



Essential GitHub Repositories for AI

AI course, resources | About 70 repositories by area (frameworks, training, inference, agents, RAG, evaluation, data, education) with owner and what to learn from each; owners checked as of 2026, no star counts

[#Resources](#) [#GitHub](#) [#OpenSource](#) [#Code](#)

Why read code, and how

An apprentice in a master's workshop

Watch the master make one chair from start to finish

Visit the big factory to see the machines

Ask why a joint is cut that way

Make your own chair, then compare

Fix a small defect in the workshop

Learning from GitHub repositories

Read a small education repo end to end (nanoGPT)

Skim a production library (transformers, vLLM) for its design

Read the issue tracker and pull-request discussions

Re-implement, then diff against the reference

Send a pull request on a 'good first issue'

WHERE THE ANALOGY BREAKS

A workshop does not change every week; popular AI repos change daily, so pin a release or commit when you study one.

Reading reference repositories is how most practitioners learn the details papers leave out

Prerequisites and conventions

Git basics (clone, branch, pull request) and Python. Every entry is owner/repo as on GitHub as of 2026; where a project moved, the current owner is given and GitHub redirects the old path. Licences given only when certain. How to use the libraries is taught in frameworks-and-libraries; here you get what to read and what it teaches.

★ Ten repositories to know first

pytorch /
pytorch

Python, C++ BSD-3-Clause

The dominant deep-learning framework: tensors, autograd, torch.compile, distributed training

```
pip install torch
```

github.com/pytorch/pytorch

huggingface /
transformers

Python Apache-2.0

Model definitions and pretrained weights for most open models; one file per architecture

```
pip install transformers
```

github.com/huggingface/transformers

vllm-project /
vllm

Python, CUDA Apache-2.0

High-throughput LLM serving: PagedAttention, continuous batching, OpenAI-compatible server

```
pip install vllm
```

github.com/vllm-project/vllm

ggml-org /
llama.cpp

C++ MIT

LLM inference on CPUs, Apple silicon and GPUs with GGUF quantised weights

```
brew install llama.cpp
```

github.com/ggml-org/llama.cpp

EleutherAI /
lm-evaluation-harness

Python MIT

Standard runner for hundreds of LLM benchmarks; same prompts for every model

```
pip install lm-eval
```

github.com/EleutherAI/lm-evaluation-harness

Foundations: the deep-learning framework, the model library, the two inference engines most guides assume, and the evaluation harness behind many leaderboards

karpathy / nanoGPT

Python

MIT

GPT-2 training in about 600 lines (model.py + train.py); reproduces GPT-2 124M

git clone https://github.com/karpathy/nanoGPT

github.com/karpathy/nanoGPT

karpathy / llm.c

C, CUDA

MIT

GPT-2 training in raw C and CUDA, no framework: see every kernel

git clone https://github.com/karpathy/llm.c

github.com/karpathy/llm.c

karpathy / micrograd

Python

MIT

A scalar autograd engine and tiny neural net library: backprop in about 150 lines

git clone https://github.com/karpathy/micrograd

github.com/karpathy/micrograd

rasbt / LLMs-from-scratch

Python

Code of the book Build a Large Language Model (From Scratch): GPT, pretraining, fine-tuning

git clone https://github.com/rasbt/LLMs-from-scratch

github.com/rasbt/LLMs-from-scratch

labmlai / annotated_deep_learning_paper_implementations

Python

Paper implementations with side-by-side notes: Transformers, diffusion, RL, optimisers

pip install labml-nn

github.com/labmlai/annotated_deep_learning_paper_implementations

Education: small repos written to teach. Read them line by line; each fits in an afternoon to a week

Frameworks and training

Repository (owner)	Area	What to learn from it
pytorch (pytorch)	framework	autograd engine, torch.compile, FSDP; read the tutorials repo too
jax (jax-ml)	framework	functional transforms grad, jit, vmap, pmap; moved from google/jax in 2024
flax (google)	framework	neural-network layer library for JAX
keras (keras-team)	framework	Keras 3 runs on JAX, PyTorch or TensorFlow backends
scikit-learn (scikit-learn)	classic ML	the fit / predict / transform API every ML library copies
xgboost (dmlc), LightGBM (microsoft), catboost (catboost)	boosting	histogram tree building, the tabular-data workhorses
tinygrad (tinygrad)	framework	a minimal framework with lazy tensors and its own compiler: readable
Megatron-LM (NVIDIA)	training	tensor and pipeline parallelism at thousands of GPUs
DeepSpeed (deepspeedai, formerly microsoft)	training	ZeRO stages 1-3 and offload: optimizer-state sharding
torchtitant (pytorch)	training	PyTorch-native large-scale pretraining with FSDP2, TP, PP
nanotron (huggingface)	training	minimal 3-D parallel pretraining; companion of the Ultra-Scale Playbook
accelerate, peft, trl (huggingface)	fine-tuning	device placement; LoRA and adapters; SFT, DPO, GRPO trainers
unsloth (unslothai)	fine-tuning	fast, memory-light LoRA / QLoRA on one GPU with custom kernels
axolotl (axolotl-ai-cloud)	fine-tuning	YAML-configured fine-tuning recipes across many models
LLaMA-Factory (hiyouga)	fine-tuning	fine-tuning many open models from a web UI or CLI

verl (volcengine), OpenRLHF (Open-RLHF)

RL for LLMs

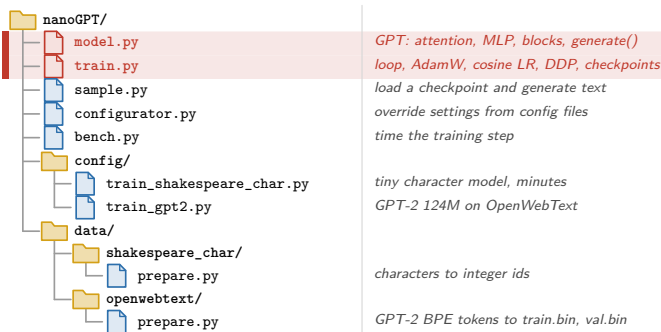
PPO and GRPO training loops for reasoning models

flash-attention (Dao-AI Lab)
triton (triton-lang)

kernels
kernels

IO-aware attention kernels used by most training stacks
write GPU kernels in Python-like code; read the tutorials

Frameworks: read the design docs and one core module; training: read the parallelism and checkpoint code



Anatomy of karpathy/nanoGPT: the whole GPT-2 recipe in a handful of files. Highlighted: read these two first

⚡ Inference, agents and RAG

Repository (owner)	Area	What to learn from it
vllm (vllm-project)	serving	PagedAttention block tables, scheduler, continuous batching
sglang (sgl-project)	serving	RadixAttention prefix caching, structured outputs
TensorRT-LLM (NVIDIA)	serving	compiled engines, in-flight batching, FP8 on NVIDIA GPUs
text-generation-inference (hugging-face)	serving	Rust router plus model shards; Hugging Face's server
llama.cpp, whisper.cpp (ggml-org)	local	GGUF format, k-quants, CPU and Metal kernels
ollama (ollama)	local	pull-and-run model manager built on llama.cpp; local REST API
mlx, mlx-lm (ml-explore)	local	Apple's array framework for Apple-silicon unified memory
onnxruntime (microsoft)	runtime	cross-platform inference for ONNX graphs, many hardware back ends
langchain, langgraph (langchain-ai)	agents	chains and integrations; graph-based stateful agents
llama_index (run-llama)	RAG	document loaders, indexes and query engines
dspy (stanfordnlp)	LLM programs	declare signatures, let optimisers tune prompts and few-shot examples
autogen (microsoft)	multi-agent	conversational multi-agent patterns
smolagents (huggingface)	agents	code-acting agents in a small, readable codebase
openai-agents-python (openai)	agents	agents, handoffs, guardrails, tracing
pydantic-ai (pydantic)	agents	typed tools and structured outputs with Pydantic models
servers, python-sdk (modelcontextprotocol)	MCP	reference MCP servers and the official Python SDK
haystack (deepset-ai)	RAG	pipeline components for search and RAG
faiss (facebookresearch)	vector search	IVF, PQ, HNSW indexes on CPU and GPU
qdrant (qdrant), milvus (milvus-io), chroma (chroma-core), pgvector (pgvector)	vector DB	storage plus ANN index behind RAG; compare in mlops 06
sentence-transformers (UKPLab, maintained with Hugging Face)	embeddings	train and use embedding and reranker models
ColBERT (stanford-futuredata)	retrieval	late-interaction retrieval, token-level matching

✓ Evaluation, data and more education repos

Repository (owner)	Area	What to learn from it
lm-evaluation-harness (EleutherAI)	eval	task YAML format, log-likelihood vs generation scoring
lighteval (huggingface)	eval	fast LLM evaluation with many backends
helm (stanford-crfm)	eval	holistic evaluation: many metrics per scenario
inspect_ai (UKGovernmentBEIS)	eval	UK AI Security Institute's framework for agentic and safety evals
mteb (embeddings-benchmark)	eval	embedding benchmark across retrieval, clustering, classification
promptfoo (promptfoo), ragas (explodinggradients)	app eval	prompt regression tests; RAG faithfulness and relevance metrics
datasets, datatrove (huggingface)	data	Arrow-backed datasets; web-scale filtering and dedup pipelines
ray (ray-project)	data, compute	distributed data processing, training and serving in Python
label-studio (HumanSignal), argilla (argilla-io)	labelling	annotation UIs; human feedback datasets
cleanlab (cleanlab)	data quality	find label errors with confident learning
mlflow (mlflow)	tracking	experiment tracking, model registry, LLM tracing
minGPT, makemore, minbpe, nn-zero-to-hero (karpathy)	education	GPT in PyTorch; character models; BPE; lecture notebooks
nanochat (karpathy)	education	2025 end-to-end small ChatGPT-style pipeline: tokenizer to chat UI
d2l-en (d2l-ai)	education	code of Dive into Deep Learning, runnable notebooks
fastbook, fastai (fastai)	education	notebooks of the fast.ai book; the fastai library
llm-course (mlabonne)	education	LLM roadmap with notebooks: fundamentals, scientist, engineer
ML-For-Beginners, generative-ai-for-beginners (microsoft)	education	structured multi-week lesson series
openai-cookbook (openai)	education	API usage examples: embeddings, function calling, evals
courses (anthropics)	education	Anthropic's prompt-engineering and API courses as notebooks
pytorch-image-models (huggingface)	vision	timm: hundreds of vision backbones with training scripts
ultralytics (ultralytics), segment-anything, sam2 (facebookresearch)	vision	YOLO training and export; promptable segmentation
Gymnasium (Farama-Foundation), stable-baselines3 (DLR-RM), cleanrl (vwxyzjn)	RL	environments API; reliable baselines; single-file RL algorithms

Category	Read to understand	Use in production
Train a Transformer	- karpathy/nanoGPT - rasbt/LLMs-from-scratch	- pytorch/torchdnn - NVIDIA/Megatron-LM
Fine-tune an LLM	huggingface/trl examples	- unslothai/unsloth - axolotl
Serve an LLM	ggml-org/llama.cpp server	- vllm-project/vllm - sgl-project/sglang
Build an agent	huggingface/smolagents	- langchain-ai/langgraph - openai-agents-python
Evaluate	EleutherAI/lm-evaluation-harness tasks	- UKGovernmentBEIS/inspect_ai - promptfoo

Where to start per goal: one small repo to understand, one production repo to use

🚩 Worked example, myths and recap

PROBLEM	nanoGPT's train_gpt2.py config trains GPT-2 124M with batch 12, block 1024, gradient accumulation 5×8 , for 600,000 iterations. Karpathy's 2024 llm.c run trained the same size on 10 billion FineWeb tokens. How many tokens does nanoGPT use, and how many training FLOPs?
GIVEN	- Tokens per iteration $T_{it} = \text{batch} \times \text{block} \times \text{accumulation steps (tokens)}$ - Total tokens $D = T_{it} \times \text{iterations}$ - Training compute $C \approx 6 \times N \times D$ FLOPs, $N = \text{parameters (124 million)}$, $D = \text{tokens}$

- 1 Tokens per iteration**
 $12 \times 1024 \times 40 = 491,520 \approx 0.5 \text{ M tokens}$
Why: the config's comment calls it a ~0.5M-token batch
- 2 Total tokens**
 $491,520 \times 600,000 \approx 2.95e11 \approx 300 \text{ B tokens}$
Why: the config targets 300B tokens
- 3 Training FLOPs**
 $6 \times 1.24e8 \times 2.95e11 \approx 2.2e20 \text{ FLOPs}$
Why: 2 FLOPs forward and 4 backward per parameter per token
- 4 Compare with llm.c**
 $2.95e11 / 1e10 \approx 30 \times \text{more tokens}$; $6 \times 1.24e8 \times 1e10 \approx 7.4e18 \text{ FLOPs}$
Why: most of the gap is data, not code speed

ANSWER	About 300 billion tokens and 2.2×10^{20} FLOPs for nanoGPT, about 30 times the llm.c run's data and compute
CHECK	✓ $300e9 / 124e6 \approx 2,400$ tokens per parameter: far past the Chinchilla 20, which is why a 10B-token run already matches GPT-2's quality

Why llm.c's GPT-2 run was so much shorter than nanoGPT's: count the tokens before comparing hardware

MYTH	FACT
✗ The biggest framework is the best place to learn	✓ Start with education repos transformers spans thousands of files; nanoGPT's model.py is about 300 lines with the same ideas.
✗ An old GitHub URL that redirects is fine to cite	✓ Cite the current owner Redirects break if a new repo takes the old name; jax-ml/jax and ggml-org/llama.cpp are current.
✗ Open source means free to use in any product	✓ Read the licence MIT and Apache-2.0 allow commercial use; some repos and most model weights use custom terms.
✗ Forks with more recent commits replace the original	✓ The owner's repo is the reference Use forks only when the original is archived and the fork is widely adopted.

Remember

Tick a box only when you could explain the point to someone else



Read small, use big

nanoGPT to learn, vLLM or Megatron to run



Owner/repo is the identity

check moves: jax-ml, ggml-org, deepspeedai



Read issues and PRs

they explain the design choices



Pin a release

main changes daily



Check the licence

code, data and weights differ

Q1 Which organisation owns llama.cpp as of 2026?

A ggml-org (moved from Georgi Gerganov's personal account in 2025).

Q2 Which two files hold nanoGPT's model and training loop?

A model.py and train.py.

Q3 Which repo trains GPT-2 in raw C and CUDA?

A karpathy/llm.c.

Q4 Where does JAX live now?

A jax-ml/jax (moved from google/jax in 2024).

Q5 Name two LLM serving engines and their key idea.

A vLLM: PagedAttention; SGLang: RadixAttention prefix caching.

Q6 Which harness runs most open LLM benchmarks?

A EleutherAI/lm-evaluation-harness.

Q7 Which Hugging Face repos cover LoRA and DPO training?

A huggingface/peft (adapters) and huggingface/trl (SFT, DPO, GRPO).

Q8 What does Megatron-LM teach?

A Tensor and pipeline parallelism for training at very large scale.

Q9 Which repo implements ZeRO optimizer-state sharding?

A DeepSpeed (deepspeedai, formerly microsoft).

