SACRA

EXPERT INTERVIEW

UPDATED
08/27/2024

Kyle Corbitt, CEO of OpenPipe, on the future of fine-tuning LLMs

TEAM

Jan-Erik Asplund
Co-Founder
jan@sacra.com

Published on **Aug 27th, 2024**

Kyle Corbitt, CEO of OpenPipe, on the future of fine-tuning LLMs

By **Jan-Erik Asplund**

Background

Fine-tuning their own custom models for specific tasks has been a key cost-saving strategy for companies running LLMs in production. We reached out to Kyle Corbitt, ex-Y Combinator, founder & CEO at OpenPipe (Y Combinator, Summer 2023 batch), to better understand the long-term trajectory of LLM fine-tuning.

Key points from our conversation via Sacra AI:

- **Companies like Ramp, Notion and Databricks take outputs from frontier models like OpenAI’s GPT-4 and use them to fine-tune smaller, cheaper and faster open source models like Mistral and Llama 3.1, reducing their costs by 90% for specific, repetitive tasks**. “Instead of trying to distill all of the capabilities of this larger model into a very small one, you use a subset of outputs for a specific task to train a smaller model. That smaller model obviously isn't going to match the larger model in a lot of other areas, but for a specific task, it can.”
- **Even with the expanding context windows of LLMs like Anthropic’s Claude (200K tokens), few shot prompting—or**

prompt engineering including a few examples of desired output—struggles to achieve consistent, high quality output, particularly for complex requests. “You can list 5 steps out in a prompt, and GPT-4 will follow those steps and get the category you expect about 75-80% of the time—fine-tuning, on the other hand, can pick up complex instructions essentially 100% of the time, as long as it's properly implemented.”

- **Fine-tuning platforms like OpenPipe ($6.7M raised), Predibase (Felicis, $28.5M raised) and Airtrain AI (Y Combinator, $3.2M raised) unbundle fine-tuning from the LLMs and rebundle it with MLOps, combining a software development kit (SDK) for collecting prompt-completion pairs with model training, eval, and deployment to launch and rapidly iterate on fine-tuned models.** “[The] buyer is more like a product team at Notion or Airtable… The data science team is off doing their own thing, and the product team doesn't want to have to interface with them, get headcount on that team, or get their stuff prioritized. They want to just be able to handle this themselves.”

Interview

Tell us about OpenPipe—what was the key insight that led you to start the company and focus on fine-tuning?

We started OpenPipe in March of last year, shortly after ChatGPT had come out and almost simultaneously with the release of GPT-4. Initially, we were experimenting with various ideas, and the first thing we built was more of an agent than a fine-tuning solution. We developed an agent to control web browsers, allowing users to take actions. This was early 2023, and there weren't many other companies working in this space at the time.

We created an impressive demo where the agent could control your browser, post on Twitter, and book flights. The demos generated significant interest, with many GitHub stars and investor inquiries. However, we encountered two main issues: reliability and cost. The agent would perform the requested task only about 60% of the time, and each demo run cost $15-$20 due to the large number of input tokens required.

We explored different approaches to improve efficiency and reliability, including extensive prompt engineering. We quickly realized that even with GPT-4, which was an extremely strong model, we couldn't achieve the necessary reliability and cost-effectiveness. Our options were to wait for better models or explore alternative approaches, and we decided to focus on fine-tuning.

We began experimenting with open-source models like FLAN-T5, which was one of the best available at the time. We had some key learnings: there was significant promise in fine-tuning, as we could achieve nearly 100% reliability for tasks that GPT-4 could only do 80% of the time, and at much lower costs. However, building the workflow to fine-tune the model, collect user feedback, and improve quality over time was a major engineering challenge.

We recognized that fine-tuning could greatly enhance capabilities and reliability, but it typically required having a specialist on staff. We realized that if we built the right workflow tool, we could replace that specialist and make it possible for anyone proficient in prompt engineering to successfully build a fine-tuned model.

That's what we ended up building, and that's what OpenPipe is today: a platform that makes it extremely easy to go from having a well-functioning prompt in production to having a fine-tuned model that is cheaper, faster, and far more reliable.

What are the core categories of unreliability that fine-tuning addresses? Is it issues such as returning incorrectly formatted JSON, or other types of problems?

I think the very basic stuff, like getting formatting right, doesn't actually need fine-tuning. There are other techniques, such as constrained decoding, that can solve that problem. The issue that fine-tuning really solves is things that are a little bit more complex. For example, you might have a 5-step process you want to go through as you're analyzing to determine if a particular page or document falls into a specific categorization. You can list those 5 steps out in a prompt, and GPT-4 will follow those steps and get the category you expect about 75-80% of the time. Sometimes, however, it'll forget to do one step.

Maybe you have very nuanced instructions, like "If this is true, then I want my output in this format, but if it's this other kind of thing, then I want the output in a different format." Anyone who's written these complex prompts realizes that you can get it to work some of the time, but it's not going to follow those complex instructions consistently. Fine-tuning, on the other hand, can pick up those complex instructions essentially 100% of the time, as long as it's properly implemented.

If you have a good dataset and build a lot of tooling to make it extremely easy to have confidence in your dataset, then those really complicated instructions that prompted LLMs have trouble with, your fine-tuned model is going to do great on.

Could you walk me through what the total workflow would look like in a standard deployment? I'm interested in the process from collecting and validating data to the iterative aspect of how the fine-tuning gets improved. Does it become more fine-tuned with more feedback? What does that whole process look like?

That is a fantastic question. Typically, when customers come to us, they have a prompt that is mostly working or working well enough that they can actually ship it to production. In fact, that's one of the things we really encourage people to do whenever possible: start with a prompt, push it as far as you can, and get it deployed in production. I'll explain why that's important.

Users usually start from that point when they join OpenPipe. They install our SDK, and we begin collecting all of the requests they're sending to OpenAI, Anthropic, or whatever model they're using, along with the responses they're getting back. We have a logging feature, essentially a tracing feature, where you can see all of those interactions.

When you're ready to fine-tune a model, we prefer that you have this deployed in production for at least a while. This ensures you have a nice representative sample of how people are actually using it and what actual model documents or input chat messages your users are using. We want to make sure we have a good representative sample, as that's really important for high-quality fine-tuning. That's why it's important to be in production before you start this journey.

Once you have that data, you pull it into OpenPipe. We have a way to select subsets, either randomly or later on if you notice one class of document or type of prompt that's having more trouble. We have the ability to up-sample data in certain areas. You might grab, say, 1000 rows that are relevant to your task.

We then have an enrichment and filtering process, which really helps the performance of the fine-tuned model. These models learn what's in the dataset fantastically, but if you don't have a high-quality dataset, they won't have great performance. So improving the data is critical.

We have several tools built into OpenPipe that help with this. One is what we call a "mixture of agents relabeling flow." This takes your real requests from your production model but, instead of just using the outputs from your production application to train on, it goes through an iterative process. It figures out the instructions and individual criteria that make a good response, checks the answer against all those criteria, and rewrites the answer until you have something high-quality. We've found that this makes a huge difference in the quality of your dataset and fine-tuned model.

We also have filtering tools and a human evaluation and relabeling flow. If you want, you can review all of these yourself and make sure they look good before you fine-tune.

Once you've gone through that process and have a dataset you're happy with – we're talking maybe a few hundred to a few thousand rows, so it doesn't have to be huge – the actual fine-tuning process is extremely straightforward. It's essentially one click through a UI: you select a base model, have your dataset selected, and just hit a button to start fine-tuning.

The process is fast and relatively cheap. I think people are often surprised by how inexpensive it is. For a typical job with a few thousand entries, we're talking maybe $50 or $100 to train a model. But often, our customers are spending thousands of dollars every month on inference before coming to us, so the prices are not a big issue.

You hit that button, fine-tune the model, and within a few hours in most cases, you'll have your model back. At that point, it's just a direct drop-in replacement. We provide an API endpoint that's 100% compatible with OpenAI's chat completion

endpoint, so you can use your existing code. You just change the base URL and the model name to point to your new fine-tuned model and start running against it in production.
We also have an evaluation feature, which we find is really important. Obviously, if users have their own evaluations, they should run those against their new model as well. But we also have an on-platform option where we can use different criteria and LLM judge-type things to compare your dataset's quality versus your fine-tuned model quality and make sure that it's going to perform well for your actual use case.

I'm curious what kinds of verticals or sectors you've seen strong product-market fit in. We've talked about the bias towards companies that are shipping product more actively, but I'm wondering if there's any crossover with areas like compliance or fintech. What does that landscape look like?

We definitely see that most of our customers are smaller companies and startups in the seed through Series B stage. We see a lot of adoption there because that's honestly where we're seeing a lot of successful products being built in generative AI right now. We're also seeing adoption from some of the larger companies that are really tech-first and still relatively young. You can think of companies like Notion, for example. These companies that are adopting generative AI rapidly are also adopting fine-tuning really rapidly.

I have found that the enterprise market is still pretty early on this. We have a lot of meetings with larger enterprises, and they're still discussing internally what their policies are and trying to find use cases. This has been interesting to me given how much energy and excitement there seems to be coming from the boardroom. It feels like the actual adoption has been much slower in that segment.

You mentioned that before OpenPipe, people were spending thousands of dollars a month. What are the key libraries or tools you've seen folks using before OpenPipe? What combinations of tools do they use to capture all these different functions they want to perform, such as data fine-tuning and iterative improvement? Essentially, what does the pre-OpenPipe version of this process look like?

It's a good question. To start with, our typical customer has never deployed a fine-tuned models in production before they come to us. The typical stack our customers are coming from is using OpenAI or, at this point, Anthropic's models. They'll often have an observability solution they're already using, which could be an LLM-specific one or their existing observability provider like Datadog. That's honestly mostly where I see people come in with the stack.

Even evaluations at this stage are pretty hit and miss. A lot of folks just do vibe checks on their outputs. We definitely have some companies that are effectively using prompt studios and evaluation frameworks as well, but it still feels like the minority at this point.

For folks who have done fine-tuning before, it's almost exclusively using open-source tools. They're using things like Unsloth or Axolotl, which are great companies that provide open-source tools that let you fine-tune on your own. But again, it's on you to get the dataset ready and things like that.

We have also seen some companies try fine-tuning through OpenAI itself because they have an API available for that, but with mixed success, I would say.

What does OpenPipe's business model and pricing look like today? How are you thinking about capturing the value you create for customers in terms of cost savings and performance improvements?

We're much more interested in creating value than capturing it. If our customers go through us and have far cheaper and faster models and are super happy, I'm pretty happy with the job we did. I think for us, there are interesting business models in the future around monitoring fine-tuned models and making sure we're detecting data drift, which we actually do with some of our enterprise customers right now. Certainly turning that into more of a hosted platform as well.

I think there are interesting future products we can build in that space, so I'm not particularly worried about business model things. There's just so much that doesn't exist out there that we need to build, and if we do, it's going to provide a ton of value to people.

Honestly, with the fine-tuning work we do, we let you export your models and you can host them anywhere you want. We are very happy if people come and have a great experience with us for fine-tuning, and then we only ever see those $30 or $50 or whatever to fine-tune their model. We're still very happy we served that customer. But we do have a lot of customers that use us for much more than that, and I expect they'll continue using us for more in the future.

It makes me think that there's potentially going to be a need for something like a Sentry or Datadog, specifically for fine-tuned models or for running models in general. It doesn't seem like it would be easy for those existing companies to add this as a functionality, so I'm curious if that's the vision.

I think more broadly with fine-tuning or fine-tuned models specifically, there is a much deeper integration you can do with monitoring the outputs than you can if you're not in the fine-tuning space. What we do is detect if an output is bad, and then we can have a fully closed-loop process where you fix the output or put it in a human queue for reviewing and fixing. Then you put it back in the dataset, retrain the model, and redeploy it. There's a ton of value added in having all of that in the same place, as opposed to having separate pieces to do the monitoring and the training.

You mentioned that OpenAI has a fine-tuning API, and I believe Hugging Face offers model fine-tuning as well. How do you think about OpenPipe's positioning against these companies with significant mindshare in generative AI? I've tried the OpenAI fine-tuning experience, and it's obviously not as feature-rich as what you're describing. On a higher level, how do you think about positioning against these other companies? Is there a potential partnering angle to it? I'm curious about your thoughts on this.

It's a fantastic question. First of all, many people don't realize that you can fine-tune OpenAI's models through OpenPipe as well. You just have to input your OpenAI API key, and we'll run that for you. The reason people do this is that we do such a great job on the dataset preparation side, which I described earlier. OpenAI, of course, doesn't operate in that space at all.

A lot of folks use us for this purpose. In fact, we have a great relationship with the fine-tuning team at OpenAI. We've had early preview access to fine-tuning a number of their models and have a very open communication channel. Based on what they've said, I think a substantial fraction of all people who are fine-tuning OpenAI's models are doing so through OpenPipe. We have many users who use us in exactly that way, and I suspect it's a very large fraction of everyone who's fine-tuned those models.

I think we are very complementary to OpenAI because we help you get the data in a great format, ensure your model is performing well, monitor outputs, and then go ahead and refine-tune for you. These are features that OpenAI doesn't appear to be interested in building, at least for now.

It occurs to me that Scale might be another relevant company. They were originally more focused on autonomous driving and hardware-heavy use cases with a lot of human labeling and government contracts. They're also more enterprise-focused. How do you think about Scale? Is it a relevant company for you in that way?

You know, we haven't had any customers comparing us to Scale. I think Scale is a very different product, and it's frankly sold to a very different buyer within an organization. Scale is for situations where you've got a data science team saying, "We're trying to build this model from scratch, and we need a million labeled examples in this area." They'll talk to you and sign something like a $10 million contract with you.

Our buyer is more like a product team in Notion or Airtable. Yes, they have a data science team, but that's not actually the person who's talking to us. The data science team is off doing their own thing, and the product team doesn't want to have to interface with them, get headcount on that team, or get their stuff prioritized. They want to just be able to handle this themselves. So these product teams we're talking to—Scale doesn't have a product for them.

Data is key for fine-tuning and OpenPipe has access to a lot of it via its customers' training data. Are there opportunities there for you to help folks get access to better data, whether through some kind of sharing

agreements or increasing your ability to generate synthetic data?
Absolutely, yes. We have thousands of teams building fine-tuned models on OpenPipe right now. Almost as importantly, they're also defining their evaluations on our platform in many cases.

This is actually hugely helpful because when we're testing a new base model, a new heuristic to set hyperparameters, or a new training regime, we can anonymously launch a bunch of jobs in a shadow background process against our existing users' datasets.

We then run their actual existing evaluations and get back a nice dashboard report showing the results for about 20 projects where we tried these new hyperparameters.

These evaluations are defined by the users to capture what they care about, allowing us to see the changes and gather data much faster than almost anyone else. There aren't any other platforms out there that integrate these features, which has been incredibly helpful for us.

I think this comes across in the results. We compare ourselves head-to-head against every other fine-tuning provider out there. We run our own evaluations on the same datasets, and our quality consistently comes out ahead, even holding the base model constant. I believe this is a true testament to the power of having actual user evaluations that we can check.

A great example we published recently is for the latest Llama 3.1 release. We fine-tuned the same datasets across that and also on GPT-4o-mini, which was released for fine-tuning on the same day, (perhaps not coincidentally).

We ran a head-to-head comparison using our own optimized hyperparameters, which we had tuned by looking at actual customer evaluations in an anonymous way. Training the same models on the same datasets, we were consistently able to get higher quality results on customer evaluations using Llama 3 18B than fine-tuning GPT-4o-mini. This is notable because GPT-4o-mini is generally considered the stronger and more capable base model. It just shows that the actual fine-tuning process can make such a big difference in achieving better results.

Meta launched a 405B parameter Llama model which they recommend folks use to train smaller models. Do you think of distillation as a paradigm that's competitive to fine-tuning or is that something that OpenPipe could look to facilitate? Or is it just a tailwind because of how it could drive more usage of open source models? What are the tradeoffs with each approach, fine-tuning vs distillation?

Okay, so distillation and fine-tuning are two separate things, although there are a number of overlaps. All of these terms are a little bit overloaded, to be honest. Distillation is, in general, the process of taking the behavior of a very large model and creating a smaller model that copies that behavior as closely as you can. It can be for general-purpose applications, which is basically what we see with the distillation that LLaMA 3.1 did. They took this incredibly powerful 405 billion parameter model and tried to distill its abilities into two different smaller models of 70 billion and 8 billion parameters.

Distillation can also happen at the per-task level. Instead of trying to distill all of the capabilities of this larger model into a very small one, you can use a subset of outputs for a specific task to train a smaller model. That smaller model obviously isn't going to match the larger model in a lot of other areas, but for your specific task, it can.

That form of distillation is basically very similar to the way a lot of users use OpenPipe, where they're going to take some larger model, whether it's GPT-4 or whatever, and then try to train a smaller model to behave the same way for their specific task.

So, fine-tuning and distillation are two different things that have a lot of overlap. Fine-tuning doesn't have to be related to distillation. You can generate the training data that you use for fine-tuning in many different ways which aren't necessarily related to distillation. But distillation is one source or one way you can generate the training data. I guess what I should say is, if you do use a very large model to generate the training data for your fine-tune, then you could consider fine-tuning in that case as a form of distillation.

Regarding the tailwinds point, is that effectively how you see it? That more great open-source models, especially

smaller ones, are a boon because they'll be in more people's hands and will likely require more fine-tuning?
I would say having stronger models is great in general because it means there are more use cases where people can successfully use generative AI. I think it's good for fine-tuning in the sense that it's going to make the whole market larger because these models are better and can do more. It doesn't really change the trade-off much between fine-tuning versus pure prompting without fine-tuning.

However good the base model is, it's always going to be way better post-fine-tuning for your specific task than if you try to just prompt it. Previously, there were fewer things you could do through prompting and also fewer things through fine-tuning. Now, with stronger base models, both of those ceilings have moved up higher.

As context windows on LLMs grow, some have argued that fine-tuning becomes less important. What are the key considerations here? What are the strengths of long context, and what are the strengths of fine-tuning?

When people say long context makes fine-tuning less relevant, they're typically referring to what's called few-shot examples or in-context learning. The basic idea is this: instead of fine-tuning a model on, say, 20 examples of inputs and outputs, you prompt a model by sending it those 20 examples in the context. Then you add your expected input at the very end, hoping the model will learn from the context window and replicate the pattern.

This approach can work pretty well. If you have 20 examples and throw them into the context, it can be effective. However, there's a limit to how many examples you can include in the context window. If you have 500 or 10,000 examples, which is common in production use cases, you're going to get much better results from fine-tuning than from in-context learning because you can't fit all of those examples in the prompt.

The important nuance that many people miss in this discussion is that even with a small number of high-quality examples, sticking them in the context window every time makes each of your prompts 20 times more expensive and significantly slower. If you're using 20,000 tokens per input instead of 800 or 1,000, you'll end up with much slower generations because the model has to preprocess that lengthy prompt each time. In contrast, with the type of fine-tuning we do, which is LoRA-based, we're updating a very small number of weights in the model - about 0.5% to 1% of the total weights. When you make a request to your fine-tuned model, we apply these updates on the fly in the GPU. This does have a performance impact, but it's only about 20% to 30% slower than calling the base model. Compare that to the 20x impact you get from applying 20 examples in context every time, and it's a far smaller impact.

In practice, you're going to end up with much faster and cheaper generations if you do it through fine-tuning as opposed to throwing all the examples in the prompt every time.

Looking ahead to the future, if everything goes right for OpenPipe over the next 5 years, how do you envision OpenPipe's role and how the world will be different?

I think the majority of all inference should happen on models that are trained through OpenPipe. Whether that's deployed through us or whether those are models you export and deploy through some other inference provider, we're totally happy either way. I believe our fine-tuning stack is already the strongest and will continue to improve.

We have barely scratched the surface on the lowest hanging fruit of things we can do to enhance model quality. I expect it'll just keep getting better. People are going to use this because it's the way to get the best possible completions at the lowest price as you move to the smaller models. That's what I think will happen.

Is there anything that we didn't talk about that you think is important to mention?

It's really easy to get started. It takes less than an hour to go from installing the SDK to successfully having a model training. It's typically a direct drop-in replacement. You can get about 80% of the total possible benefits from fine-tuning without any effort at all - literally just capturing your request logs and then training the model. So yeah, it's very easy to use, very easy to adopt, and in my opinion, there's no reason not to.

SACRA

Disclaimers