Introduction to Deep Learning and Its History

Applied Deep Learning — Chapter 1

Yin Yang

Foundation Books Project

Roadmap

Why Deep Learning Now?

AI, Machine Learning, and Deep Learning

What Deep Learning Is in Practice

Artificial vs Biological Neural Networks

A Brief History of Deep Learning

Why Deep Learning Became Dominant

A Practical Checklist for the Rest of the Book

Summary

Why Deep Learning Now?

Why Deep Learning Now?

- Working language of much of modern AI:
  - image classification, speech recognition, machine translation
  - large-scale recommendation, protein structure prediction
  - text generation, coding assistance
- Not a universal tool — but unavoidable for an applied AI map.
- This book starts conceptual, then becomes hands-on.

Treat deep learning as a powerful default to understand — not as the right first tool for every problem.

The Running Example: Handwritten Digits

Goal: receive a grayscale image of a handwritten digit, return one label from $\{0, 1, \ldots, 9\}$.

This small task already contains the core ingredients of applied deep learning:

- input representation
- target label
- model
- training procedure
- way to evaluate success

Later chapters return to it via MNIST and extend it to natural images with CIFAR-10.

Supervised Learning in Compact Notation

Training data:

$$D = \{(x_i, y_i)\}_{i=1}^{n},$$

with input x_i and desired output y_i.

A model f_θ uses parameters θ to map inputs to predictions; a loss $\ell(f_\theta(x_i), y_i)$ scores one prediction.

Training tries to find parameters with small average loss:

$$\theta^\star \approx \arg\min_\theta \frac{1}{n} \sum_{i=1}^{n} \ell(f_\theta(x_i), y_i).$$

Choose the loss so that low loss means better task behavior under the metric and split you plan to report.

AI, Machine Learning, and Deep Learning

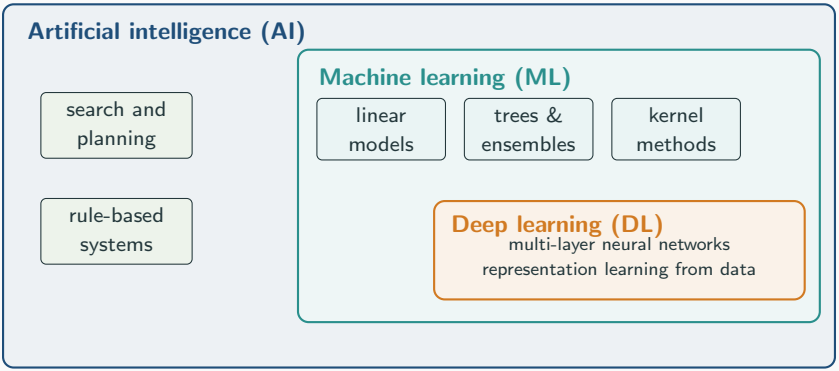
Three Definitions

Artificial intelligence (AI)
Building systems that perform tasks associated with intelligent behavior — perception, reasoning, planning, language, decision-making, control.

Machine learning (ML)
Methods that improve performance on a task by using *data or experience* rather than only hand-written rules.

Deep learning (DL)
Multi-layer neural networks that learn *representations* and task-specific functions from data.

A Map: AI ⊃ ML ⊃ DL



- Beginner-friendly: DL ⊂ ML ⊂ AI.
- Incomplete: AI also has non-ML methods; ML has many non-DL families.

Three Approaches to Digit Classification

Rule-based AI Hand-written rule: "one closed loop and a vertical stroke $\Rightarrow$ 9."

Classical ML Hand-designed features (stroke counts, aspect ratio, edge histograms), then a classifier such as an SVM.

Deep learning Learn the intermediate representation *and* the classifier. Designer chooses architecture, loss, training procedure — the model adjusts its own internal weights from data.

Deep learning has not replaced classical ML. Small data, interpretability, or simple structure may favor simpler methods.

What Deep Learning Is in Practice

Layered Representation Learning

"At least two hidden layers" is a classroom rule, not a definition.

The real idea: layered representation learning.

- Several trainable transformations applied in sequence.
- Earlier layers learn simple patterns.
- Later layers compose them into task-specific concepts.
- "Deep" = composition of many trainable stages, not a magic layer count.

Image example: edges $\rightarrow$ textures $\rightarrow$ shapes $\rightarrow$ object parts $\rightarrow$ prediction.

One Artificial Neuron — A Concrete Calculation

Inputs $x = (0.9, 0.2, 0.1)$, weights $w = (1.0, -1.5, 0.5)$, bias $b = 0.2$.

Pre-activation:

$$z = 0.9(1.0) + 0.2(-1.5) + 0.1(0.5) + 0.2 = 0.85.$$

ReLU activation:

$$a = \max(0, z) = \max(0, 0.85) = 0.85.$$

The pattern: *multiply, add, apply a nonlinearity, pass to the next layer.* Stack many such units and the network becomes a trainable composition of simple operations — not one mysterious formula.

A Small MNIST-Sized Network

Layer sizes: $784 \to 256 \to 128 \to 10$.

- Input 784: a 28×28 image flattened.
- Output 10: one logit per digit class.

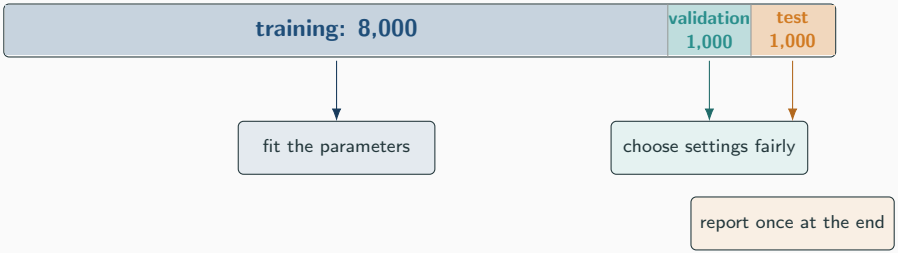
Weight count:

$$784 \times 256 + 256 \times 128 + 128 \times 10 = 234{,}752.$$

Plus biases: $256 + 128 + 10 = 394$.

Total: 235,146 trainable parameters — already substantial for a toy problem.

Train / Validation / Test Discipline



The test set is not a dashboard. Tuning on it converts final evaluation into hidden model selection.

Artificial vs Biological Neural Networks

Inspiration, Not Fidelity

Artificial neuron

- Numeric inputs
- Weighted sum + bias
- Activation function
- Output value

$$a = \phi(w^\top x + b)$$

A useful abstraction for optimization and matrix computation.

Biological neuron

- Many synaptic inputs
- Membrane and cell dynamics
- Spiking output
- Neurotransmitters, dendrites, axons
- Metabolism, anatomy, time

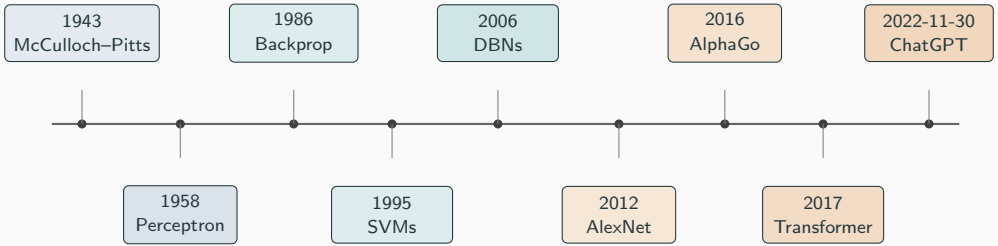
Useful analogy: connected units that adapt with experience. Important limit: inspiration $\neq$ biological fidelity.

Why the Distinction Matters

- A standard feedforward network is not a faithful miniature of the human brain.
- Biology inspired part of deep learning; ANNs can give neuroscience useful tools and hypotheses — but the mechanisms differ.
- **Project Prakash:** after sight restoration, visual parsing and recognition still require learning and experience — not switched on instantly.
- **Connectome-constrained fly networks:** ANN architectures shaped by measured connectivity, evaluated on specific neural-prediction tasks. Not a literal uploaded brain.
- **Neuralink PRIME:** a brain–computer interface decoding brain activity — biomedical engineering, not evidence of equivalence.

A Brief History of Deep Learning

A Compact Timeline



Pattern: cycles, not steady linear progress. Enthusiasm → criticism → renewed methods → benchmark breakthroughs.

Early Neural Models (1943–1958)

- **1943 – McCulloch & Pitts:** simplified neurons combined with logical structure into a computational model.
- **1949 – Hebb's Organization of Behavior:** learning tied to changing relationships among neurons.
- **1958 – Rosenblatt's perceptron:** translated inspiration into a concrete trainable classifier.

These models made learning machines *thinkable in quantitative terms* — but optimism often outran capability.

Criticism, Competition, and Revival (1969–2006)

- **1969 – Minsky & Papert, Perceptrons:** important limitations of single-layer perceptron-style models. Often retold as the sole cause of an "AI winter" — the actual story was broader (data, hardware, training methods, expectations).
- **1980 – Fukushima's neocognitron:** anticipated convolutional ideas.
- **1986 – Backpropagation** (Rumelhart, Hinton, Williams): a practical learning method for multi-layer networks.
- **1990s–2000s – SVMs and kernel methods:** often competitive with neural networks on small/medium data.
- **2006 – Deep belief networks:** Hinton, Osindero, Teh; deeper networks could be trained more effectively than assumed.

Deep learning didn't win because everything else was foolish — later conditions favored it.

Large-Scale Breakthroughs (2012–)

- **2012 – AlexNet:** won ImageNet by a large margin; large CNNs on GPUs decisively beat prior approaches. Question shifted from "neural network at all?" to "which architecture and recipe?"
- **2016 – AlphaGo:** deep nets + search + RL master Go.
- **2017 – Transformer:** non-recurrent attention architecture.
- **2018–2019 – BERT:** bidirectional Transformer pretraining yields reusable language representations.
- **2020 – DDPMs** & **2022 – latent diffusion:** drive high-quality image generation.
- **2021 – AlphaFold2:** protein structure prediction.
- **2022-11-30 – ChatGPT:** foundation-model interaction reaches a broad public.

Why Go Was a Hard AI Benchmark

- **Large branching:** many legal moves; several look plausible.
- **Delayed value:** effects can appear many turns later.
- **Local and global:** small fights change whole-board territory.

AlphaGo's lesson
Deep neural networks worked when combined with search and reinforcement learning — not as a single drop-in replacement for prior methods.

Why Deep Learning Became Dominant

Several Forces Aligned

Enabling factors

- Larger datasets
- GPUs and accelerators
- Better optimization & regularization
- Architectures and transfer learning
- Software, tooling, and benchmarks

Resulting outcomes

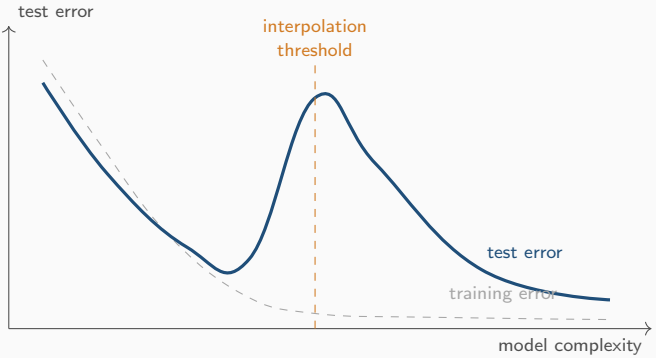
- High-performance vision, speech, language
- Faster reuse via pretrained models
- Stronger workflows: experimentation, scaling, sharing

Important limit: dominant is not universal. Small tabular problems and high-interpretability settings may still favor simpler methods.

Scale Changes the Question

Setting	Data scale	Model / run scale	Applied lesson
Handwritten digits	60,000 train, 10,000 test; 28×28 grayscale	Small FC net: 235,146 parameters	Small enough for shape checks and hand arithmetic
ImageNet-scale vision	$>$ 1M images, 1000 classes	AlexNet: large CNN trained on GPUs	Learned features won when data and compute aligned
Experiment budget	10-min run $\rightarrow$ 10 trials $\approx$ 100 min	6-hour run $\rightarrow$ 10 trials $\approx$ 60 hours	Compute cost shapes baselines, seeds, and ablations

Double Descent (Schematic)



Larger models can interact with data, optimization, and regularization in ways the simple overfitting story misses.

Software Maturity Cuts Both Ways

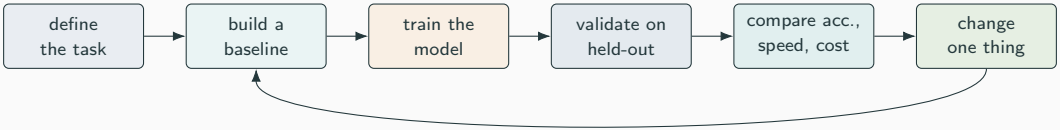
Today, a first neural network can be defined in a few lines of Keras or PyTorch and trained on standard hardware.

- **Upside:** useful experiments are accessible; many components are now reliably implemented.
- **Downside:** easy to trust a run with the wrong split, shifted labels, or accidental train/val preprocessing differences.

Educational order in this book: visible experiments first, then programming, then model families and training recipes.

A Practical Checklist for the Rest of the Book

The Experiment Loop



Evidence-driven, not hype-driven: stable baseline, honest validation, one variable at a time.

The Minimum Useful Experiment Record

Write it before the run starts:

- task, data split, baseline
- model, changed setting, seed
- hardware / runtime
- validation metric, final-test rule
- elapsed time, artifact path

Useful: "ResNet18 on CIFAR-10, seed 1234, AdamW warmup changed from no warmup to one epoch, MPS, validation accuracy 41.4%, 10.7s, artifacts in `runs/resnet18-warmup`, test set still locked."

Not useful: "warmup helped."

Six Habits Going Forward

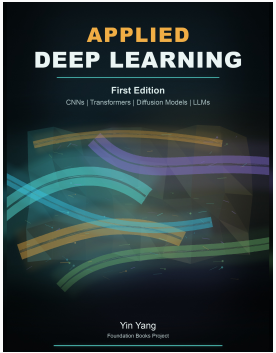
1. **Define the task before changing the model.**
2. **Start from a correct baseline** — shapes, parameter counts, validation behavior, final evaluation discipline.
3. **Check basic failure points first** — loader, labels, shapes, loss decreases — before tuning architecture or optimizer.
4. **Compare quality, speed, and cost together.**
5. **Report domain-specific evidence** — error examples, class confusions, masks, cited sources, evaluation episodes.
6. **Reuse mature tools, then change one thing at a time.**

Summary

Summary

- **Map:** AI $\supset$ ML $\supset$ DL — useful first approximation, but AI has non-ML methods and ML has many non-DL families.
- **Deep learning** = layered representation learning with multi-layer neural networks. Not a magic layer count.
- **Biology** inspired the term *neural*, but ANNs are engineering abstractions, not faithful copies of brains.
- **History** is cyclical: early ideas, criticism, renewed methods, AlexNet, AlphaGo, Transformer, AlphaFold2, ChatGPT.
- **Dominance came from alignment** of data, compute, optimization, architectures, and software — not a single invention.
- **Carry forward an evidence standard:** make claims only as large as the task, split, metric, baseline, and error analysis support.

Next: TensorFlow Playground, MNIST in Python, CNN basics, neural-network training, and CNN architectures.



Applied Deep Learning

Chapter overview slides for the book by Yin Yang.

Book page	`foundation-books.github.io/applied-deep-learning`
Code	`github.com/foundation-books/applied-deep-learning-code`
Print	`amazon.com/dp/B0GZF7QRSH`
Kindle	`amazon.com/dp/B0GZF9221X`
License	CC BY-NC-ND 4.0

These free overview slides may be shared non-commercially with attribution and without modifications. Full explanations, exercises, and companion-code context are in the book and linked repository.