< Go to the original

How to Learn LLMs From Scratch

A 5 Step Roadmap

Marina Wyss - Gratitude Driven

Follow

DSC Data Science Collective androidstudio ~5 min read · July 8, 2025 (Updated: July 8, 2025) ·
Free: No

Today's post is all about how to learn LLMs from scratch. Not how to use an API, but how to really understand what's happening under the hood — from the core math to architecture details, and all the way through to post-training and alignment.

We have five stages on the agenda.

By the end, you'll have a clear path to prototype your own small GPT-style model from scratch and to understand what it would take

Let's get started.

Stage #1: Foundational Math

I'm assuming you know very basic math and can code in Python, but that's it.

So the first thing we need to do is learn enough calculus, linear algebra, and probability to be able to follow the calculations.

If you're brand new, I recommend beginning with intuition. Watch the 3Blue1Brown Essence of Linear Algebra and Essence of Calculus series. These are visual, conceptual, and really helpful for grasping what the math is actually doing under the hood.

Then, once you have some intuition, move on to more structured learning:

- The DeepLearning.AI Math for Machine Learning specialization on Coursera is beginner-friendly and covers all the essential topics — calculus, linear algebra, and probability — at a gentle pace.
- After that, if you want to get better at doing the calculations by hand, follow up the DeepLearning.AI specialization with the first two courses from the Imperial College London Math for ML Specialization. These will give you a lot of practice calculating derivatives and matrix operations so you'll completely be able to follow along with the next courses.

By the way — **Coursera is running a big special right now: You can get 40% off an annual subscription to Coursera Plus**, which gives you free access to all their courses with a certificate for no extra

The promo is only running for the next couple of weeks, so definitely check it out!

Stage #2: Neural Networks

Now that the math is in place, we can move on to deep learning fundamentals.

As always, I like to start with visuals to build intuition. I highly recommend the first four videos in 3Blue1Brown's Neural Networks playlist, starting with: But what is a neural network? | Deep learning chapter 1

StatQuest also has a very accessible playlist on deep learning fundamentals that I'd recommend if you want a friendly overview of what's to come in the more advanced courses. Specifically, I'd watch videos 1–18 from his deep learning playlist: The Essential Main Ideas of Neural Networks

You can either watch the entire playlist back-to-back, or check for the StatQuest version of a topic before you watch the more formal course which I'll recommend in a bit. This kind of learning works well for me because I go into a more math and code-heavy course already knowing the core concepts.

Speaking of code implementation, of course I have to recommend the famous videos by Andrej Karpathy at this point. These videos are clear, step-by-step explanations of backpropagation and training neural networks: The spelled-out intro to neural networks and backpropagation: building micrograd core of LLMs!

But for those that want an even more solid understanding first, I'd recommend the Deep Learning specialization from DeepLearning.AI, and the book *Deep Learning* by Ian Goodfellow and colleagues. The Coursera specialization is a classic for a reason, and has gotten a lot of updates to stay current.

Stage #3: Transformers & Pre-training

After all this learning, you have the foundation required to understand the architecture powering modern LLMs: The transformer.

I'm sure this is going to be a big surprise, but I'm going to recommend starting with a high-level visual introduction. ;-)

Here are some good resources:

3blue1brown neural networks playlist episode 5 and onwards, and videos 19–22 of the StatQuest neural networks playlist.

You can also check out the blog Illustrated Transformer if you prefer reading.

One level up are the famous Andrej Karpathy tutorials. He has an excellent walkthrough of building GPT from scratch, and building the GPT tokenizer.

- Let's build GPT: from scratch, in code, spelled out.
- Let's build the GPT Tokenizer

If you need help understanding it, this video from Yannic Kilcher is highly recommended.

After all this, you understand pre-training, which is where a model is trained on a huge corpus to predict the next token. But modern LLMs go through a few more steps, so let's go onto those next.

Stage #4: Fine-tuning

The pre-trained model knows how language works, but it doesn't yet know how to specialize in anything useful. That's where fine-tuning comes in.

Fine-tuning takes a base model and trains it a bit more on a domain-specific dataset, so it can perform well in a specific domain like legal, medical, financial, or something else entirely.

To understand fine-tuning, I recommend starting with the short course Finetuning Large Language Models by DeepLearning.AI.

For more depth, check out the book *Natural Language Processing with Transformers* and the August 2024 technical review The Ultimate Guide to Fine-Tuning LLMs from Basics to Breakthroughs.

Like everything in this space, the techniques are advancing all the time. It's worth taking some time to understand newer more parameter-efficient fine-tuning methods like LoRA and QLoRA.

Stage #5: Alignment

Reinforcement Learning with Human Feedback is for — it teaches the model to generate responses humans actually prefer.

Here's a good progression for learning about RLHF:

First, if there's a StatQuest video on a topic, you know that's what I'm going to recommend. So let's start with this overview: Reinforcement Learning with Human Feedback (RLHF), Clearly Explained!!!

HuggingFace has a nice intro video as well: Reinforcement Learning from Human Feedback: From Zero to chatGPT

You should also check out *Spinning Up in Deep RL* by OpenAI, which is a great beginner-friendly introduction to reinforcement learning in general. It's not specific to RLHF, but it gives you the solid grounding you'll need to understand the reward modeling used in alignment.

And finally, if you want to go full deep dive, read OpenAI's InstructGPT paper. It lays out the full RLHF process, including supervised fine-tuning (SFT), reward modeling, and optimization. Pair it with this video walkthrough from Umar Jamil that has all the derivations and code, and you're in a solid position.

More recently, Direct Preference Optimization (DPO) is getting attention as a more stable alternative to RLHF.

So, this is just another reminder that even though we've gone through the roadmap for understanding LLMs today, that doesn't mean we're done. This field is advancing quickly, so it's important to

Now, if you're someone who is actually more interested in building AI applications on top of existing models, check out my AI Engineering roadmap post next!

— — —

Btw, this post contains affiliate links. If you make a purchase I'll make a small commission, at no cost to you. Thank you for your support. ❤

#large-language-models #llm #data-science #machine-learning #ai