EPISODE 1122

[INTRODUCTION]

[00:00:00] JM: CrowdFlower was a company started in 2007 by Lukas Biewald, an entrepreneur and computer scientist. CrowdFlower solves some of the data labeling problems which were not being solved by Amazon Mechanical Turk. A decade after starting CrowdFlower, the company was sold for several hundred million dollars. Today, data labeling has only grown in volume and scope, but Lucas has moved on to a different part of the machine learning stack, tooling for hyperparameter search and machine learning monitoring.

Lukas joins the show to talk about the problems he was solving with CrowdFlower, the solutions that he developed as part of that company and the efforts with his current focus, Weights & Biases, a machine learning tooling company.

If you are a writer, then you might be interested in writing for Software Engineering Daily, either articles or helping with research and preparation for these shows, send me an email, jeff@softwareengineeringdaily.com if you're interested. Also, I am investing in software companies. If you are building some kind of software company or infrastructure company, send me an email, jeff@softwareengineeringdaily.com.

[SPONSOR MESSAGE]

[00:01:13] JM: This episode of Software Engineering Daily is sponsored by Datadog. Datadog is a cloud monitoring platform built by engineers for engineers enabling full stack observability for modern applications. Datadog integrates seamlessly to gather metrics and events from more than 400 technologies including cloud providers, databases and webservers. Easily identify slow running queries, error rates, bottlenecks and more fast with built-in dashboards, algorithmic alerts and end-to-end request tracing and log management from Datadog.

Datadog helps engineering teams troubleshoot and collaborate together in one place to enhance performance and prevent downtime. You can start a free trial of Datadog today and

Datadog will send you a free t-shirt. Visit softwareengineeringdaily.com/datadog to get started. That's softwareengineeringdaily.com/datadog.

[INTERVIEW]

[00:02:18] JM: Lukas Biewald, welcome to the show.

[00:02:20] LB: It's great to be here.

[00:02:21] JM: You started CrowdFlower, which is now figure 8 back in 2007. Explain the problem you were solving in 2007 with CrowdFlower.

[00:02:30] LB: Sure. And I was really early with this problem I realized now. But I had been working at Yahoo and then a startup called PowerSet that became Microsoft Bing. And we're building machine learning models to deploy into production to make search engine results better. And we had to collect lots of training data to teach the algorithms how to run. And that turned out to be kind of the hardest part for building machine learning back then, and some would argue even today.

And so I started CrowdFlower to help companies collect training data for their machine learning models. So back then, it was typically applications were search, and ecommerce did a lot of search. And then overtime as image and audio applications got better, we started doing more autonomous vehicles and things like that.

[00:03:17] JM: Why wasn't this problem solved by Mechanical Turk.

[00:03:21] LB: Ooh! Good question. It was somewhat solved by Mechanical Turk. So I was a heavy user of Mechanical Turk actually before I started CrowdFlower, and there are parts of it that works pretty well. It was a very low barrier to entry. But I think the vision of Mechanical Turk where you would put labeling tasks out into the ether and then people would just work on them, it wasn't really realistic for real-world applications.

So a lot of labeling tasks seem kind of obvious if you just think about the sort of plutonic idea of this situation. And then actually the details make them hard. So I'll give you an example. Say, you're trying to label pedestrians for your autonomous vehicle. So it's obviously very important. You want your car to know where the pedestrians are. And if they're crossing in front of your car, you want your car to stop. So that seems like a simple task that humans would be really good at, like identifying humans in an image.

In fact, humans are excellent at it. But there are lots of details that come up if you start to actually look at real-world examples. One I remember is what if there's a human on a billboard? Should that get labeled as a person? And that actually kind of depends on what you're doing downstream. Or if you see a human and the bottom half of the human is included by something and you're trying to put a bounding box around them, should you put the bounding box around where you guess the bottom of the human is? Or what if you see a human in a wheelchair, should you like include the wheelchair in the bounding box? What if you see a baby carriage and you know that there's a baby or you're guessing there's probably a baby inside there, should you include that? Should the bounding box include the parent that's pushing the baby carriage and the carriage? Should it just include the parent? All these details that turn out to be incredibly important.

The big trouble that Mechanical Turk had was that you would post into their marketplace a task and you couldn't really easily communicate with the people doing the task. And so it was hard to get the level of quality that companies needed to make their machine learning algorithms work.

[00:05:22] JM: What kinds of labeling problems did CrowdFlower specifically solved? Can you go into a little more detail over the labeling problems it solved and what it was not capable of solving?

[00:05:32] LB: Totally. We were a horizontal tool. So we did lots of different problems. You can log it to our platform and set up all kinds of different tasks. And the kinds of tasks that were common changed over the course of the time I was running it. And I think a lot of the change was really due to the machine learning tasks that were possible. In the early days, I remember one very common task that you wouldn't necessarily guess would be common was parsing addresses. This happens a lot because companies will have addresses in a CRM and they want

to know the addresses of where all their customers are. They want to know like which parts of the address is like the physical address. What's the ZIP code? What's the street and so on?

So they would set this up this task, which is another one that sounds super simple until you start looking at real-world addresses. And then you deal with, "Do you want the headquarters of the business or do you want every branch location? Do you want the –" Often, places will have a delivery address and a physical address. And then if you start looking at international addresses, it can be super hard to parse. I remember looking at Irish addresses and that looking like it almost seem like something you tell us where the gas station. Like used to be like, "Oh, past the big building and then down by the rock over there in the left." So it's super important task to businesses. That's still, I think, not totally solved.

And then overtime we started to see more and more diverse sets of, especially, image applications. We would see things like – I'm just thinking of some things I really remember. But there were things of like looking inside people's ears and saying if they had an ear infection, which is a pretty easily labeling task actually for layman, or at least it can be super gross and clear if someone has an infection, or looking at cells and picking out – Even counting the number of cells in a slide can be important for medical applications.

It's really all over the board. I mean, it was basically all the different applications of machine learning that you see today. As autonomous vehicles and autonomous delivery stuff got popular, there are lots of things that those companies need to do. Like something called semantic segmentation, where you literally try to take every pixel in an image and label, "Is this part of a road? Is it part of a sidewalk? Is it part of a tree? Is it safe to drive?" And I think a lot of people maybe don't realize that all these different tasks are actually how machine learning works underneath the surface.

The reason that eBay can show you reasonable search results is because they've had lots and lots of people labeling. This is a good search result for this kind of query. This is a bad search result for this kind of query, and then built an algorithm to basically kind of automate that process.

[00:08:08] JM: Tell me about some of the most difficult times in running CrowdFlower.

[00:08:12] LB: Well, we had kind of two big pivots that were pretty hard, right? I'll tell you, like one thing that happened was we had an on-demand business, but it was very hard to predict the volume. So when I started the company, we basically priced per label that someone wanted to collect. And what's tough about that was that companies themselves often didn't know how many labels they were going to collect.

So it made it very, very hard to forecast the volume of labels that someone is going to collect. That it made it very, very hard for us to forecast our revenue, and that might seem like it's not just a business problem. Then it makes it hard to know how many engineers you can safely hire. How many salespeople you can hire? And so it often felt like we were really running by the skin of our teeth, and it felt like we were little bit maybe misaligned with our customers and that we wanted to kind of push them to get as many labels as possible.

The challenge was that the customers didn't often necessarily need so many labels if they labeled it well or something, right? But I realized the whole company was set up to actually encourage customers to do as much labeling as they want. Like not get as much value out of the labels as they could. So we made this really hard decision to pivot to selling out software as a SaaS technology offering that would help customers collect the label. So we just priced that out as a SaaS piece of software. And then on top of that, like a marketplace of labels where they could request labels to get collected from humans.

As part of that change, we actually had to let go a huge number of people, especially in sales and support as almost half of the company. It was shrinking from maybe 80 people down to 30 people or so. And that's the worst experience for the people that are affected by it. It was a really tough call to make. It's really hard on everyone. It's hard on the people that – Extreme hard that people who left. It was hard on the people that stayed to work at a much smaller company. It really felt like a reset for everyone, even though we had been doing it for I think 5 or 6 years at that point.

[00:10:12] JM: So over this 11-year long haul that you were with the company, there were a lot of changes in the market. And you've already touched on some of these. What were the other changes in the market that altered labeling requirements for training data?

[00:10:27] LB: The biggest change that we experienced over the 11 years of running CrowdFlower was a change in the market from kind of a desert where machine learning wasn't considered to be something interesting to work on. Investors considered it a liability more than an advantage to a really hot space. For the first six, seven years, we didn't have any competitors. And that really wasn't a good thing, right? That was just because it wasn't really an interesting space to anyone. We would try to figure out what conferences they go to, and there weren't really conferences that made sense where people would be buying software to – Towards the end, it started to become a really hot space with like lots of [inaudible 00:11:05] competitors.

And now I go to conferences and I see tens or no maybe even more labeling companies and people have sort of decided that labeling is like a really meaningful space, and labeling for machine learning, it's a standalone venture-backable business. And I would say for the first half decade that I was running CrowdFlower, that was definitely not the case. People thought that the market was too small.

In fact, I remember a VC telling me that they thought the total market for what we're doing was $4 million of annual revenue, and we'd already – We're at, I think, $3 million of annual revenue at that point. So they were very negative on this space that we were in.

[00:11:45] JM: Today you work on Weights & Biases, which is a new company. How did your experience with CrowdFlower inform the work with Weights & Biases?

[00:11:53] MR: Well, when I sold CrowdFlower, I thought a lot about kind of the things that I liked about CrowdFlower and the things that were hard about CrowdFlower, and I felt like my favorite thing, the thing that I felt like I could really dedicate my life to is supporting the people doing machine learning, the actual practitioners.

When we started CrowdFlower, we couldn't really sell right into machine learning practitioners. They just didn't have enough budget. They weren't really good buyers. And I'd go to CMO conferences and I'd go to CIO conferences, because those are the people that were really buying the CrowdFlower software, kind of on behalf of the machine learning practitioners. We

aimed a lot of a marketing really at the buyer, and that's was kind of hard for me. I found it kind of counterintuitive often to sell into CIOs and CMO's. I was never a CIO or a CMO. And it'd be kind of even harder to like make small talk at those conferences, whereas when I would go to academic machine learning conferences, it was much more fun for me. It was much more exciting to me the things of people are working on.

And so I really wanted to build a company that was all around the machine learning practitioner themselves, and I felt like it would be a lot of fun to just build more stuff for the practitioner. And I felt like the market was ready for a company that just sold into practitioners themselves. That was really the core thinking, was that I really liked this customer and I really wanted to make more stuff for this very specific customer.

In a way, it's an extension of a lot of the stuff I worked on at CrowdFlower. But instead of kind of starting with a solution like I did with CrowdFlower, it was starting with the customer in mind and sort of doing an open-ended exploration of the things that that customer profile might want.

[00:13:36] JM: Why is your company called Weights & Biases?

[00:13:38] MR: It was actually my cofounders idea. But I think it kind of reflects the fact that we are completely on the side of the ML practitioner, because it makes no sense to anyone else. Weights & Biases is a technical term inside of stats or machine learning or in deep learning, and it's basically the coefficients that you have in your model.

A deep learning model will consist of numbers that are called weights or biases. But I think the reason we really liked the name was that we really wanted to make the point that our company at its core is on the side of the ML practitioner. So we're not concerned if we have a name that makes no sense to anyone else, because they're not our target market.

[SPONSOR MESSAGE]

[00:14:26] JM:

[INTERVIEW CONTINUED]

[00:14:26] JM: Bridgecrew is a developer-first cloud security platform. In addition to helping teams address cloud resources in production , Bridgecrew shifts cloud security left by scanning for fixing and preventing missed configurations in the infrastructure as code level. We've done a show about Bridgecrew, and it supports AWS, Azure, Google Cloud, Kubernetes, Terraform, many other things. Bridgecrew makes cloud security accessible to developers by integrating earlier in the development lifecycle, VS VCI/CD, and implementing fixes directly back into developer workflows via pole requests.

You can go to bridge crew.io/sedaily to learn more. That's bridge crew.io/se daily. Checkup Bridgecrew, and it would help out the show if you go to Bridgecrew.io/sedaily.

[INTERVIEW CONTINUED]

[00:15:21] JM: You have a quote on the Weights & Biases website. Those who do not track training are doomed to repeat it. What does that mean?

[00:15:29] MR: That’s a controversial quote. I’m glad you found that quote. The idea behind a lot of what we do at Weights & Biases is around what's called reproducibility in machine learning. So maybe some of your listeners will have heard of something people call them reproducibility crisis. And this is a problem that I think is particular to machine learning, although with some of my testing, I have issues with this with regular software. But it's much worse than machine learning.

Unlike with writing conventional software, machine learning can be quite nondeterministic. And so it can be quite hard to reproduce the models that you built in the past. That's a huge issue, right? It’s a safety issue, frankly, because if you're not careful, it can be very hard to kind of even reproduce the models that you may be deploying to something super important like a self-driving car or some kind of health application. But it's also a practical issue. Just for doing testing and understanding what your models are doing, right? If you don't know exactly what went into the training of a model, it can be very heartsick kind of reach conclusions about what to do next. Or if you want to reevaluate some conclusion made in the past, it can be hard to figure out what was going on.

What we say is that you should carefully track your training or else you're going to end up doing a thing that everyone in the field does, which is just rerunning your model training over and over. In the past, rerunning model training wasn't maybe not such a big deal. I would argue, it was never a really good thing to do. But as models become more and more expensive to train, like some of the open AI models, like GPT3 that we're hearing a lot about these days require millions of dollars of compute just to train one-time.

I mean, I think a more typical model might be maybe thousands of dollars of compute to train. But nevertheless, you shouldn't be really ever retraining your models. You should be carefully monitoring everything that went into your training and learning from that training. And that's a lot of what the Weights & Biases tools help you with, is just passively tracking all the things that went into the training so that you don't have to do your training over and over again if you have a new question about it or you're not sure exactly what you were doing six months ago when you train the model that suddenly becomes important because it's getting deployed into production.

[00:17:49] JM: Can you say more about the tooling that machine learning practitioners need and what you envision needing to build at Weights & Biases?

[00:17:57] MR: Yeah. I mean, I think there're a lot of new tools necessary, because I think that machine learning differs in some fundamental ways from conventional software development. I know not everyone agrees with this. I think some people would say machine learning is just kind of another sort of sub-branch of software development, and there's truth to that. But my perspective is that machine learning differs in some key ways. So the fundamental difference here is that instead of writing code that does something, you're really writing code that generates code that does something, right?

So the code that you write as an ML practitioner looks a lot like kind of standard software development, but the code that gets generated by the code that you write, that gets trained some set of data and then deployed, that's quite different, right? Just some simple examples, right? So the models that you train, the code that's automatically generated, these files tend to

be much bigger, right? A model, unless it's trying to be small, trying hard to be small, could be hundreds of megabytes or even gigabytes of essentially binary data.

First of all, just versioning those models can be tricky, and you can imagine, because you're automatically generating these models, you'd probably end up generating a lot more of them. So the path is a lot different than conventional software development. So people try to check these models into Git, and they're too big, and they don't branch in the same way that normal software development does. And they don't really dif in the same way either, right?

You could train the same model on two different days on the exact same data, but slightly different graphics cards used for the training. And you'll end up with models that are essentially doing the same thing, right? Practically, they're the same model, but every single number in that model, every single piece of data in that model is going to be different.

The dif would be the 100% dif, right? Whereas like with software, typically you'll have these versions that kind of dif nicely. So all those things kind of point to the fact that version control for these models is quite a different animal. And then debugging is totally different too, because you can't really step through the code in the same way that you can step through code that you train with kind of conventional software development, right? These models aren't trying to be built for anyone to understand them. In fact, they're sort of like famously more and more understandable.

So you see a lot of company that talk about explainable AI, and you might think that problems are getting solved. But actually these companies exist because this problem is getting worse and worse and worse. Kind of in every step, we sort of see that making the model is more complicated, deeper they call in the field actually makes them harder and harder to explain. So the debugging gets super tricky.

And even the sort of like CI is a totally different animal, because you don't always expect the model to always be perfect, right? It's tough to have tests in traditional software development where you'd expect 100% pass rate, or you wouldn't deploy the thing. That can be unrealistic for the types of models that you want to build if it's doing kind of tricky visual recognition and you

have millions of test cases. You sort of have to do some kind of statistical thresholding versus the opposite.

So I kind of think that actually for lots and lots of things where there is an analog in conventional software development, the problems is the same with machine learning, but the solution might be different, which is why I think you see lots of new tools developing around machine learning production workflows in particular.

[00:21:17] JM: If I make a dashboard for my machine learning models, what is that dashboard look like? What kinds of materials and information do I want in my machine learning visualization?

[00:21:29] MR: That's a great question, and it definitely depends on the application that you're doing. So maybe let's pick a specific application in particular just to make it concrete. Why don't we talk about, say, you're trying to build a robot that can move around autonomously and not crash into things. So maybe you're making a model for that robot that just identifies the things in the world as it moves around. There are different things you'd want to see on your dashboard.

One thing that you're going to want to see is kind of all the inputs into your models. So when you build these models, you make a lot of choices. These things are called hyperparameters. And a typical hyperparameter might be like something called learning rate, which is like how fast I want my model to learn as it gets new training data. But in a real-world case, there'd be lots of these kind of hyperparameters that could be hard to interpret.

I guess another one might be the resolution of the images that I'm feeding into my model. And so one thing that you'd want to look at is these hyperparameters. And then another thing that you want to look at is kind of how the model performance changes as it trains. So typically, as you train your model, you're going to be actually testing the performance of the same time to know if it's getting better or worse, and this is something that you as a human really want to look at, right?

I mean, one simple reason you want to look at it is to know when to stop the training. Because at some point, your model is going to stop improving as you feed more data in and you just want

to go and want to stop the training so that your computer can do something else. But there are more subtle things you might want to look at. So people want to look at what they call the gradients of their model. And so this is something that sort of shows how much the parameters are changing, the individual parameters and kind of like at what points the model the parameters are changing.

You might expect that the parameters that are kind of closer as the inputs would be changing the most in the beginning. And then over time, it'd be the parameters deeper in the model that would be changing more. So that's something people kind of look for for diagnostic reasons. And then there're things that you want to look at to just get kind of a feel for what the model is doing. So it's dangerous to kind of just look at the statistics of the model. It's often a really good thing to just look at as the model train some particular data points and kind of see what the model is doing. Because machine learning is right for kind of subtle problems.

I mean, I remember I was training a model in my little robot maybe about a year ago, and it was having trouble with the images just that's it looking at around my room, and it is especially having trouble when the images weren't centered. And then I learned, actually, just from looking at the data. It was just anything the object wasn't right in the center of the image, it was having trouble with it. And then actually learned that the sort of standard image datasets are collected by humans that are mostly centered, and it's kind of a common phenomenon that robots get confused when they're trained on data off the Internet.

So there's sot of this mix of like specific examples that you want to look at, and then like lots of graphs of different things that are happening in your model. You also might have different things that you're optimizing at the same time. So like a typical self-driving car company training this kind of model, they would look at not just the accuracy of the model, but they would look at the accuracy of the model on pedestrians, and the accuracy of the model on bicyclists, and maybe the accuracy of the model in the snow. And they would be creating a dashboard of all these things, because obviously a highly accurate model that just misses the rare case when there's a pedestrian on a bicycle is not a model that's safe to deploy. It's really important to build these big dashboards where you can see everything going on.

And then they get even more interesting, because when it comes time to decide what model to deploy or try to figure out what's the right, say, learning rate, you want to look at lots of models together. So people build these sort of meta-dashboards where they compare two or even hundreds of thousands of models together to try to find patterns and what's the best model to deploy or what's the sort of like best set of inputs that's likely to make a model that's safe to deploy.

[00:25:28] JM: All right. Let's get into a specific engineering term for machine learning research. What is a hyperparameter search?

[00:25:38] MR: Good question. Hyperparameter search, it's something that machine learning researchers do where they try to find the best set of what's called hyperparameters for their model. Now, I need to really define what hyperparameter is. When you build a model, when you build a machine learning model, actually what you're really doing is searching for the best parameters. So it's sort of like the best numbers that's going to kind of make this machine learning equation do the thing that you want.

But then there are questions like how do you pick the best parameters? And those things are called hyperparameters. So I keep mentioning learning [inaudible 00:26:12] intuitive one, which is like every time a model sees a new piece of data that it hasn't seen before. Maybe it's confused about how much should it update its beliefs about the world.

So you if you have like lots of it that it's similar, like a high-learning rate might be fine. But if you have data that's quite diverse or maybe there's some mislabelings in your data, you might want to use a lower learning rate, because you might not want the model to overreact to one piece of data. It could be there's something mislabeled about it or just something weird about it.

And so as you kind of build models, you might wonder what's the best learning rate to use. And so what a hyperparameter search would do over a learning rate is it would try to actually build models with different learning rates and then it would look at how that set of data and see, "Okay, which of these learning rates led me to the most accurate model on a kind of held out set of data?" So you'd say, "Okay. You know what? I have a high-learning rate. Maybe it's 80% accurate. When I have a lower learning rate, maybe it's 85% accurate. But then if it's too low,

maybe the model, that accuracy starts dropping." So I should use that kind of sweet spot of a middle tier learning rate.

But, of course, in the real-world, you have thousands of these kind of input parameters and you don't have infinite computing resources. So it can be tricky to figure out what's the best set of learning rates to try.

[00:27:28] JM: We've done a few shows recently on hyperparameter tuning and hyperparameter search. Can you explore the process of looking at different hyper parameters in more detail and how this affects a machine learning development process?

[00:27:43] MR: Yeah, that's a good question. I think – I mean, in the real world, hyperparameter tuning is not as pure as a it kind of sounds, because the compute resources can be so expensive. You might think like, "Well, just kind of one time, you try every set of hyperparameters and you try a ton of hyperparameters and then you deploy your model. But in the real-world, it can get a lot messier than that, because one thing that's happening is you're often getting new training data. For lots of locations, again, say you're a car company. You're probably collecting new images from your test cars or maybe even your customers' cars, and those are getting labeled. Or for any application, you're often collecting more data and your requirements are changing and you kind of want to look at the hyperparameter searches that you did in the past to build an intuition about what hyperparameters are likely to work. You wouldn't necessarily want to try every learning rate every time. You might decide, "You know what, like, typically, on this application, lower learning rates tend to be better. So I'm just going to try kind of a smaller range of lower learning rates."

We have seen kind of more earn more companies actually, like Google's Auto ML, or I think Data Robot has a tool like this that this completely automates the whole process. It just finds the best set of parameters. I think one reason that people don't use those as much as you might think when they certainly sound useful. They are useful. But an issue you can be that there's just so many hyperparameters. Actually, there's kind of more and more hyperparameters.

I'll try to give you some other kind of interesting examples. One thing people might do is they might decide that, "You know what? If I rotate the image a little bit, I can kind of feed that

through the same label. But you don't want to rotate the image too much, because then maybe you're really distorting the data. So there's sort of like some randomization of my input images. That's called data augmentation. And then you have all these new hyper parameters. How much do I rotate an image? Maybe I squish the image a little bit.

And suddenly I have this whole new set of hyperparameters to try. So I think typically there sort of like a research phase where people are trying big sets of hyperparameters, and then may be a production phase where before each model gets deployed, there might be like a smaller hyperparameter search over some subset of the possibly useful parameters before the algorithm is deployed. But I would say, we see from our customers quite a range of feelings about hyperparameter search.

I mean, some really like to do these gigantic hyperparameter searches to eke out the last tiny bit of accuracy in the model. And then some of our customers think that that's like really wasteful of compute resources and also it kind of keeps you from really developing deep intuitions about your data which they feel as important. So they will do much less emphasis on hyperparameter search.

And I think is also kind of a dichotomy. Some people do the process in a very, very organized way where they just kind of let the computer run for months and then spit out a number, whereas other people are kind of watching the process and tweaking it and making changes.

[00:30:40] JM: Describe the ways in which a machine learning experiment can be parallelized and how your tooling helps with that.

[00:30:45] MR: Machine learning can be parallelized in lots of different ways. I mean, I think it's actually one of these sort of embarrassingly parallel problems. There're sort of many different points where you can parallelize it. So one point where you can parallelize it is actually in the hyperparameter search itself, right? If you're training different models to see what the best set of hyper parameters is, you can actually kick off lots of runs at the same time and you could just run those on completely different machines typically.

Now it's not necessarily ideal, because in a more advanced hyperparameter search, you'd want to be able to learn from each run so that we try a better set of hyper parameters each time. In the academic literature, typically, the hyperparameter searches people talk about that are smarter using strategies that you might've heard of called Bayesian optimization. That sort of uses the past runs to the side of the next [inaudible 00:31:33] used to do.

So if you parallelize those strategies, if you just ask the Bayesian optimization algorithm to give you the best run to try next based on some past data, if you parallelize that, your algorithm might just tell you to try the same set of hyperparameters and times that it's a huge waste. So one of things that our software does, and it's not the only software to do it, but we do do it as we help you. If you know you're going to run 10 runs in parallel, our hyperparameter optimization software will help you pick 10 runs so that they're kind spread out over this space. So you try kind of 10 different things as supposed to trying the same thing 10 times.

But I would say that's a very good point to parallelize, because you can even kind of change your parallelization over time. So if you have lots of new machines available, it's pretty easy to kind of spin up to the lots more hyperparameter search runs at once. And then if you're more constrained resources, you can shut down these machines without too much cost. So I would say that's kind of the easiest place to paralyze things.

Then kind of in more advanced research topics, you can actually try to do your training on multiple machines at the same time. And that is kind of an active area of research, but a lot of our customers do do it. We actually don't help you do that, because there's good frameworks like PyTorch and TensorFlow will help you with that. But what we help you with is monitoring. And the monitoring actually can get really unwieldy as you starts to run on lots machines at once. And it's also really important, because these budget start to get expensive. So if you're spending up a whole bunch of machines with lots of GPUs, you're going to be chewing up – I mean, many dollars per minute.

So you're going to want to know if something is going haywire. At some point up of parallelization, it's virtually guaranteed that a few of your machines are going to be acting weird. And so our software help you pinpoint that super quickly.

And then there's even sort of like a finer grain parallelization, which I would say it used to be a research topic. I think now it’s becoming just sort of standard practice. But it's also a bit of a pain still, which is parallelizing across a single run across multiple GPUs. So a typical ML training machine might have 8 GPUs on it. So people want to kind of have each of those GPUs working at the same time on a single process often. And that's a little bit simpler than parallelizing across lots of machines at once. I think those are sort of the three places where people parallelize. And a lot of people do all three of those at once.

[SPONSOR MESSAGE]

[00:34:02] JM: As your infrastructure grows, you have SSH host and Kubernetes clusters in staging, production and maybe even on edge locations and smart devices. You want to implement the best practices to access them all, and that can be time-consuming. Security tools can get in the way of engineering work. Well, there is a tool called Teleport. It's open source software. It's written in Go. It's a drop-in replacement for open SSH. It also has native support for Kubernetes. It's built by Gravitational.

Teleport provides identity-aware access using short-lived certificates with SSO, session recording and other features, and it ensures compliance and audit requirements. Teleport also prioritizes developer and convenience, because it's built by engineers for engineers. You can give it a try going by going to try.gravitational.com/sed. There are links to downloads, documentations, and a GitHub repository. That's try.gravitational.com/sed to try out Teleport from Gravitational.

Thanks to Gravitational for being a sponsor. And again, if you want to try it out, you go to try.gravitational.com/sed.

[INTERVIEW CONTINUED]

[00:35:18] JM: Is it hard for a machine learning engineer to program the parallelism or is it just like a configuration setting to easily parallelize?

[00:35:28] MR: That depends on who you ask. So the software does have configuration settings to help you with this. TensorFlow and PyTorch will help you, but I think that it is pretty tricky, because to do it right often requires violating some of the assumptions that are in the academic literature. So I think until recently, a lot of the ML papers would assume things like kind of when you – Like all your data is getting fed into your model one at a time. That was kind of an important assumption where they actually call it a batch. It'd be like basically your model gets a batch of data and then it updates and then it gets another batch and then it updates.

If you're on different machines, it's not like realistic anymore. And so your kind of model is like changing parameters based on data in sort of an unknown sequence. There's sort of like this academic question of like what should the learning rate be. How should you train models in this world? Then there's also this, I guess I would call a kind of a DevOps question of like is this like same thing to do? Where are the bottlenecks in this communication? Is it actually kind of too expensive to communicate across these machines to even make it worth it to run it on multiple machines? And I think that's where you hear a lot about ML ops these days and a lot about kind of collaboration between DevOps and machine learning.

I mean, I will say I kind of came through more of a machine learning point of view on this. And so kind of setting up networks for multiple machines to communicate with them quickly was really not obvious to me. And it's been really fun. I have excellent – One of my cofounders, Sean, was not my cofounder in previous company, but is my cofounder at Weights & Biases, and he's really good at this. And it's been really fun to kind of learn from him how to set all these stuff up to work really well. I mean, he kind of find it trivial, I feel like, to get this distributed training happening. But it's definitely not trivial for me. So I think this is one of these areas where people need sort of a pre-diverse set of skills. And I think it's really useful for kind of cross-functional collaboration.

[00:37:22] JM: How does data augmentation work or auto generated data? Does that actually provide value to training a machine learning model? If I take some data and make some synthetic data to augment my current dataset?

[00:37:38] MR: I will say it is an open research question. But I think the trend is that it definitely can help. So there are sort of different forms of this, which I think are sort of different levels of

certain that it works, right? The thing that definitely works on images and audio is kind of slightly modifying the data.

So one thing that machine learning does, which can be really frustrating for humans, because humans generalize incredibly well. We don't kind of hone in on one specific part of the data and just focus on it. Machine learning is very willing to kind of hone in on a specific thing of the data and just sort of over emphasizing, right?

If it's like always a black pixel in the upper left when you're looking at the toaster, it can be pretty hard to get machine learning algorithms to not just always think, "Okay, if I see a black pixel in the upper left, it's definitely a toaster," when that was just weird artifact of the data. And there's lots of different ways to kind of combat that. I think a lot of machine learning really is designed to kind of help with that problems. It's called over-fitting. But one of the best way is actually data augmentation, and it can be very intuitive. So like I was talking about sort of rotating a data a little bit. So like if it's a picture of a toaster and then I just rotate the image 10°, it's definitely a picture of a toaster.

So that can be a way to make it seem like you have lots of our training data than you did. So if I have a finite about of labels, if I start rotating the images or stretching the images or slightly changing the colorization of the images, that can make it seem like I have actually lots more data. And I think that's definitely a good idea. You can mess things up, right? If I turn the toaster completely upside down, that might have an unintended consequence. But for like a satellite photo, you can rotate the image 360° and it's probably the same thing. So it requires some domain expertise and some care, but I think data augmentation would be considered definitely best practice in machine learning.

The thing that I think is still somewhat controversial that a lot of people are experimenting with it, and I think there's some success cases, is generating synthetic data. The idea here is like video games, for example, are so realistic that why not use them as inputs for machine learning. And the thing in a videogame is that you actually know what every pixel stands for, because you know the underlying mechanism that say generated a scene with like a road and a car. You know which pixels correspond to a road and a car. So instead of taking a photo and like kind of

having a human painstakingly label it after the fact, you have a graphics engine that generates a road and a car and actually generates the labels at the same time.

So in that way, you get infinite numbers of training data. It's sounds really promising, especially since a lot of companies will spend millions and millions of dollars on data labeling. And I would say it's starting to work. I did a little bit of research on this myself at OpenAI briefly, and some of the work that I worked on a little bit, and OpenAI worked on a lot actually showed that synthetic data alone could help a hand learn to manipulate a Rubik's cube, which is pretty cool. And I think most self-driving car companies these days are looking into this. But I wouldn't say that generates synthetic data is standard best practice yet, but it seems promising and interesting.

[00:40:39] JM: There are numerous machine learning frameworks, TensorFlow, PyTorch. Do you have any strong opinions on what framework should be used and for what kinds of applications?

[00:40:49] MR: That's a good question. I don't have strong opinions. And maybe I shouldn't have strong opinions as someone that makes tools that kind of support all the frameworks. But I'll tell you sort of the – We see people from all these different frameworks in our tools. I kind of tell you this sort of the ontology of them, I guess.

So TensorFlow and PyTorch are comparable, and there's – It's a little bit of like a VI versus Emacs thing. They've kind of, at this point, copied all of each other's features in my view. So at first, PyTorch was sort of like what they call dynamically generating the compute graph and TensorFlow wasn't, but then they both sort of – Now they both do both. And so PyTorch is thought to be a little bit lighter weight, maybe a little bit more popular among researchers. TensorFlow is thought to be a little bit more heavyweight, maybe a little bit more performant. But it does seem like the trend over the last year has been people moving from TensorFlow to PyTorch, but unclear. I think that's still unsettled, which is the better framework. But they are two most low-level framework. So they're basically [inaudible 00:41:46] for like kind of doing the math underlying machine learning.

And then Keras is kind of a higher level framework built on top of TensorFlow. So Keras kind of makes TensorFlow easier to use, but it's not like a training wheels. It's more like something that

you just want to use if you have normal applications. And so I would always recommend Keras and the same way that you'd normally recommend somebody uses Python instead of C.

And PyTorch now has a couple different higher-level things built on top of it. There's one called FastAI. There's one called PyTorch Lightning, and there's one called PyTorch Ignite. I think the jury is still out on which of those is going to become dominated. I think PyTorch Lightning is getting very popular. And FastAI was actually built for a course, although it's kind of a cool standalone framework also. And a version 2 is coming out, which people are pretty excited about.

And then sort of orthogonal to that off to the side, there're other frameworks, like Scikit is an old framework and beautifully done framework, but it doesn't handle deep learning that well. So for kind of all non-deep learning applications, you're probably better off using Scikit with the one exception of when you're kind of handling very high volumes of data, Spark has an ML framework that works really well Spark if you're already using that. And then there's also some specialized framework for what's called boosted trees that work well in certain applications. So there's something LiihtGBM and XGBoost. And I would say that covers all the frameworks we regularly see. There's always kind of new ones coming along if there's new programming languages. But that would cover 99% of these cases we've seen.

[00:43:16] JM: Do you have any idea what kinds of tools we need to help debug machine learning code?

[00:43:22] MR: I mean, I think it's tricky. I think we haven't necessarily seen all the tools there might be yet. One of the differences about debugging machine learning code versus debugging conventional code is that when you debug machine learning code, you need to look a lot at sort of like the aggregate effects of what the code doing. So I feel more like a scientist, I think, debugging machine learning code versus debugging my non-machine Christian learning code.

So when I write like some frontend React stuff, I'm mostly kind of walking through what's happening. And I guess there are certain types of code that also feel like this. But machine learning is the most extreme, where I'm typically looking at kind of statistics on what's going on. And so I think Weights & Biases has an important role here to play for showing people what's

going on to help them debug their code. But there might become of smarter things that sort of help people point exactly to where the problem is. We haven't really made something like that yet, but I've some interesting research on it. And I think a lot of this stuff around sort of explainable AI might help a lot with debugging. But I guess the sort of the state of affairs right now is that debugging machine learning is really harden and people should definitely use like any tool they can possibly find that might help them with that.

But I also think when you're trying to deploy machine learning, because the debugging process is so hard, you really to budget a lot of extra time to getting that machine learning model into deployment. And that's the world right now as of what? July 2020. Things will certainly change rapidly in the field.

[00:44:52] JM: July 2020, one if the elevated projects right now is GPT3. Have you looked into GPT3 in much detail?

[00:45:01] MR: Yeah. I mean, of course I've seen all these stuff on Twitter. And OpenAI has been a long time customer of Weights & Biases. So you we get excited about all these stuff that they put out. I don't have any kind of deep insights on it, but I guess I will say the thing that I think everyone noticed, which is that it's pretty impressive what it can do. I guess I started out in linguistics where people really wanted sort of lots deep linguistic insights to build models that generate human looking language. But it sort of seems like the trend is just these very sort of almost like blank slate models do an unbelievably amazing job of mimicking human writing.

[00:45:36] JM: What impact do you think GPT3 will have? I mean, there's a lot of excitement around it that has tons of potential real-world applications. Any perspective there?

[00:45:45] MR: I mean, I'll say my experience of GPT3 in general is that I think GPT2 is also quite amazing. And I was surprised. This came out about a year ago with maybe a little a less fanfare than GPT3. But in my view, it's almost as good and quite impressive in generating human-looking text. But the thing that I think is maybe the most interesting is that we haven't seen a lot of these cases come out yet. So like you would think that being able to sort of synthesize human text would lead to tons of applications, but maybe it's not yet kind of quite good enough to really make a realistic chatbot. Or maybe it is.

I mean, I think once kind of computers can chat with you, and you can't tell the difference between a human and a computer. It does feel like we're in a totally different world. But I would say that's not like my area of expertise. That's sort of me pontificating on a topic.

[00:46:33] JM: A few questions on company building in the machine learning tooling space. So you don't have one specific tool that you're offering with Weights & Biases. It's more like a collection of tools. Why are you taking this approach in company building? Like the typical company building approaches, you get some very specific wedge into a market and then you find up cells and you find lateral products. This feels like more of like a collection of somewhat disconnected products. Do you feel that way, or maybe you could tell me your strategy when it comes to machine learning tooling.

[00:47:07] MR: Totally. I mean, I do think that the tools are pretty connected. So we try to make all the tools work together super well. But we also don't want people to be kind of forced into like a Weights & Biases-only ecosystem. And I think that was really just because we were listening carefully to what our customers were asking for. There are a lot of companies out there that do a complete kind of end-to-end machine learning platform. And I think that sort of makes more sense as an investment, right? You'll probably get more VCs to purchase that. But the challenge with that is that customers don't want it.

So what we wanted to do is basically build the thing that customers want. And we think that what they want is a set of tools. And we sort of have a wedge. I'd say that our first tool that we've built, the experiment-tracking tool has been super popular and it's kind of what we are known for. So the way could think of us using that as a wedge that kind of encourages people to use other tools. But I mean a wedge sounds a little aggressive with customers. I mean, I might it's like a really useful initial thing that starts a conversation. And then the space is moving so fast that we are able to build our other tools kind of in conversation with our customers.

[00:48:15] JM: Last question. What is the biggest technical problem that you have right now at Weights & Biases? What's the biggest unsolved technical challenge?

[00:48:22] MR: I would say the problem that feels like hair on fire right now I think is – And this is sort of a general answer. Maybe it's a lame answer, but it really is scaling. We never had my last company, CrowdFlower, like a tool that tens of thousands of people used every day. I'd say at Weights & Biases, we didn't expect the growth for the experiment tracking that we've had. We also didn't expect the different ways that people would use our tools. So people are logging a lot more data than we thought and in the lot more diverse ways.

Literally, just kind of making the visualizations fast off of the data that people are streaming into us is probably the biggest challenge that the technical team is thinking about every day. I mean, of course, we also think about like how to make our interfaces intuitive and nice, but those are less technical and kind of more UI. The big technical thing is like our tool can never go down. It can never lose people's data. Those are hard requirements. But then the piece that I think we would keep thinking about is how do we make this fast and how do we make kind of more the use cases fast?

I'll give you an example. I mean, we thought people might log thousands of hyperparameters, but we weren't prepared for people logging millions of hyperparameters. But that's important to some. Or we thought that people might run kind of thousands of runs in a hyperparameter search. But you OpenAI will typically run millions of runs in some of their hyperparameter searches. And so this sort of like difference and sort of massive extra use that we weren't expecting has created a whole bunch of database optimization challenges.

[00:49:54] JM: Okay. Well, Lukas, it's been great talking to you. Thanks for coming on the show.

[00:49:57] MR: Thank you so much.

[END OF INTERVIEW]

[00:50:07] JM: Are you spending too much time debugging in production? If you are, the days of endless logging and debugging sessions should be long gone. You can give your engineers 40% of their time back and fetch data instantly from production systems with Rookout. To check it out, visit rookout.com/sedaily. That's rookout.com/sedaily.

[END]

