Complete AI Learning Roadmap

A structured, step-by-step guide from absolute beginner to production AI engineer

7 PHASES • 30+ STEPS • ~12 MONTHS • BEGINNER TO ADVANCED

Foundation | Core ML | Deep Learning | NLP | CV | MLOps | Advanced

Contents

- Phase 02 — Core Machine Learning
- Phase 03 — Deep Learning Fundamentals
- Phase 04 — Natural Language Processing (NLP)
- Phase 05 — Computer Vision
- Phase 06 — MLOps & Production AI
- Phase 07 — Specialisation & Advanced Topics

This roadmap is designed to take you from zero programming knowledge to deploying production-ready AI systems. Each phase builds on the previous one. You can move faster or slower depending on your background — the durations are estimates for someone learning part-time (~10 hrs/week).

01 Phase 1 — Foundation — Math & Programming

Duration: 6 – 8 weeks | Difficulty: Beginner

Every AI journey starts here. This phase builds the mathematical intuition and coding fluency that underpin all machine learning algorithms.

■ Mathematics for AI

Master the three pillars of ML mathematics before touching any algorithm:

- **Linear Algebra:** Vectors, matrices, dot products, eigenvalues, SVD.
- **Calculus:** Derivatives, partial derivatives, chain rule, gradient descent intuition.
- **Probability & Statistics:** Distributions, Bayes' theorem, expectation, variance, hypothesis testing.

Resources:

→ *3Blue1Brown — Essence of Linear Algebra (YouTube)*
→ *Khan Academy — Multivariable Calculus*
→ *StatQuest with Josh Starmer (YouTube)*

■ *Don't aim for perfection — build enough intuition to understand WHY algorithms work, then deepen as you go.*

■ Python Programming

Python is the primary language of AI/ML. Focus on:

- Core syntax, functions, OOP, file I/O, error handling.
- **NumPy:** Array operations, broadcasting, vectorisation.
- **Pandas:** DataFrames, merging, groupby, data cleaning.
- **Matplotlib / Seaborn:** Visualising data distributions and relationships.

Resources:

→ *Python.org official tutorial*
→ *CS50P — Harvard (free)*
→ *Kaggle Learn — Pandas micro-course*

■ *Practice daily on small datasets. Kaggle's 'Titanic' and 'Iris' are perfect starter projects.*

■ Phase Outcomes

- Comfortable with matrix operations and calculus rules
- Can write clean Python scripts using NumPy & Pandas
- Able to visualise and describe a dataset statistically

02 Phase 2 — Core Machine Learning

Duration: 8 – 10 weeks | Difficulty: Beginner → Intermediate

Learn how machines learn from data. This phase covers classical ML algorithms, the scikit-learn ecosystem, and the critical skills of evaluation and feature engineering.

■ Supervised Learning Algorithms

- **Regression:** Linear Regression, Ridge, Lasso — predicting continuous values.
- **Classification:** Logistic Regression, Decision Trees, K-Nearest Neighbours, SVM.
- **Ensemble Methods:** Random Forest, Gradient Boosting (XGBoost, LightGBM) — industry workhorses.

Understand the intuition, assumptions, and limitations of each algorithm.

Resources:

→ *Hands-On Machine Learning (Aurélien Géron)*
→ *scikit-learn documentation + examples*
→ *StatQuest ML playlist (YouTube)*

■ *Implement at least one algorithm from scratch (e.g., Linear Regression with gradient descent) to solidify understanding.*

■ Unsupervised Learning

- **Clustering:** K-Means, DBSCAN, Hierarchical clustering.
- **Dimensionality Reduction:** PCA, t-SNE, UMAP — visualising high-dimensional data.
- **Anomaly Detection:** Isolation Forest, One-Class SVM.

Resources:

→ *sklearn cluster documentation*
→ *UMAP documentation + examples*

■ *Use t-SNE/UMAP to visualise embeddings — it's a skill you'll use repeatedly in NLP and computer vision.*

■ Model Evaluation & Feature Engineering

- Evaluation metrics: Accuracy, Precision, Recall, F1, AUC-ROC, MAE, RMSE.
- Cross-validation: K-Fold, Stratified K-Fold, TimeSeriesSplit.
- Hyperparameter tuning: GridSearchCV, RandomizedSearchCV, Optuna.
- Feature engineering: encoding, scaling, imputation, interaction features.
- Handling imbalanced data: SMOTE, class weights, threshold tuning.

Resources:

→ *Kaggle competitions (practice ground)*
→ *Feature Engineering for Machine Learning — O'Reilly*

■ *Enter a Kaggle competition at this stage. Even finishing in the bottom half teaches you more than any course.*

■ Phase Outcomes

- Can train, evaluate, and tune classical ML models end-to-end
- Knows when to use which algorithm and why
- Comfortable with the full sklearn Pipeline pattern

03 Phase 3 — Deep Learning Fundamentals

Duration: 8 – 12 weeks | Difficulty: Intermediate

Enter the world of neural networks. This phase takes you from perceptrons to convolutional networks, building the foundation for modern AI.

■ Neural Network Basics

- Perceptron, activation functions (ReLU, Sigmoid, Tanh, GELU).
- Forward pass, backpropagation, chain rule in practice.
- Loss functions: MSE for regression, Cross-Entropy for classification.
- Optimisers: SGD, Adam, AdamW — learning rate schedules.
- Regularisation: Dropout, Batch Normalisation, Early Stopping, L1/L2.

Resources:

→ *Neural Networks: Zero to Hero — Andrej Karpathy (YouTube)*
→ *Deep Learning — Ian Goodfellow (free online)*
→ *fast.ai Part 1*

■ *Build a neural network from scratch in NumPy first. Then use PyTorch — you'll appreciate the abstraction far more.*

■■ Convolutional Neural Networks (CNNs)

- Convolution, pooling, stride, padding — the building blocks.
- Classic architectures: LeNet, AlexNet, VGG, ResNet, EfficientNet.
- Transfer learning: fine-tuning pretrained models (ImageNet weights).
- Applications: image classification, object detection (YOLO), segmentation.

Resources:

→ *CS231n — Stanford (free lecture notes)*
→ *PyTorch vision tutorials*
→ *timm library (pretrained models)*

■ *Use a pretrained ResNet or EfficientNet for your first project — training from scratch needs massive data and compute.*

■■ PyTorch / TensorFlow Framework

Choose one framework and master it deeply. PyTorch is recommended for research and production in 2024+.

- Tensors, autograd, computational graphs.
- nn.Module — building custom layers and models.
- DataLoader, Dataset — efficient data pipelines.
- Training loops, checkpointing, mixed precision (AMP).

Resources:

→ *PyTorch official tutorials (pytorch.org)*
→ *Lightning AI — PyTorch Lightning*
→ *Hugging Face Accelerate*

■ *Learn to use a GPU on Google Colab (free) or Kaggle Notebooks. GPU training is 10–100x faster for deep learning.*

■ Phase Outcomes

- Can design, train, and debug neural networks in PyTorch
- Understands backpropagation mathematically
- Can apply transfer learning to image tasks with minimal data

04 Phase 4 — Natural Language Processing (NLP)

Duration: 8 – 10 weeks | Difficulty: Intermediate

Language is where AI has seen the most dramatic advances. This phase covers the evolution from bag-of-words to Large Language Models.

■ Classical NLP

- Text preprocessing: tokenisation, stemming, lemmatisation, stopwords.
- Representations: Bag-of-Words, TF-IDF, N-grams.
- Word embeddings: Word2Vec, GloVe, FastText — dense semantic representations.
- Tasks: text classification, sentiment analysis, named entity recognition.

Resources:

→ *NLTK documentation*
→ *spaCy course (free)*
→ *Speech and Language Processing — Jurafsky & Martin*

■ *Always benchmark TF-IDF + Logistic Regression before reaching for a transformer — it often performs surprisingly well.*

■ Sequence Models & Transformers

- RNN, LSTM, GRU — recurrent architectures and their limitations.
- Attention mechanism — the key innovation behind modern NLP.
- **Transformer architecture:** self-attention, multi-head attention, positional encoding.
- BERT (encoder) vs GPT (decoder) — when to use each.

Resources:

→ *Attention Is All You Need (original paper)*
→ *The Illustrated Transformer — Jay Alammar (blog)*
→ *Hugging Face NLP Course (free)*

■ *Read 'The Illustrated BERT' by Jay Alammar. It's the clearest visual explanation of transformers available.*

■ Hugging Face & LLM Ecosystem

- Hugging Face Transformers library — loading and fine-tuning pretrained models.
- Fine-tuning BERT/RoBERTa for classification, NER, Q&A.;
- Working with LLMs: prompt engineering, few-shot learning, RAG basics.
- Datasets library, Tokenizers, Model Hub.

Resources:

→ *huggingface.co/learn (free NLP course)*
→ *LangChain documentation*
→ *OpenAI Cookbook (GitHub)*

■ *Fine-tune a small model (DistilBERT) on a custom text classification task. It demonstrates LLM skills concretely to employers.*

■ Phase Outcomes

- Can fine-tune transformer models for NLP tasks
- Understands the Transformer architecture in depth
- Can build a basic RAG pipeline or chatbot

05 Phase 5 — Computer Vision

Duration: 6 – 8 weeks | Difficulty: Intermediate

Vision is the other major pillar of applied AI. This phase covers modern computer vision from classification to generative models.

■■ Object Detection & Segmentation

- Detection architectures: YOLO (v8/v10), DETR, Faster R-CNN.
- Instance segmentation: Mask R-CNN, SAM (Segment Anything Model).
- Semantic segmentation: U-Net, DeepLab.
- Evaluation metrics: mAP, IoU, precision-recall curves.

Resources:

→ *Ultralytics YOLO documentation*
→ *Roboflow tutorials*
→ *MMDetection framework*

■ *Roboflow provides free tools for labelling, augmenting, and training object detection models — perfect for projects.*

■ Generative Models

- **GANs:** Generator vs Discriminator training, mode collapse, evaluation (FID score).
- **VAEs:** Latent space learning, reconstruction vs KL loss.
- **Diffusion Models:** Denoising score matching, DDPM, Stable Diffusion architecture.
- **Vision Transformers (ViT):** Image patches as tokens, CLIP, DINO.

Resources:

→ *Stable Diffusion deep-dive — Hugging Face blog*
→ *GAN Lab (interactive demo)*
→ *Lilian Weng's blog (lilianweng.github.io)*

■ *Experiment with Stable Diffusion locally or via Replicate API. Understanding diffusion models is increasingly required in industry.*

■ Phase Outcomes

- Can train and deploy object detection models
- Understands generative model trade-offs (GAN vs Diffusion)
- Can apply ViT / CLIP for zero-shot image tasks

06 Phase 6 — MLOps & Production AI

Duration: 6 – 8 weeks | Difficulty: Intermediate → Advanced

Building a model is only 20% of the job. This phase covers everything needed to deploy, monitor, and maintain AI systems in production reliably.

■ Model Deployment

- REST APIs: FastAPI / Flask — serving model predictions as endpoints.
- Containerisation: Docker — packaging models with all dependencies.
- Cloud deployment: AWS SageMaker, GCP Vertex AI, Azure ML, Hugging Face Spaces.
- Model serialisation: ONNX, TorchScript, SavedModel — framework-agnostic formats.
- Optimisation: quantisation (INT8), pruning, TensorRT for low-latency inference.

Resources:

→ *FastAPI documentation*
→ *Docker official get-started guide*
→ *AWS SageMaker Developer Guide*

■ *Deploy at least one model as a public API (Hugging Face Spaces is free). A live demo URL impresses employers far more than a notebook.*

■ Experiment Tracking & ML Pipelines

- Experiment tracking: MLflow, Weights & Biases (W&B;) — log metrics, params, artifacts.
- Data versioning: DVC — track datasets and models like code.
- ML Pipelines: Kubeflow, ZenML, Prefect, Apache Airflow — orchestrate training workflows.
- Feature Stores: Feast, Tecton — serving consistent features for training and inference.

Resources:

→ *MLflow quickstart*
→ *W&B; documentation*
→ *ZenML documentation*

■ *Use W&B; for every experiment — the free tier is generous and the visualisation tools will save you hours.*

■ Monitoring & Responsible AI

- Model monitoring: data drift, concept drift, prediction distribution shifts.
- Tools: Evidently AI, Arize, Grafana + Prometheus.
- CI/CD for ML: automated retraining pipelines, model validation gates.
- Responsible AI: fairness auditing, bias detection, model explainability (SHAP, LIME).
- Security: adversarial robustness, model privacy, data governance.

Resources:

→ *Evidently AI documentation*
→ *SHAP documentation*
→ *Google PAIR — Responsible AI practices*

■ *Monitoring is the most neglected skill in ML interviews. Knowing Evidently AI alone puts you ahead of 80% of candidates.*

■ Phase Outcomes

- Can deploy a model as a containerised API on cloud
- Tracks experiments systematically with MLflow or W&B;
- Understands drift monitoring and CI/CD for ML

07 Phase 7 — Specialisation & Advanced Topics

Duration: Ongoing | Difficulty: Advanced

Choose one or two specialisations to go deep. The AI landscape rewards depth alongside breadth at the senior level.

■ Large Language Models (LLMs) & Generative AI

- LLM fine-tuning techniques: LoRA, QLoRA, PEFT — parameter-efficient fine-tuning.
- Retrieval-Augmented Generation (RAG): vector databases (Pinecone, Weaviate, ChromaDB).
- Agents & Tool use: LangChain, LlamaIndex, OpenAI Assistants API.
- LLM evaluation: BLEU, ROUGE, BERTScore, LLM-as-judge.
- Prompt engineering: chain-of-thought, few-shot, structured output.

Resources:

→ *LangChain documentation*
→ *LlamaIndex documentation*
→ *Hugging Face PEFT library*

■ *Build a RAG-powered Q&A; system over your own documents. It's the #1 most requested LLM project in industry right now.*

■ Reinforcement Learning

- Core concepts: Markov Decision Processes, policies, value functions, Bellman equation.
- Algorithms: Q-Learning, Deep Q-Network (DQN), Proximal Policy Optimisation (PPO).
- Libraries: Gymnasium (OpenAI Gym), Stable-Baselines3, RLlib.
- RLHF (Reinforcement Learning from Human Feedback) — how ChatGPT is trained.

Resources:

→ *Spinning Up in Deep RL — OpenAI*
→ *Stable-Baselines3 documentation*
→ *Deep RL Bootcamp — Berkeley (free)*

■ *Train an RL agent to play a simple game (CartPole, LunarLander). It makes RL concepts immediately tangible.*

■ Research & Staying Current

• Read AI papers: arXiv.org (cs.LG, cs.CL, cs.CV sections).
• Paper walkthroughs: Yannic Kilcher, AI Coffee Break (YouTube).
• Follow AI labs: OpenAI, Anthropic, Google DeepMind, Meta AI blogs.
• Replicate papers: implementing a paper from scratch is the fastest way to understand it deeply.
• Contribute to open source: Hugging Face, PyTorch, LangChain — builds portfolio and network.

Resources:

→ *Papers With Code (paperswithcode.com)*
→ *The Batch — deeplearning.ai newsletter*
→ *AI Alignment Forum*

■ *Set aside 2 hours per week to read one paper and discuss it with peers or in an online community. Consistency beats intensity.*

■ Phase Outcomes

• Can fine-tune LLMs with LoRA/QLoRA

• Builds production RAG pipelines

• Actively reads and replicates recent research

• Contributes to the AI community

0 ★

Final Advice — How to Stay on Track

The habits that separate those who finish from those who don't

■■ Learn consistently, not intensely

10 focused hours per week beats one 40-hour weekend sprint. Spaced repetition builds durable knowledge.

■■ Build projects at every phase

Theory without application fades. After each phase, build something — even a small Streamlit app. Projects = portfolio.

■ Join the community

Follow AI practitioners on X/Twitter, join Discord servers (Hugging Face, fast.ai), attend local or virtual ML meetups.

■ Track your projects publicly

Push code to GitHub, write short posts on LinkedIn or Medium about what you learn. It compounds into career opportunities.

■ Return and revise

AI evolves fast. Revisit Phase 1–2 concepts every 6 months. A strong foundation will always be more valuable than chasing trends.

■ Apply early, apply often

Start applying for junior/intern AI roles after Phase 3. Interviews reveal gaps better than any course. Fail fast, learn faster.