Podcast Transcript

EPISODE #170

The Heavy Metal of Machine Learning: Inside PyTorch

Published January 05, 2026 • Runtime: 22:54

https://myweirdprompts.com/episode/pytorch-inner-workings-history/

EPISODE SYNOPSIS

In this episode of My Weird Prompts, Herman and Corn break down the powerhouse that is PyTorch. They explore its origins from the Lua-based Torch to its current status as a community-governed giant under the Linux Foundation. You'll learn why its "define-by-run" philosophy beat out early TensorFlow, how Autograd handles the heavy lifting of calculus, and what "torch.compile" means for the future of speed. Whether you're a developer wondering why your builds are so massive or just curious about the "bridge" between Python and GPU hardware, this deep dive explains the engineering marvel behind today's AI revolution.

DANIEL'S PROMPT

Daniel

I’d like to learn more about PyTorch—its history, what it does, and its major versions. I’m also interested in who is behind the project and how such a major Python library is coordinated and managed, especially given its complexity and the vast number of dependencies involved.

TRANSCRIPT

Corn

Hey everyone, welcome back to My Weird Prompts. I am Corn, and I am sitting here in our living room in Jerusalem with my brother, the man who probably has more tabs open on his browser than there are atoms in the observable universe.

Herman

Herman Poppleberry, at your service. And you are not wrong, Corn. My RAM is currently begging for mercy, but it is all in the name of science. We have got a fantastic prompt today from our housemate Daniel. He was asking about the inner workings of PyTorch.

Corn

Yeah, Daniel mentioned that whenever he tries to build a PyTorch image, his computer starts sounding like a jet engine taking off. He wants to know what is actually under the hood, the history, how it is managed, and why it feels so massive.

Herman

It is a great question because PyTorch has basically become the oxygen of the artificial intelligence research world. If you are doing deep learning in twenty twenty-six, you are almost certainly using PyTorch, whether you realize it or not.

Corn

It is interesting because, to a casual observer, it is just another library you import in Python. But Daniel's right, it feels different. It is heavy. It is complex. So, Herman, let's start at the beginning. Where did this thing even come from? It feels like it just appeared and took over everything.

Herman

It definitely feels that way, but it has a really fascinating lineage. Before PyTorch, there was just Torch. And Torch was not even written in Python. It was written in a language called Lua.

Corn

Lua? That is usually what people use for game scripting, right? Like in Roblox or World of Warcraft.

Herman

Exactly. It is fast and lightweight. Torch was developed at places like New York University and the Idiap Research Institute starting way back in the early two thousands. But the real turning point was when the Facebook Artificial Intelligence Research group, or FAIR, started using it. They loved the flexibility of Torch, but they realized that the rest of the world was moving toward Python.

Corn

Right, because Python has that massive ecosystem. If you want to do data science, you go to Python. If you want to do web scraping, you go to Python. Lua is great, but it is a bit of an island.

Herman

Precisely. So, around late two thousand sixteen, the team at Meta, well, Facebook at the time, released PyTorch. The goal was to take the powerful C plus plus core of Torch and wrap it in a way that felt native to Python. They wanted it to be "imperative," which is a fancy way of saying it should behave like regular code. You write a line, it executes, and you can see the result immediately.

Corn

That sounds obvious, but I remember you telling me that TensorFlow, which was the big rival at the time, did not work like that back then.

Herman

Oh, man, early TensorFlow was a nightmare for a lot of researchers. It used a "static graph" approach. You had to define your entire neural network architecture upfront, like building a massive plumbing system, and then only after the whole thing was built could you pour data through it. If something broke in the middle, good luck debugging it. It was like trying to fix a leak in a pipe that is buried under ten feet of concrete.

Corn

And PyTorch changed that?

Herman

Yes. PyTorch introduced the "dynamic computation graph." In PyTorch, the graph is built on the fly as the code runs. If you want to use an if-statement to change how the data flows based on some condition, you just write a standard Python if-statement. This "define-by-run" philosophy is why researchers flocked to it. It felt like playing with Lego bricks instead of pouring concrete.

Corn

So, it was the developer experience that won people over. But what is it actually doing? When Daniel imports PyTorch, what is happening in those millions of lines of code?

Herman

At its core, PyTorch is two things. First, it is a tensor library, similar to NumPy, but with the ability to run on GPUs, or Graphics Processing Units. A tensor is just a fancy word for a multi-dimensional array of numbers. If a scalar is a single number and a vector is a list of numbers, a tensor can be anything from a simple table to a massive high-dimensional block of data.

Corn

Okay, so it is great at moving big blocks of numbers around. But lots of libraries do that. What is the second thing?

Herman

The second thing is the real magic: the automatic differentiation engine, or Autograd. When you train a neural network, you are basically doing a massive amount of calculus. You are calculating gradients to figure out how to tweak the weights of the network to make it smarter. Doing that math by hand for a model with billions of parameters is impossible. Autograd tracks every operation you perform on your tensors and automatically calculates the derivatives for you.

Corn

That explains why it is so compute-intensive. It is not just doing the math you asked for; it is keeping a "receipt" of every single calculation so it can work backward later.

Herman

Exactly. It is building a massive map of dependencies in the background. And when Daniel says his GPU starts smoking, it is because PyTorch is talking directly to the hardware. It uses something called CUDA, which is a platform created by NVIDIA, to offload those massive matrix multiplications to the thousands of tiny cores inside the GPU.

Corn

We talked about hardware acceleration a bit back in episode two hundred fifty-eight when we were looking at why some systems feel faster than others. It seems like PyTorch is the bridge between high-level Python code and the raw, screaming metal of the graphics card.

Herman

That is a perfect way to put it. It is a bridge. And because it has to support so many different types of hardware, from NVIDIA GPUs to AMD cards to Apple’s M-series chips, the dependency tree becomes enormous. You are not just downloading Python code; you are downloading pre-compiled C plus plus binaries, shared libraries, and hardware-specific drivers.

Corn

Which explains the "heavy" feeling. It is not just a library; it is almost like a sub-operating system for math.

Herman

It really is. And it has evolved so much. We are currently in the era of PyTorch two point zero and beyond. The jump from one point zero to two point zero was huge because they introduced something called "torch dot compile."

Corn

I have seen that in some of the newer documentation. What does the compilation step actually do if it was already supposed to be fast?

Herman

So, remember how I said PyTorch was "eager" and ran line-by-line? That is great for debugging, but it is actually a bit slower for the hardware because the GPU has to wait for Python to tell it what to do next. "Torch dot compile" takes your dynamic Python code and turns it into an optimized "graph" right before it runs. It is like having the flexibility of the Lego bricks during design, but then glueing them together into a solid block when it is time to actually use it. It can lead to speedups of thirty or forty percent without changing your model code.

Corn

That is a massive gain. It feels like they are trying to have their cake and eat it too—the ease of Python with the speed of static graphs.

Herman

Precisely. It is a very sophisticated piece of engineering. But before we get into who is actually running this show and how they manage all that complexity, let’s take a quick break for our sponsors.

Corn

Good idea. We will be right back. Larry: Are you tired of your brain feeling like a browser with forty-seven tabs open? Do you wish you could just "torch dot compile" your own thoughts? Introducing the Neuro-Sync Head-Band. This revolutionary, non-invasive, slightly itchy device uses low-frequency vibrations to "defragment" your consciousness. Users report a sixty percent increase in their ability to remember where they put their keys, though side effects may include hearing a faint dial-up modem sound during quiet moments and a sudden, inexplicable craving for lukewarm kale juice. The Neuro-Sync Head-Band—it is like a disk cleanup for your soul. BUY NOW!

Corn

...Thanks, Larry. I think I will stick to my messy brain for now, but I do appreciate the enthusiasm. Anyway, back to PyTorch. Herman, Daniel was really curious about the "who" and the "how." This is a massive project. Is it still just a Meta project, or has it become something bigger?

Herman

That is a really important distinction. For a long time, PyTorch was heavily identified with Meta. They were the primary maintainers, and most of the core developers worked there. But in late twenty twenty-two, they made a massive move. They transitioned PyTorch to the Linux Foundation and created the PyTorch Foundation.

Corn

Like how Linux or Kubernetes is managed?

Herman

Exactly. The goal was to move it to a neutral governance model. Meta is still a huge contributor, but they are joined by companies like Microsoft, Amazon, Google, NVIDIA, and AMD. This is crucial because it ensures that no single company can pull the plug or steer it in a direction that only benefits them. It is now a true community-governed project.

Corn

That sounds great in theory, but how do you actually coordinate thousands of developers across different companies? I mean, Daniel mentioned the "vast number of dependencies." If one person changes something in the core, doesn't it break everything else?

Herman

It is a constant battle, Corn. They use a system of "maintainers." There are core maintainers who have the final say on the most critical parts of the code, but there are also module maintainers for specific areas like "distributed" or "vision" or "audio."

Corn

I imagine the testing suite for this must be legendary.

Herman

It is intense. Every time a developer proposes a change, or a "pull request," it triggers a massive CI-CD pipeline—that is Continuous Integration and Continuous Deployment. They run thousands of tests across every imaginable hardware configuration. They test on Linux, Windows, Mac, on various versions of Python, and on different generations of GPUs. If your change makes a specific model five percent slower on an old NVIDIA Pascal card, the system will flag it.

Corn

That is the part that blows my mind. The sheer scale of the infrastructure needed just to *check* the code. It is not just about writing the math; it is about ensuring the math works everywhere, every time.

Herman

And you have to manage the "ecosystem" too. PyTorch is not just the core library. There is TorchVision for image processing, TorchAudio for sound, and TorchText for natural language. Then you have third-party libraries like Hugging Face Transformers or Lightning AI, which sit on top of PyTorch. The core team has to make sure they don't break those libraries when they update the core. It is a very delicate dance of versioning and compatibility.

Corn

It reminds me of what we discussed last week in episode two hundred seventy-five about air-gapped AI. When you have these massive, complex systems, the "supply chain" of code becomes a security and stability concern. If one of those dependencies has a bug, it can ripple through the entire AI world.

Herman

Absolutely. We actually saw a "dependency confusion" attack on the PyTorch nightly builds a few years ago. Someone uploaded a malicious package with the same name as a PyTorch dependency to the public Python repository, and some systems downloaded the malicious one instead of the real one. It was a huge wake-up call for the community about how vulnerable these massive projects can be.

Corn

So, how do they handle that now?

Herman

Better signing of packages, more rigorous checks on dependencies, and move toward more "hermetic" builds. But as Daniel noted, the complexity is still there. When you install PyTorch, you are often pulling in hundreds of megabytes, or even gigabytes, of data. A lot of that is the NVIDIA CUDA libraries. If you are on a slow connection, it can take forever.

Corn

Is there any move toward making it lighter? Or is this just the price we pay for high-performance AI?

Herman

There are definitely efforts. There is something called "ExecuTorch" which is a new, lightweight version designed specifically for mobile and edge devices—like your phone or a small sensor. It strips out all the "training" stuff and just focuses on "inference," which is running the model once it is already smart.

Corn

That makes sense. You don't need the "receipt" or the Autograd engine if you are just trying to recognize a face in a photo. You just need the final math.

Herman

Exactly. But for the researchers and the people building the next generation of models, the "heavy" version is still the gold standard. It is the flexibility that matters most.

Corn

You know, what I find most interesting about the history is how PyTorch essentially "won" the research war against TensorFlow. If you look at academic papers today, the vast majority use PyTorch. Why do you think that is? Was it just the "eager execution" we talked about earlier?

Herman

That was the spark, but I think the real reason is that PyTorch feels "pythonic." It feels like it was written by Python developers for Python developers. TensorFlow always felt like a Google product that was being "translated" into Python. PyTorch embraced the community. They made it easy to write custom extensions in C plus plus. They made the documentation excellent. And they stayed very close to the research community.

Corn

It is like the difference between a tool that is handed down to you by a big corporation and a tool that is built by your peers in the workshop next door.

Herman

That is a great analogy. Even though Meta is a giant corporation, the FAIR team always acted more like an academic lab. They published their work, they shared their code, and they listened to feedback. That culture is baked into PyTorch.

Corn

So, for Daniel, who is seeing his computer smoke and wondering why this thing is so massive—the takeaway is that the "massiveness" is actually a feature, not a bug. It is the weight of every possible mathematical operation, every hardware driver, and every research breakthrough of the last decade, all bundled into one place.

Herman

Precisely. It is a heavy-duty industrial machine that we have managed to fit into a Python wrapper. It is the "Steam Engine" of the twenty-first century. It is bulky, it is hot, and it requires a lot of fuel, but it is what is driving the entire revolution.

Corn

It is also worth mentioning the versioning. Daniel asked about the major versions. We had the initial release, then the one point zero milestone which signaled it was ready for production, not just research. And now we have the two point zero era, which is all about optimization and compilation.

Herman

And looking forward to twenty-six and beyond, the focus is really on "distributed" computing. As models get bigger—we are talking trillions of parameters now—they can't fit on one GPU. They can't even fit on one server. PyTorch is evolving to make it seamless to split a model across hundreds or thousands of GPUs as if it were one single machine.

Corn

That sounds like a whole other level of complexity. How do you even debug something that is spread across a data center?

Herman

With great difficulty, Corn. With great difficulty. But PyTorch is building tools like "TorchSnapshot" and improved distributed debuggers to help with that. It is all about managing that complexity so the researcher can just focus on the ideas.

Corn

It is fascinating to see how a project that started as a way to make Lua more accessible has turned into the foundation of global AI infrastructure. And the fact that it is now under a foundation like the Linux Foundation gives me some hope that it will stay open and accessible.

Herman

I agree. It is one of the most successful examples of open-source collaboration in history, right up there with the Linux kernel itself.

Corn

So, what can our listeners actually take away from this? If they are looking to get into PyTorch or if they are already using it and feeling overwhelmed by the complexity?

Herman

My first takeaway would be: don't be intimidated by the size. You don't need to understand the C plus plus core to be a great PyTorch user. Start with the high-level concepts—tensors and autograd. If you understand those two, the rest starts to make sense.

Corn

And I would add, lean on the community. Because PyTorch is so dominant in research, if you have a problem, someone else has probably already solved it on a forum or a GitHub issue. You are not alone in the complexity.

Herman

Also, keep an eye on "torch dot compile." If you are still running your code in the old "eager" mode, you might be leaving a lot of performance on the table for very little effort.

Corn

And for the developers out there, maybe take a moment to appreciate the "maintainers" we talked about. The people who are running those thousands of tests every day to make sure your code doesn't break. It is a thankless but vital job.

Herman

Absolutely. They are the unsung heroes of the AI age.

Corn

Well, I think we have thoroughly explored the world of PyTorch for today. Daniel, I hope that helps explain why your GPU is working so hard. It is doing a lot of very clever math on your behalf.

Herman

And if it actually starts smoking, Daniel, please, for the love of all that is holy, turn it off. We live in the same house, and I don't want to have to explain to the insurance company why the living room smells like burnt silicon.

Corn

Good point. Safety first, even in deep learning.

Herman

Always.

Corn

Well, this has been a great dive. If you are listening and you found this helpful, or even if you just enjoyed the mental image of Herman's browser tabs, we would really appreciate a review on your podcast app or on Spotify. It genuinely helps other people find the show and helps us keep doing this.

Herman

It really does. And remember, you can find all our past episodes and a way to get in touch with us at our website, myweirdprompts dot com. We are also on Spotify, obviously.

Corn

Thanks again to Daniel for the prompt. It is always fun to dig into the tools we use every day but often take for granted.

Herman

Definitely. Until next time, keep your tensors aligned and your gradients flowing.

Corn

This has been My Weird Prompts. I am Corn.

Herman

And I am Herman Poppleberry.

Corn

See you next time, everyone.

Herman

Bye!

Corn

Let’s go see if Daniel’s computer is actually on fire.

Herman

I’ll grab the fire extinguisher, you grab the marshmallows.

Corn

Deal. Larry: BUY NOW!