic regression vs. RBF SVM on moons.	17
3.7	Logistic regression probability contour with 0.5 boundary.	18
4.1	Distance contrast shrinks quickly with dimension.	22
4.2	Polynomial feature count grows combinatorially.	23
4.3	Kernel methods and the n^2 memory wall.	23
4.4	Decision tree accuracy vs depth (under/overfitting).	25
4.5	Learning curve for logistic regression on moons.	25
4.6	Residuals reveal misspecification (curved pattern).	26
5.1	Broadcasted sum heatmap (shape 3×4).	32
5.2	Computation graph for a linear layer + MSE loss.	32
5.3	Quadratic with gradient and GD steps.	34
6.1	A simple neuron (inputs, weights, bias, activation).	37
6.2	Activation functions: sigmoid, tanh, ReLU.	37
6.3	Two-layer MLP solves XOR; linear cannot.	38
7.1	Training loss over iterations (mini demo).	42
7.2	Train/validation curves across epochs.	42
7.3	Decision boundary after training (tiny MLP on moons).	43
8.1	Training loss by epoch (nn.Module).	47
8.2	Decision boundary after training (nn.Module MLP).	49
9.1	Raw vs standardized features.	54
9.2	Padded batch heatmap (zeros added to the right).	55
9.3	Data pipeline: Dataset \rightarrow DataLoader \rightarrow Model.	56
9.4	Batch size affects training dynamics (loss curves).	56
9.5	Loss with and without shuffling.	57
9.6	Class imbalance vs weighted sampling.	57

9.7 Epoch time with on-the-fly vs precomputed transforms.	58
10.1 Train/validation curves — no reg vs weight decay.	62
10.2 Decision boundary: none vs stronger weight decay.	62
10.3 Dropout $p = 0.0$ vs $p = 0.3$ — validation curves.	63
10.4 LR schedules: constant, step decay, cosine.	64
11.1 Sliding a 3×3 kernel: edge-emphasizing response.	67
11.2 Padding vs stride — output sizes side-by-side.	67
11.3 Pooling (2×2 , stride 2) on a toy feature map.	68
11.4 Feature map sizes through the CNN.	69
12.1 Throughput vs batch size (toy timing).	72
12.2 Throughput with AMP vs FP32.	73
12.3 Accumulation factor vs memory footprint.	73
12.4 Cosine vs constant learning rate (loss curves).	74
12.5 Checkpoint contents and flow.	75
13.1 Padding and truncation for variable sequence lengths.	79
13.2 Toy embedding space (2D projection).	79
13.3 Mean-embedding vs bag-of-words on a toy sentiment boundary.	80
14.1 Scores, softmax weights, and output components for one token.	83
14.2 Attention as paint mixing: weights act as blending ratios.	83
14.3 Causal mask (upper triangle blocked, $j > i$).	85
14.4 Multi-head attention patterns (2×2 heads).	86
15.1 Single node, 4 GPUs with DDP (one process per GPU).	89
15.2 Effective batch with gradient accumulation and replicas.	89
15.3 GradScaler dynamics on a short run (illustrative).	90
15.4 Illustrative throughput vs GPUs (near-linear to 4, then taper).	91
15.5 ZeRO-style sharding overview (parameters, optimizer state, grads).	92
16.1 Risk checkpoints across the ML pipeline.	96
16.2 Group TPR/FPR disparities (illustrative).	96
16.3 Model card sections you should fill out.	97
17.1 A learning path you can adapt.	101
17.2 Scoping portfolio projects by effort vs impact.	101
A.1 Sine and cosine over $[0, 2\pi]$.	107
A.2 Histogram of 10,000 standard normal samples.	108
A.3 Scatter with a broadcasted transformation.	108
B.1 Binomial($n = 10, p = 0.3$) PMF across k successes.	113
B.2 Standard normal PDF with $\mu = 0, \sigma = 1$.	114
B.3 Running mean of $\mathcal{N}(0, 1)$ samples converges toward 0.	114
C.1 Linear combination in 2D (u, v , and $0.5u + 1.2v$).	116
C.2 Linear map acting on a unit grid (shear + scale).	117
C.3 Eigenvectors of a symmetric 2×2 matrix.	118
D.1 Tangent to $f(x) = x^2$ at $x_0 = 1$.	124
D.2 Finite-difference error decays faster with central differences.	124

D.3 Quadratic contours with a short gradient-descent path.	125
--	-----

Contact

Deep Learning Basics with PyTorch
The AI Engineer

Get in touch:

<https://linktr.ee/dyjh>

<https://theaiengineer.dev>

© 2026 Dr. Yves J. Hilpisch — All rights reserved.

