OVERVIEW
Machine Learning & Deep Learning

A depth-first curriculum, from math foundations to production

Phase goal

Turn "I can call `.fit()`" into "I understand, can implement, and can ship." This guide is the map for the eight phases that follow; each phase ships as its own self-contained PDF.

Estimated time: 9–14 months at 8–12 hrs/week | **Track:** Comprehensive ML & Deep Learning Curriculum

A self-paced, depth-first path from mathematical foundations to production deep learning.

Contents

1 How to use this curriculum

This is a *depth-first* path written for someone who already programs well. The assumption is Python comfort and solid software-engineering instincts; the gap we are closing is the *conceptual and mathematical* one. Every phase therefore insists that you implement core ideas from scratch at least once before reaching for a library. You only truly understand backpropagation after you have computed a gradient by hand and watched your NumPy version agree with PyTorch's autograd to five decimal places.

Key idea

The fastest way to *forget* machine learning is to learn it as a sequence of API calls. The fastest way to *own* it is to alternate between three modes: (1) derive the math, (2) implement it from scratch, (3) then use the production library and understand exactly what it is doing for you. Each phase is built around that loop.

1.1 The three-pass method for every topic

1. **Intuition pass.** Watch a video (3Blue1Brown, StatQuest, Karpathy) or read a blog until you can explain the idea to a colleague in two sentences without equations.
2. **Derivation pass.** Work the math with pen and paper. Re-derive at least the central result (the normal equation, the backprop recursion, the attention formula).
3. **Implementation pass.** Code it from scratch in NumPy, verify against a reference library, *then* learn the library's idioms.

1.2 What "done" looks like

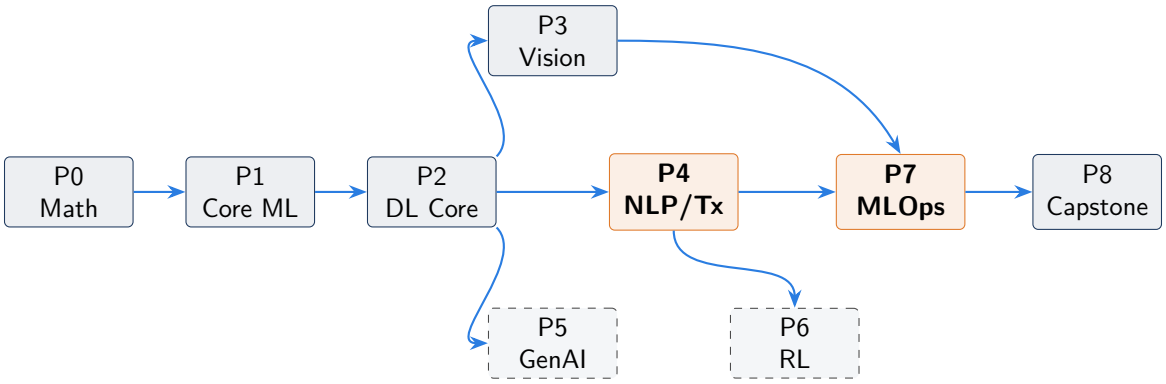
Each phase ends with a **checkpoint project**. Do not skip these. They are the difference between recognition ("I have seen ResNet") and recall-with-application ("I can fine-tune a ResNet, explain skip connections, and serve it behind an API"). Treat each checkpoint as a small portfolio piece: a clean repository, a short write-up, and a reproducible result.

2 The eight phases at a glance

#	Phase	What you walk away able to do	Time
0	Math & Tooling	Read ML papers' math; implement gradient descent from scratch	3–5 wk
1	Core ML	Implement and tune classical models; build leakage-free pipelines	6–8 wk
2	DL Foundations	Hand-derive backprop; train MLPs in raw NumPy and PyTorch	6–8 wk
3	Computer Vision	Fine-tune CNNs/ViTs; understand detection & segmentation	4–6 wk
4	NLP & Transformers	Build a Transformer and a RAG system from first principles	6–8 wk
5	Generative Models	Train VAEs/GANs/diffusion; understand modern image generation	4–5 wk
6	Reinforcement Learning	Implement DQN and policy gradients; understand RLHF	3–4 wk
7	MLOps & Deployment	Containerize, serve, monitor, and CI/CD an ML system	4–6 wk
8	Capstones	Ship 2–3 end-to-end projects in your own domains	ongoing

2.1 Dependency graph

The arrows show hard prerequisites. Phases 3, 5, and 6 branch off the deep-learning core and can be reordered or deferred; Phase 4 (NLP/Transformers) and Phase 7 (MLOps) carry the most current career value.



Intuition

Orange = highest current leverage (NLP/Transformers and MLOps). Dashed = optional/deferrable on a first pass (Generative models, RL). If your time is constrained, the

compressed path below skips the dashed nodes and circles back later.

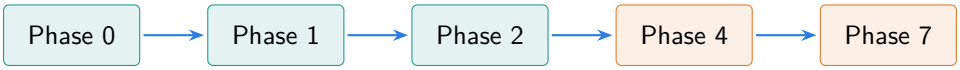
3 Two routes through the material

3.1 The complete route (recommended)

Phase 0 → 1 → 2 → 3 → 4 → 5 → 6 → 7 → 8. This is the canonical order and builds the most robust mental model. Budget 9–14 months.

3.2 The compressed high-value route

Phase 0 → 1 → 2 → 4 → 7, then capstones. This skips computer vision, generative models, and RL on the first pass and concentrates on the two areas where most real-world and career value currently sits: *transformers/NLP/LLMs* and *deployment*. You can always return for Phases 3, 5, and 6. Budget 5–7 months.



The compressed route: foundations, then transformers, then ship.

4 Tooling you will set up once and reuse

- **Environment:** Python 3.11+, managed with `conda` or `uv` / `venv`. One environment per major phase keeps dependency conflicts contained.
- **Core stack:** `numpy`, `pandas`, `matplotlib`, `scikit-learn`, `jupyterlab`.
- **Deep learning:** `torch` (primary), `torchvision`, later `transformers`, `datasets`.
- **Compute:** a laptop is fine through Phase 2. From Phase 3 on, use a GPU — Google Colab (free/Pro), Kaggle Notebooks (free GPU), or a cloud instance.
- **Experiment hygiene:** Git from day one. Add Weights & Biases or MLflow by Phase 2 so every run is logged.

Common pitfall

Do not let environment setup become a multi-week yak-shave. Get a minimal working stack, start learning, and add tools as a phase demands them. "Notebook that runs" beats "perfect reproducible Docker-Compose monorepo" when you are on Phase 0.

5 A reference library to keep on hand

Books (keep within reach all year)

- **Hands-On Machine Learning** — Aurélien Géron. The best single practical book for Phases 1–3.
- **Mathematics for Machine Learning** — Deisenroth, Faisal, Ong. Free PDF; pairs with Phase 0.
- **Deep Learning** — Goodfellow, Bengio, Courville. The field's reference text; dense but authoritative.
- **Designing Machine Learning Systems** — Chip Huyen. The practitioner's MLOps

bible (Phase 7).
- **Reinforcement Learning: An Introduction** — Sutton & Barto. Free PDF; the canonical RL text.

Courses & video series

- Andrew Ng — *Machine Learning* & *Deep Learning* Specializations (Coursera).
- Andrej Karpathy — *Neural Networks: Zero to Hero* (YouTube). Exceptional, code-first.
- Stanford **CS231n** (vision) and **CS224n** (NLP) — free lecture videos.
- **fast.ai** — *Practical Deep Learning for Coders*: a top-down alternative path.
- **StatQuest** (Josh Starmer) and **3Blue1Brown** — the best intuition builders.

Communities & staying current

- **Kaggle** — competitions and, crucially, the winning-solution write-ups.
- **Papers With Code** — papers paired with reference implementations.
- **Hugging Face Papers** / **arXiv** — for staying current after the curriculum.

6 Domain tie-ins (make it compound)

Wherever possible, pull your own problem domains into the checkpoints — shipping/logistics data, bilingual (Chinese↔English) document processing, mobile/on-device inference, and API/backend integration. Learning compounds fastest when each new technique is immediately applied to a problem you already understand deeply. The capstone phase makes this explicit.

How to study with these PDFs

For each phase document: (1) skim the whole PDF once for the map; (2) work section by section, pausing at every *Worked example* to reproduce it yourself; (3) heed every *Common pitfall*; (4) do the checkpoint project before moving on; (5) keep a running "cheat-sheet" of formulas and gotchas in your own words. Teaching the material — even to a rubber duck — is the final test of understanding.