Founders and Coders

LEVEL 6 APPRENTICESHIP · OLD STREET, LONDON

A place for curious minds to learn frontier AI

Advanced in-person training in deep learning and agentic engineering, for experienced software engineers.

The programme

We teach one programme, a Level 6 machine learning engineer apprenticeship for experienced software engineers. Workshops run in person near Old Street, one afternoon a week, from 2pm to 6pm. The next cohort starts on 4 March 2027.

FORMAT	WHERE	LENGTH	COST
One afternoon a week	**Old Street, London**	**58 workshops, about 15 months**	**Fully funded by the levy**

We bring together experienced engineers from different industries and work through the modern deep learning and agentic engineering stack. You come in one afternoon a week, for 58 workshops. With breaks, that takes about 15 months. The programme runs in three terms, with breaks at Christmas, in the spring and at the end of the summer, and the week numbers count teaching weeks only. If you want to move faster, you can finish after 12 months.

The work is meant to be used. Four of the seven modules are projects, and if your employer has a problem that fits, you work on that. In the last module you build an AI system for your employer and put it into service.

The first thing you write is a coding agent, from scratch. We give you a server, and you turn it into a Kubernetes cluster and serve the models you train from it for the rest of the programme. You build retrieval with hybrid search, reranking, and an agent that searches for itself. Later you build a transformer from the ground up in PyTorch. In Frontier ML each afternoon goes to one paper, read in the room and reproduced at small scale.

We provide a server, GPU access, and OpenAI, Claude and OpenRouter credits for the workshops. Practitioners lead them, and speakers come in from organisations such as Google DeepMind.

How we teach

What we're trying to build

We're trying to build the best programme of its kind: a place where experienced software engineers develop a technical understanding of deep learning and agentic engineering, and use it to lead AI projects at work.

Who's this programme for?

Founders and Coders is for you if you are kind, curious, and ready to build. You'll be learning alongside experienced engineers. Most people who join have written code for a living, whether for a few years or a few decades. If you haven't, you'll need to show that you can build working software from scratch, and be prepared to put the hours in to cover the fundamentals.

What you'll build

You'll spend an afternoon each week programming at the edge of your abilities. You'll write an agent harness from scratch, turn a bare server into a Kubernetes cluster and serve your own models from it, build a transformer from the ground up in PyTorch, and reproduce frontier papers.

What an afternoon looks like

Each session opens with a talk. Then you break into groups and work through a set of problems at the whiteboard. The middle of the afternoon is for building, or for focused study. Each session ends with a lightning talk and discussion. One member of the community shares a new tool or technique, something that did or didn't work in production, or a problem they're stuck on.

In person

All training is in person at our Old Street campus, one afternoon a week for about 15 months. Online content is everywhere and often excellent, but we believe you learn faster when you're surrounded by people trying to get good at the same thing.

Fully funded

There is no cost to you or your employer. Each place costs £22,000 and is funded through the apprenticeship levy. If your employer does not pay the levy, we'll arrange funding. You need to be employed on PAYE and your employer needs to be on board.

We train engineers from

ZOOPLA ticketmaster sky UNIVERSITY OF CAMBRIDGE ROLLS ROYCE FT wellcome Queen Mary University of London

WHAT APPRENTICES SAY

Jon Besga
Neo4j

AI Engineer

The programme gave me a practical, hands-on understanding of machine learning and AI through real projects rather than passive lectures. By the end, I felt significantly more prepared to join an AI engineering team, and I did.

Anvar Abdullaev
xAI

AI Finance Expert

The apprenticeship helped me bridge my existing expertise with applied AI. Shortly after completing it, I joined xAI as an AI Finance Expert, working on AI reasoning and evaluation in finance.

Amber Rignell
Kestrix

Product Engineer

Get the right group of people together and give them the challenge of figuring out machine learning, and something remarkable happens.

Serena Chan
Sony

ML Engineer

We had opportunities to build models from scratch, which helped me gain a deeper understanding of concepts that are often abstracted away. Fantastic preparation for a role in AI engineering.

The programme at a glance

Seven modules and nine standalone workshops, one afternoon a week. The week numbers count teaching weeks only, with breaks at Christmas, in the spring and at the end of the summer.

STANDALONE WORKSHOPS

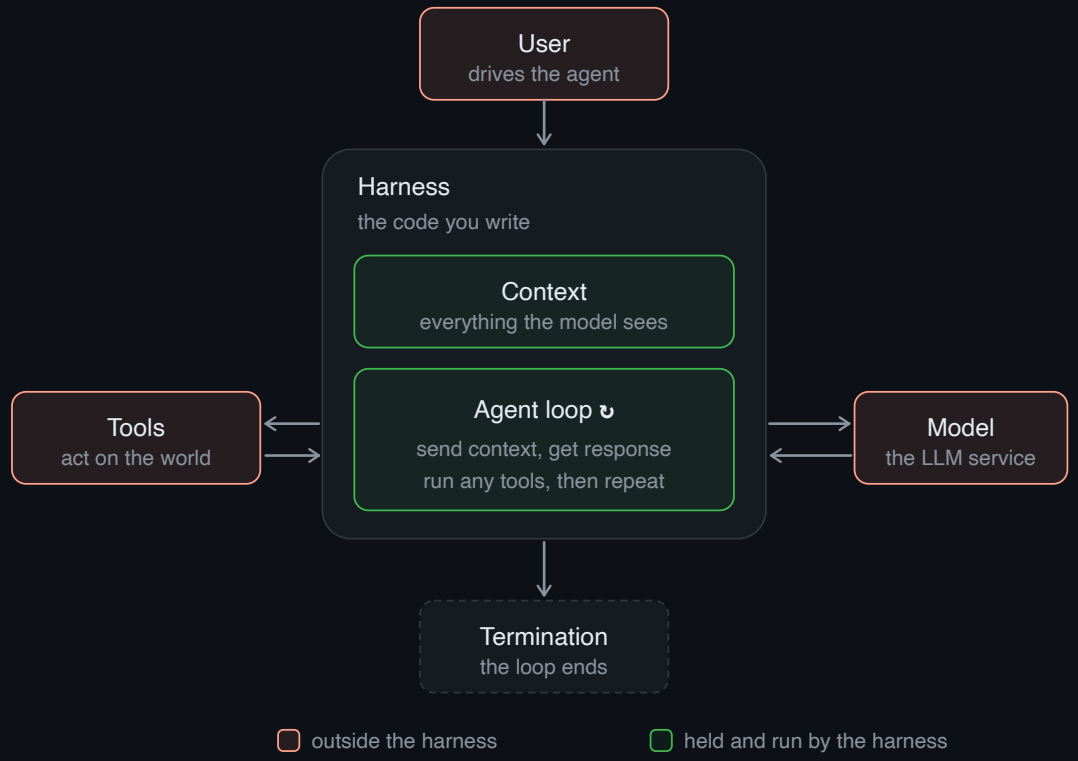
S1	Programme launch	Week 1	S2	Evals and error analysis	Week 9	S3	Your own cluster	Week 10
S4	Red-teaming agents	Week 17	S5	Maths refresher	Week 25	S6	A voice agent	Week 32
S7	Security and responsible AI	Week 39	S8	A GPU kernel	Week 40	S9	AI safety and risk	Week 48

MODULE 1

Agent harnesses

Weeks 2 to 8 · seven workshops

An agent is a language model using tools in a loop. The harness is everything in that loop except the model, from the context it is given to the checks that run on what it produces.



Python · Claude Code · MCP · Docker

MODULE 1

Agent harnesses

In the first workshop you write a coding agent from scratch, a small working version of Claude Code. In the second you look at harness engineering, and at the controls that improve what Claude Code produces. One short session goes to the Model Context Protocol. The rest of the module is about building your own agent harness.

WHAT YOU WILL BUILD

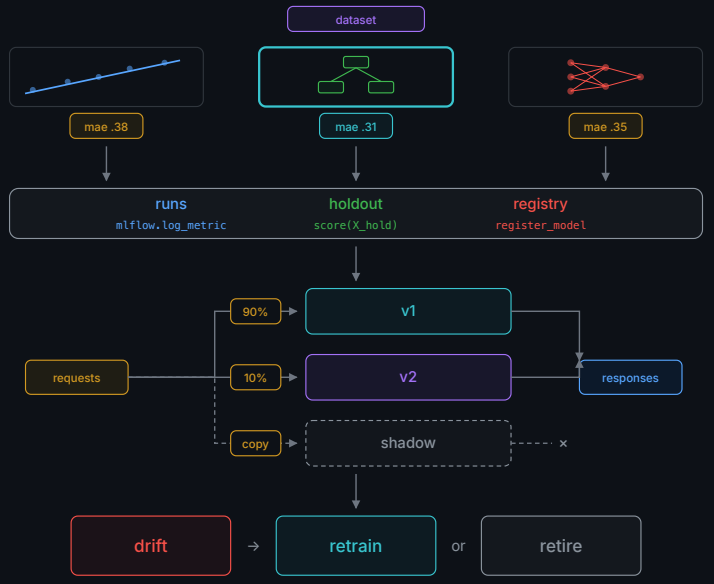
- A coding agent written from scratch, a small working version of Claude Code
- An improved version of the coding harness and workflow you already use
- A harness of your own, whether built from scratch, wrapped round the Claude Agent SDK, or written as conventions for coding agents at your employer
- Part of Terminal-Bench run against your harness, in Docker

MODULE 2

Production ML

Weeks 11 to 16 · six workshops

You start with raw data and end with a model answering live requests. The aim is to do a machine learning engineer's job once, end to end.




```python
for model in (linear, trees, net):
    with mlflow.start_run():
        mlflow.log_metric("mae", score(model, X_hold, y_hold))
best = mlflow.register_model(f"runs:/{best_run}/model", "churn")
deploy("churn", best.version, traffic=0.9)
deploy("churn", candidate, traffic=0.1)
deploy("churn", shadow_version, mode="shadow")
retrain() if psi(train_ref, live_window) > 0.2 else retire()
```

Python · scikit-learn · XGBoost · PyTorch · MLflow · FastAPI · Docker · Kubernetes

MODULE 2

Production ML

You curate the data, then fit a linear regression baseline, gradient-boosted trees and a small neural network, score all three on the same held-out split, and log every run to MLflow.

You serve the model behind FastAPI on your own Kubernetes cluster, then change it while it is running, with a second version on 10% of the traffic and a third as a shadow on a copy of every request. In the last workshop you look at data drift, and at whether to retrain the model or retire it.

WHAT YOU WILL BUILD

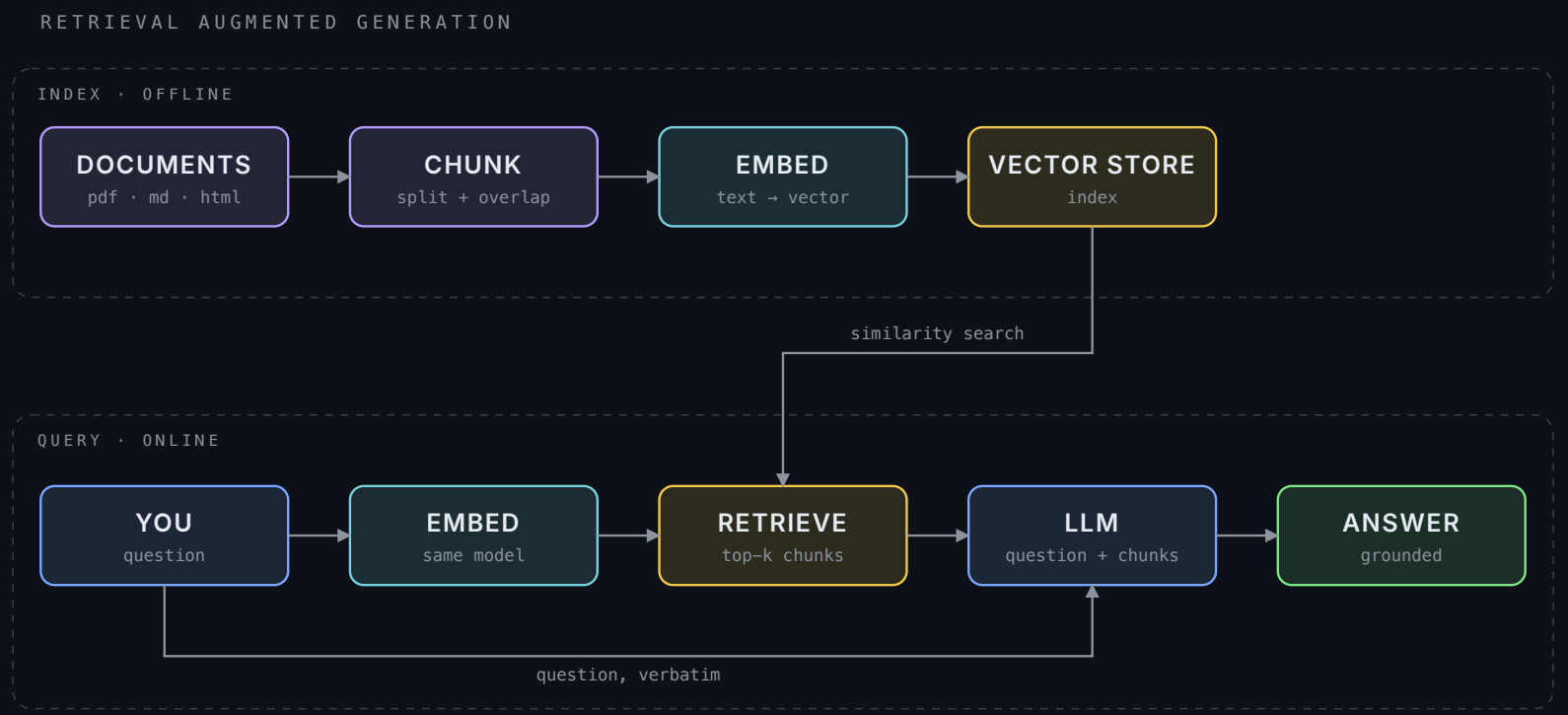
- A linear regression baseline, gradient-boosted trees and a small neural network, scored on the same held-out split
- An MLflow run log, with one model promoted to the registry
- The model served behind FastAPI on your own Kubernetes cluster
- A second version on 10% of the traffic, and a third running as a shadow
- Drift detection on the live data, and a decision to retrain or retire

MODULE 3

Retrieval and context engineering

Weeks 18 to 24 · seven workshops

Context engineering is the job of getting the right data in front of the model. Retrieval is how you do that when the corpus is too large to load.



Python NumPy Hugging Face Postgres

MODULE 3

Retrieval and context engineering

You start with RAG as it was built in 2023. Documents are chunked, embedded and stored in pgvector. A query is embedded, the nearest chunks come back, and they go into the prompt. You build it first on a hosted embedding API, then on a local model, then strip it down to cosine search in NumPy. You write the test questions before any of that, part by hand and part generated, check the generated ones against human judgement, and score every change on precision, recall and mean reciprocal rank.

Then you move on to what the field does now. A BM25 index sits beside the dense one and the two are fused by reciprocal rank fusion. A cross-encoder reranks the shortlist before it reaches the model, and you measure what that costs in latency. After that the model takes over retrieval. An agent with search tools decides what to fetch, judges what came back and fetches again, and you score it against loading the whole corpus into the window. The last taught afternoon is LLM operations. You route requests between a cheap model and a reasoning model, cache what repeats, version the prompts, track what every call costs and add a fallback for when a provider fails.

The module ends with a group project, a retrieval system over a set of documents, ideally your employer’s.

MODULE 3

Retrieval and context engineering

WHAT YOU WILL BUILD

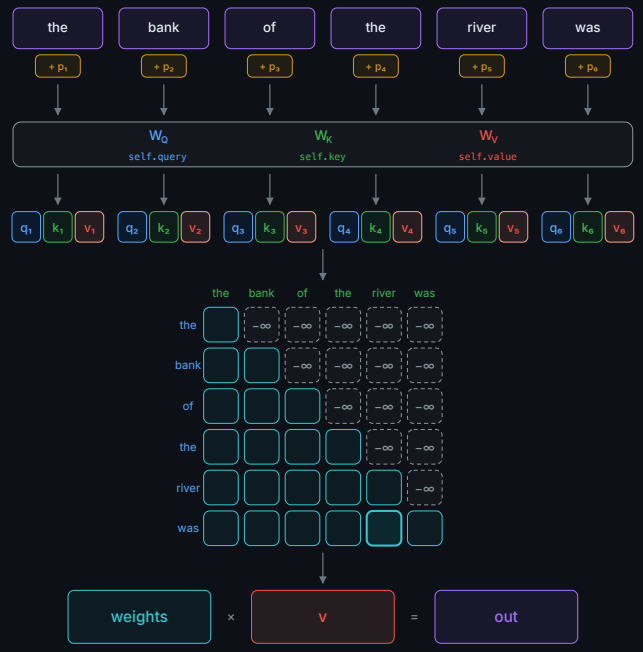
- A dense retriever built three ways, from hosted embeddings in pgvector to a local embedding model to cosine search in plain NumPy
- A BM25 index over the same documents, fused with the dense results by reciprocal rank fusion
- A cross-encoder reranker over the shortlist, and a measurement of the latency it adds
- A labelled query set written before the system, part hand-written and part synthetic, and a harness that reports precision, recall and MRR after every change
- An agent that searches the corpus itself, scored against loading everything into the window
- A routing layer between a cheap model and a reasoning model, with caching, prompt versioning, cost tracking and a fallback
- A retrieval system over a set of documents, ideally your employer's, built as a team

MODULE 4

Transformers from first principles

Weeks 26 to 31 · six workshops

You build a transformer from the ground up in PyTorch. The aim is to understand the substrate at the heart of the current AI boom.




```python
q = self.query(x)
k = self.key(x)
v = self.value(x)
scores = q @ k.transpose(-2, -1) / head_size ** 0.5
scores = scores.masked_fill(self.tril[:T, :T] == 0, float("-inf"))
weights = F.softmax(scores, dim=-1)
out = weights @ v
```

Python NumPy PyTorch Hugging Face

MODULE 4

Transformers from first principles

You read the foundational papers alongside the code, starting with ‘Attention Is All You Need’, and the engineering decisions in your model come from them. Attention comes first, worked out by hand on one sentence before it goes into code, then the rest of the block, then training on a small dataset. Towards the end of the module you train a model on a GPU with mixed precision and fine-tune a small open model with LoRA.

By the end of the module you have a clear idea of what deep learning researchers do day to day.

WHAT YOU WILL BUILD

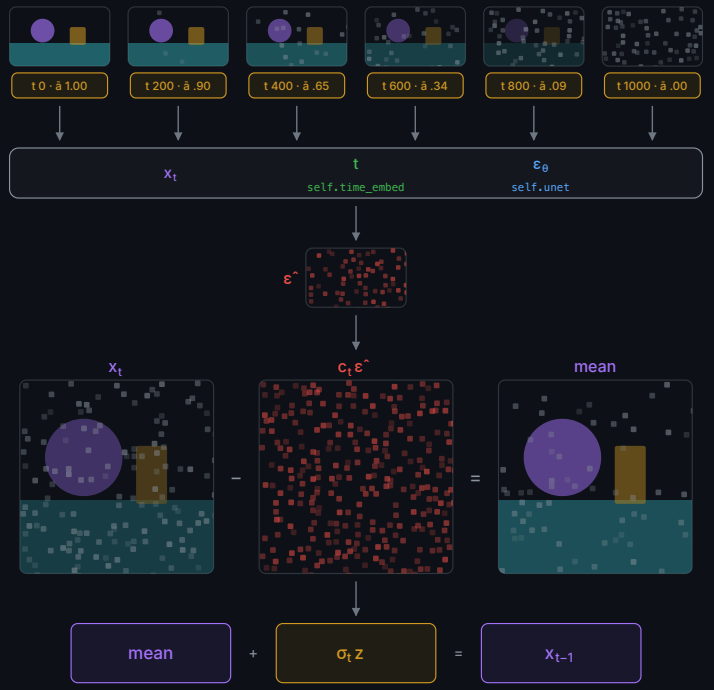
- Scaled dot-product attention, worked by hand on one sentence and then written in NumPy
- Multi-head self-attention and positional encodings in PyTorch
- A decoder-only transformer with causal masking, residual connections and layer normalisation, trained on a small dataset with cross-entropy loss
- Training moved to a GPU with mixed precision, and the throughput before and after
- A small open model fine-tuned with LoRA

MODULE 5

Frontier ML

Weeks 33 to 38 · six workshops

This module has no fixed syllabus. We choose the papers eight weeks before it starts, from whatever the field is arguing about at the time.




```python
eps = self.unet(x, self.time_embed(t))
a, a_bar = self.alphas[t], self.alphas_bar[t]
a_prev = self.alphas_bar_prev[t]
coef = (1 - a) / (1 - a_bar) ** 0.5
mean = (x - coef * eps) / a ** 0.5
sigma = ((1 - a_prev) / (1 - a_bar) * (1 - a)) ** 0.5
x = mean + sigma * torch.randn_like(x) if t > 0 else mean
```

Python PyTorch Hugging Face

MODULE 5

Frontier ML

Each afternoon goes to one paper. You read it in the room, argue over it at the whiteboard in groups of four, then reproduce part of it at small scale and take the implementation as far as the afternoon allows.

IF THE MODULE RAN TODAY

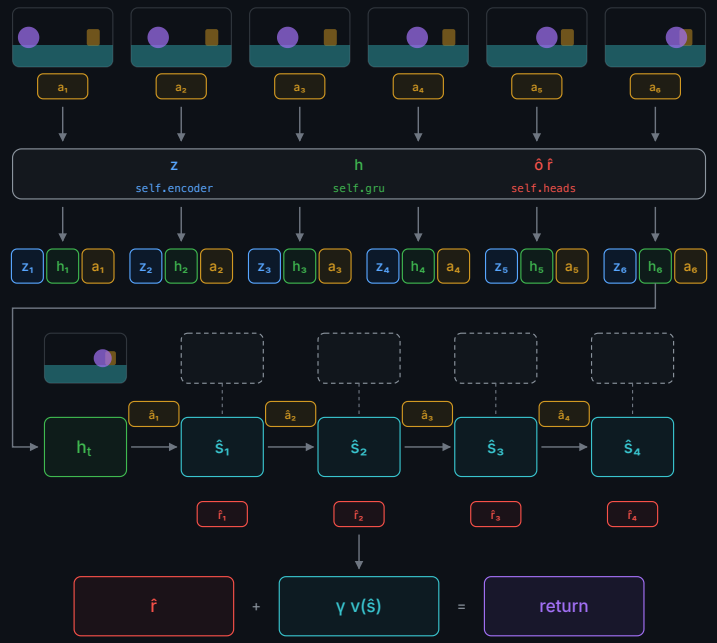
- Preference optimisation
- Reinforcement learning from verifiable rewards
- Diffusion models for images
- Diffusion models for text
- The cost of inference
- World models

MODULE 6

Research sprint

Weeks 41 to 47 · seven workshops

Your team picks a paper and a problem, ideally one set by an employer. You reproduce the paper's result at small scale, then apply it to the problem.




```python
z = self.encoder(obs)
h = self.gru(torch.cat([z, a], -1), h)
prior = self.dynamics(h)
post = self.posterior(h, z)
loss = recon(obs) + kl(post, prior)
for _ in range(H): h, s = self.imagine(h, self.actor(h))
ret = self.reward(s) + gamma * self.critic(s)
```

PyTorch · Hugging Face · Docker · Kubernetes

MODULE 6

Research sprint

The one rule is to serve early. By the second week the team has a model in service, on a cluster one of you already runs, and at the end you know what it cost.

WHAT YOU WILL BUILD

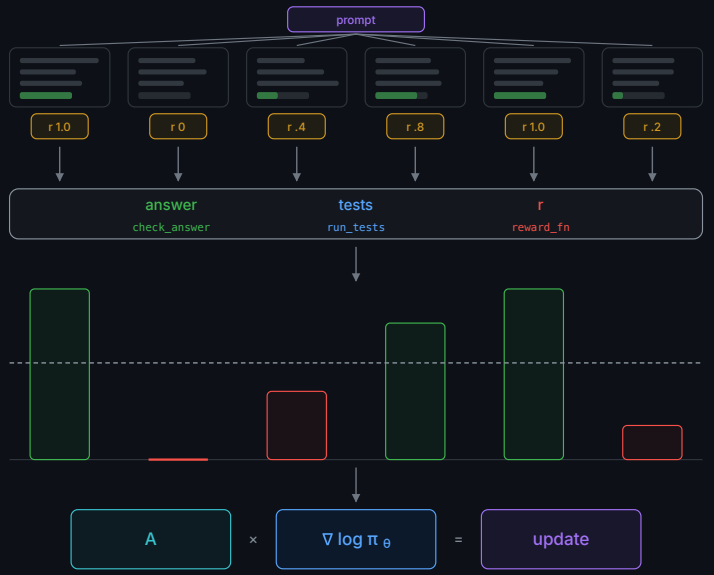
- A reproduction of a published result at small scale
- A model in service by the second week

MODULE 7

Employer project

Weeks 49 to 58 · ten workshops

You build an AI system for your employer, to a brief they set, and put it into service. Your assessment and grade are based on that work.




```python
out, old_logp = policy.generate(prompt, num_samples=G)
r = torch.tensor([verify(o, gold) for o in out])
adv = (r - r.mean()) / (r.std() + 1e-8)
logp, ref_logp = policy.log_probs(out), ref.log_probs(out)
ratio = (logp - old_logp).exp()
clipped = ratio.clamp(1 - clip_eps, 1 + clip_eps)
loss = -torch.min(ratio * adv, clipped * adv).mean()
loss += beta * kl(logp, ref_logp)
```

PyTorch MLflow Docker Kubernetes

MODULE 7

Employer project

The cohort reviews your work twice, first on the architecture and the data, then on the evaluation and the risks. You document the system, hand it over, and present it first to the cohort and then to your employer.

WHAT YOU WILL BUILD

- A brief agreed with your employer, naming the problem, the data and who it is for
- An AI system built on that brief and put into service
- Monitoring on the live system
- Documentation and a handover to your employer
- A written report and a presentation, which form the first part of your assessment

Standalone workshops

Programme launch
Week 1

The first afternoon starts with lunch. You meet the cohort, hear what the next 15 months hold, and set up your tools.

Evals and error analysis
Week 9

You read 100 traces from the agent you built and sort its failures by type. You then write an automated judge and measure how often it agrees with you.

Your own cluster
Week 10

We give you a server and you turn it into a single-node Kubernetes cluster with DNS, ingress and TLS certificates. It is yours for the rest of the programme, and you serve the models you train later from it.

Red-teaming agents
Week 17

An agent is open to attack when it holds private data, reads untrusted text and can send messages out. You attack each other's agents by prompt injection, jailbreaking and data extraction, then close the holes in your own with input validation and output filtering.

Maths refresher
Week 25

We revise the linear algebra, calculus and probability that the next module relies on.

A voice agent
Week 32

You build a voice agent, either on a real-time speech API or as a pipeline of speech recognition, a language model and speech synthesis.

Security and responsible AI
Week 39

Most of a machine learning system is other people's models, datasets and libraries. You look at what can be hidden in them, and you test a model for bias before you release it.

A GPU kernel
Week 40

You write a fused softmax kernel in Triton and profile it against the PyTorch version, then read the FlashAttention kernel line by line. Memory bandwidth limits these kernels more than arithmetic does.

AI safety and risk
Week 48

More capable systems bring new risks, including misuse and agents that act with too much autonomy. You read how labs evaluate a model for dangerous capabilities, and how they decide whether it is safe to release.

For employers

A levy-funded machine learning engineer apprenticeship for experienced software engineers and data scientists. One afternoon a week at our central London campus, with project work feeding back into your roadmap.

Build AI capability in your team

Your engineer develops a technical understanding of deep learning and agentic engineering, and learns to use it to lead AI projects at work. Over the programme they'll build agent harnesses, train a transformer from first principles, deploy their own models, and take machine learning projects from raw data to production.

Projects tied to your roadmap

The programme is designed to feed useful work back into your team. Where the curriculum overlaps with your roadmap, your engineer can apply the work to your data, systems and product problems. Their final project is set with you and built around a real problem for the business.

One afternoon a week

Training is in person at our Old Street campus, one afternoon a week for about 15 months. That afternoon is the only fixed time away from work. The rest of the week, your engineer is back with the team, applying what they're learning as they go.

Fully funded

There is no additional cost to your business. Each place costs £22,000 and is funded through the apprenticeship levy. If you do not pay the levy, we'll arrange funding.

Learning alongside experienced engineers

Your engineer learns alongside other experienced software engineers, spending an afternoon each week programming at the edge of their abilities. Sessions combine talks, whiteboard problems, building and focused study, and end with a lightning talk and discussion.

Rated Strong by Ofsted

Ofsted's most recent inspection rated us Strong in curriculum, teaching, and apprentice development.

Before you start, and how it ends

BEFORE YOU START

Maths

The Transformers from first principles module assumes linear algebra, calculus and probability. You study them at home, from modules on our learning platform, and we run one refresher workshop in week 25, the week before it starts.

Pre-course

We run a pre-course before the programme starts. Come to as many of those sessions as you can.

COST AND ASSESSMENT

Cost

Each place costs £22,000, and the government pays all of it through the apprenticeship levy. Larger employers have already paid for the training through their taxes. For smaller employers, our partners pay the full cost. We can explain the levy when you speak to us.

Assessment

The apprenticeship ends with an assessment in two parts. The first is a written report and a presentation on the system you built for your employer. The second is a discussion of your work with an independent assessor.

Founders and Coders

APPLY FOR THE MARCH 2027 COHORT

The first workshop is on 4 March 2027, from 2pm to 6pm.

Start your application at foundersandcoders.com/apply, or come to a taster workshop first. Employers can book a call with our CEO at foundersandcoders.com/employers.