EPISODE 570

[INTRODUCTION]

[0:00:00.3] JM: Rajat Monga is a direct of engineering at Google where he works on TensorFlow. TensorFlow is a framework for a numerical computation developed at Google. The majority of TensorFlow users are building machine learning applications such as image recognition, recommendation systems and natural language processing, but TensorFlow is actually applicable to a broader range of scientific computation than just machine learning. TensorFlow has APIs for decision trees, support vector machines and linear algebra libraries. The current focus of the TensorFlow team is usability. There are thousands of engineers building data-intensive applications with TensorFlow, but Rajat and the rest of the TensorFlow team would like to see millions more.

In today's show, Rajat and I discussed how TensorFlow is becoming more usable and more accessible and we also discussed some of the developments in TensorFlow around edge computing, TensorFlow hub, which allows people to share modules of their TensorFlow applications, and we also talked about TensorFlow.js, which allows TensorFlow to run in the browser. We'll have an upcoming show about TensorFlow.js where we'll do a deeper dive in the near future.

If you want to hear the previous episode that we recorded with Rajat as well as some other episodes about TensorFlow, Keras and other related topics to machine learning, we have all of our episodes in the Software Engineering Daily apps for iOS or android. We've got tons of episodes on blockchains, distributed systems, business, lots of other topics. You can also find all these episodes at softwaredaily.com. If you want to become a paid subscriber to Software Engineering Daily, you can hear all of our content without advertisements and you can subscribe at softwaredaily.com. Also, all of the code for our apps and our websites, it's all open source. If you're looking for an open source community to be a part of, you can come check it out at github.com/softwareeningeeringdaily. We'd love to have you as part of our community.

Thanks for listening, and let's get on with this episode.

© 2018 Software Engineering Daily

[SPONSOR MESSAGE]

[0:02:16.1] JM: This episode of Software Engineering Daily is sponsored by Datadog. With automated monitoring, distributed tracing and logging, Datadog provides deep end-to-end visibility into the health and performance of modern applications. Build rich dashboards, set alerts to identify anomalies and collaborate with your team to troubleshoot and fix issues fast. Try it yourself by starting a free 14-day trial today.

Listeners of this podcast will also receive a free Datadog t-shirt at softwareengineeringdaily.com/datadog. That's softwareengineeringdaily.com/datadog.

[INTERVIEW]

[0:03:02.6] JM: Rajat Monga, you are the director of engineering at TensorFlow. Welcome back to Software Engineering Daily.

[0:03:07.7] RM: Thanks, Jeff.

[0:03:08.5] JM: The last time you came on the show, we talked about some of the basics of TensorFlow and your work on the framework. Today I'd like to talk more about the progress of machine learning and how you are seeing that through the lens of your role on the TensorFlow team at Google. Let's just talk about machine learning developments to start with. What are the machine learning developments in the last year that have surprised you?

[0:03:36.9] RM: I don't know if they're surprising. They're all making great progress in the direction we started in, the pace of progress is fast of course. A number of exciting areas that have been some of the progress in robotics where you can now learn from other humans, you can learn from simulations rather than needing lots and lots of data. Progress in areas where you can learn with less data has been very, very interesting. The progress on the generative models with the — Especially the generative adversarial networks and the variations of that has been really great and we're seeing some amazing things that they've been able to create as well.

Then we are starting to see very exciting — Again, from a research perspective, exciting ways where people are trying to mix this with other things that people do. For example, in the music domain, the Magenta Project has made a lot of progress in combining some of these creative techniques from machine learning with artists and see how the two can work together and come up with very interesting pieces of music.

[0:04:46.1] JM: You gave a number of interesting developments. Magenta is particularly appealing to me as a musician. I've talked to the Magenta Team and I find that project particularly appealing because I see it as a — That's a place where it leverages the human computer interaction side of things. They seem to really be exploring how can a machine learn quickly from human and learn to collaborate with the human more effectively.

But on the TensorFlow side which is where you're focused, the objective of the TensorFlow team over the last year has been to make the framework more usable. Why has the goal of usability been a top of mind for the last year?

[0:05:31.2] RM: I think where we started out was enabling new kinds of research, and that continues to be important. We definitely want to make sure machine learning continues to progress. But then beyond that, one of our big goals has been really to bring all these exciting developments, exciting advancements to many more people across the world. To do that, we really need to make software. We need to make tools that are really, really easy to use, right?

So with TensorFlow, the goals have been you don't necessarily need to understand all the math. You don't necessarily need to understand even back propagation because a lot of that is taken care for you. Then just in terms of the library, the API, going from the lowest level where you have full control, we're trying to go higher and higher where you get to do things and then slowly you can — As you learn more, you can do more interesting things or more different things. But just to get started, it should very, very easy.

[0:06:31.6] JM: What are some of the more specific usability efforts that you focused on?

[0:06:37.5] RM: So I think a couple I'm really excited about. So one is what we call eager execution, and what that does is what TensorFlow — What it started with was this idea of you

build a graph and then you execute it and the graph represented the machine learning model. The reason we did that was that really allows us to optimize it using a lot of the compiler techniques and really scale it out across large scale distributed systems and so on. Now, that's great, but what that does is when a developer starts out, they have to think about that graph and think about it that way.

With this new idea of eager execution, the idea is you don't have to build a graph, you're just programming. If you're familiar with Python, you're just writing Python cord and that internally just executes TensorFlow cord and is optimized for CPUs, GPUs, TPUs, whatever, and then — So it takes away one of the things that you have to think about earlier and just makes it a lot easier and more natural in Python itself.

Now going from there to scaling, you still have all the good stuff that you used to have at TensorFlow, so you can still use pretty much the same cord and create a graph from that instead, or scale it out to a really large cluster if that's what makes sense. But it really allows you to cut that friction down and starting out. So that's one.

Another interesting piece has been just the high level APIs we started to invest in building these ideas of layers and models and so on. What we realized — and we have François Scholly here as well who's the author of [inaudible 0:08:15.1], and that was an exciting way for our developers to access these models to access deep learning as they're getting started. We really worked to integrate that right into TensorFlow and make it really easy to get started with that as well.

I think for typical Python developers, those are two big things. I would say there's a different class of developers almost, some people who are like different languages, JavaScript being one, especially on the web, and that's another area that I'm happy to talk about. There's tons that's happened there as well, if you brought TensorFlow to the web browser as well.

[0:08:50.8] JM: Yeah, we could just jump in to TensorFlow.js. That was a recent announcement, and this allows for training and deployment of machine learning models in JavaScript, but what are the domains in which people want to use JavaScript for machine learning?

[0:09:07.6] RM: I think we are just starting out. So there are a number of things that we can think of, right? Often, we push — We've seen overtime that people want to do machine learning outside of the data center as well. Over the last couple of years, we have seen people push these things on phones and so on.

Over the last year, some folks, some developers on our team said, "Okay. What if we could do more in the browser? What if we could accelerate it?" There were some libraries before that would do the basic math, but what these guys did was not just take those ideas that we did in the data center, but also accelerate them in the browser using the GPU or whatever you have on your machine. So it can go pretty fast.

Now, with that, what we've seen is areas where — The first one that comes to mind is you want to get started. You want to teach people something. You have demos to show. You can include ML and all of those. So we have some very exciting things that people have built. People have built games in the browser using different kind of gestures, etc. Also, once it's in the browser, it has access to your camera if you wanted to. It has access to your audio stream if you wanted to, and you can do pretty interesting things in combining all those together.

[0:10:26.2] JM: Is that important because you can have learning take place on the client side device and not necessarily have to shuttle the data back to the server?

[0:10:37.8] RM: That's definitely a part of it, I think. The other part I would say is the fact that you don't have to install anything. You just go to this website and it works. It's a no-install. Completely, you go someplace and it's just working. I think all of that, again, reduces the friction for a developer and beyond that, the users as well to get started, which is a huge value.

[0:10:59.8] JM: When you say TensorFlow runs in the browser, does that — Well, or TensorFlow.js, it runs in the browser. Does that meant that these — What had to be built in order to implement that? because I know TensorFlow runs in many different languages, but I'm wondering what that actually means. Does that mean that you can write the code for models in many different languages or that you have defined interfaces for compiled models to interface with those languages? What exactly do you have to define to say that you've ported TensorFlow to a different language?

[0:11:38.3] RM: Right. I think there are a number of things that happened with different languages and browser is somewhat special in some ways. Often, what we do for different languages and to people who have used Scala, Java, even Julia and many other languages really who do that. What we often do is TensorFlow has an API that offers access to all the operations that it provides, offers access to just execution of these operations or building graphs and all of that stuff.

Now, often, for these languages what that means is they create rappers around these functions and these operations and make them available in that language so people can then build on top of that, people can again build their own models there, or more often just models that have been trained, say, using Python, the more common one today, and then take those models and really execute them from a different language. Now, all of these typically uses the same TensorFlow runtime and the backend that we have, which can scale and do all of that stuff.

In the browser, the interesting thing is you don't have access to all these other backend, because that's in C++, the browser doesn't allow you to do it for good reasons, for security. So in the case of the browser, we actually took the same APIs that TensorFlow has in many other languages and really wrote a custom backend or a custom optimized operations for these that can run in the browser and that are potentially accelerated using techniques like WebGL, which give JavaScript code in the browser.

So on the JavaScript side and the browser, you can build your models with JavaScript or, again, you can take models that were trained in TensorFlow device, convert them using a tool and then run in the browser using all these APIs.

[0:13:35.4] JM: So when I build and train a model in Python using TensorFlow, the end result is a model that runs — It requires C++ to run. Is that what you said?

[0:13:51.1] RM: Right. Behind the scenes — When you write your code in Python, behind the scenes most of TensorFlow code is written in C++. It's all compiled, so you don't care. But that is a library or a binary that's running there and that isn't something that can be shipped to the

browser, but for most other languages, that's what it's used, because it really allows us to optimize that down to whatever we need to.

[0:14:16.0] JM: You had to write some kind of interpreter to translate that C++ code, the model that would run in C++ to some minified JavaScript version that could run in the browser?

[0:14:32.1] RM: It's actually a number of things there. One, in the browser, we defined a nice interface that the developers can code against. Then from taking the model and deploying it, the model itself is not in C++. The execution engine is in C++. One way I like to explain that is you could think of the model as your Java byte code. You write a Java program. In this case, you write often in Python, but it could be in Java too. Then when you sort of train the model or you execute it, eventually we get this graph, which is sort of our byte code or an intermediate representation, and that's represented in proto buffers typically along with some information and the weights in the data, etc. It's this intermediate representation that's really converted into an optimal format for the browser, and then the code in the browser actually interprets that.

[0:15:29.4] JM: All right. Okay, it's in a protocol buffer after you train the model. So in the browser, you need to define a way to interface with that proto buff. Is that what you said?

[0:15:41.7] RM: That's correct. Think of that proto buffer as really defining a simple program that represents your model and the browser has code that, interprets that basically and executes that model.

[0:15:54.8] JM: This reminds me of, I think, when I was reading about some of the other announcement, the recent TensorFlow announcements, there was TensorFlow Lite, which is another model for running TensorFlow in less resource, or a resource-constrained environments. I think in those environments, you actually want to define not a protocol buffer, but — Was it a flat buffer? Is that right?

[0:16:20.5] RM: That's right. That's another format that's used that's optimized for those environments that uses less memory. It's easier to get started. It's faster to get started and so on.

[SPONSOR MESSAGE]

[0:16:39.6] JM: The octopus, a sea creature known for its intelligence and flexibility. Octopus Deploy, a friendly deployment automation tool for deploying applications like .NET apps, Java apps and more. Ask any developer and they'll tell you that it's never fun pushing code at 5 p.m. on a Friday and then crossing your fingers hoping for the best. We've all been there. We've all done that, and that's where Octopus Deploy comes into the picture.

Octopus Deploy is a friendly deployment automation tool taking over where your build or CI server ends. Use Octopus to promote releases on prem or to the cloud. Octopus integrates with your existing build pipeline, TFS and VSTS, Bamboo, Team City and Jenkins. It integrates with AWS, Azure and on-prem environments. You can reliably and repeatedly deploy your .NET and Java apps and more. If you can package it, Octopus can deploy it.

It's quick and easy to install and you can just go to octopus.com to trial Octopus free for 45 days. That's octopus.com, O-C-T-O-P-U-S.com.

[INTERVIEW CONTINUED]

[0:18:10.1] JM: What are the tradeoffs you make — If you train a model and you throw it into a proto buff versus a flat buffer, what are the tradeoffs you're making?

[0:18:20.4] RM: So from the model developer's perspective, it really doesn't matter. They both represent the exact same thing. You can have the same kind of model running in the browser or on the device. The difference is more from — These are formats that were built for different reasons, and the reason the protocol buffers are interesting and why we started from them, they offer a lot of flexibility. We use protocol buffers for lots of different things and storing things where they might change overtime, so versioning, etc. They're really, really good at that.

Flat buffers still offer some of those advantages. They still have some of that flexibility, but they tradeoff some of the other flexibility for providing, making it really lightweight, because if you want to deploy to, say, a small phone or even a tiny device, you really want to optimize for that last bit of space. When you get started, say, when you load the model the first time, the way a

flat buffer is encoded, it's just much faster to get started, so the overheads are lower there. You're trading off a bit of flexibility for a bit more performance there.

[0:19:30.5] JM: I see. So it's not about execution as much as perhaps the format itself. So protocol buffers are used for microservices and all kinds of other things throughout Google and they're widely used outside of Google as well, but you may not use flat buffers for defining interfaces for your microservices. You want to use flat buffers more specifically for these kinds of domains, like resource constrained, machine learning execution environments.

[0:20:04.6] RM: Exactly. In fact, I believe they were built initially for game developers, for gaming kind of environments.

[0:20:11.0] JM: Speaking of devices and IoT machine learning deployments, are there devices on the market that maybe they're just type of devices where we would love to deploy machine learning models to them, but because of legacy technology, I'm thinking of industrial environments, maybe oil rigs or agricultural situations, are there computers running in these environments where we can't yet deploy machine learning models to them just because of the way that they compute infrastructure was built?

[0:20:46.2] RM: I would say not really. Today, where we are, I think we can deploy machine learning to pretty much every place you have a computer. Now, of course, depending on your environment how you take advantage of it and how it's connected to what you have there is different. If you in a legacy environment, if you don't have access to all the sensors that you're collecting data from on that computer, you'll not really gain much value out of it.

But that said, if you have a computer — And you don't need a lot of computing power for some of these deployments, you can really deploy things now. In fact, one goal that we have, we want to make sure that TensorFlow Lite was — This was a big reason to build TensorFlow Lite, that we want to make sure anytime, anyplace, wherever you are, if you have a computing device, you should be able to run machine learning on it, because if not now, you will want to do that in the future.

[0:21:45.9] JM: Are there other tradeoffs that the developer needs to keep in mind when optimizing for resource constrained environments, whether we're talking about mobile devices or sensors in an agricultural field?

[0:21:59.5] RM: Definitely. So from these devices perspective, if we're used to our workstations, or laptops, or even tablets, those have a lot of memory space and the kinds of things that they can do. As you start to go down to these smaller sizes and form factors, it's as if you are like 20 years ago where desktops were, and you're constrained. The amount of memory you have is much less, the amount of storage space you have is much less. So you have to be careful about what you do with it. You say on a workstation you'd be fine starting with, say, just writing a simple loop that just works, and maybe on your workstation, that takes a second or even less to execute.

Now, you run the same thing on this tiny device and it might take 10 times that because of the compute power, because the LAN capacity and so on. So you have to think about that as you write the code for these kind of environments.

[0:23:06.0] JM: TensorFlow started with an emphasis on deep learning and neural nets and now there's a wider array of libraries available. So within TensorFlow, you can interface with decision trees, and support vector machines and linear algebra libraries, and when I think about those features, it starts to look more like a flexible mathematical tool that's not necessarily constrained to machine learning operations. Do you see TensorFlow being a tool that could have the functionality of other scientific computing tools? I'm thinking MapLab, for example.

[0:23:45.8] RM: Absolutely. I think that at its core, when we designed TensorFlow, we wanted to layer things in. The core itself is really a numerical computation library exactly as you said. You can build lots of different things on top. Of course, the first thing we started with was deep learning in neural networks, but as you've seen, a lot of these different things have been built, some by us, some by other teams at Google, and many by folks outside Google as well, because they see this core computing library that offers a number of things. It offers a support for distributing across many nodes. So like in the scientific use cases, you want to use a super computer, you can do that with TensorFlow. If you want to accelerate your cord on accelerators, like GPUs or TPUs, you can do that with TensorFlow. You are not limited to the CPUs itself.

So it offers a lot of the advantages, the good things that you want for a math library, from a numerical computation library. Then on top we've layered really nice APIs and so on that you can directly interact with or build other kinds of things on top, and that's what you see with all these different libraries and probability or linear algebra and so on.

[0:24:58.5] JM: It really expands my — What I think about TensorFlow, the potential of it, because initially I saw TensorFlow and I think, "Oh! This is a machine learning library that's like Torch or something like that," but thinking more about the name, TensorFlow, all that really means is a multidimensional matrix of numbers that is positioned through a dataflow graph. If you just think about numbers flowing through a dataflow graph, well, what that symbolizes from a computer scientists, from a programmer's perspective, is a much broader array of tools. I haven't thought deeply about where that could potentially go. Do you have an idea for any kinds of applications that somebody could build with this kind of framework that maybe we can't even — Somebody wouldn't really be able to build today or people are not building with TensorFlow. What is your grandest ambition for the kinds of applications that people might be building with TensorFlow in the future?

[0:26:10.1] RM: Some of the AIs that we are already to see some interest is you talked about scientific applications at large scales. A lot of these are on scale super computers. Now, we are seeing some of those are simulations. So let's say weather patterns you're simulating what it might be in a day or two. Now, often what people want to do, and it's interesting where they want to mix some of these simulations with the machine learning side so they won't apply deep learning models to some of these predictions as well.

Where we started seeing people doing is already applying deep learning and scaling it and running TensorFlow on these huge super computers to do these machine learning things. I think what we'll see more of is combining that with the simulations as well, and so you have one environment, one computer platform where you can do all these kinds of things. I think at the large scale, I think that's really, really exciting to me.

On the other side, you were just talking about devices, there are lots you want to do on the really remote at the edge, because more and more, you talked about these industrial places

where you have these tiny devices and so on, I think all of them are collecting data, all of them have sensors today. The reason to have them is you can make smart decisions about what's happening.

Now, it's not really feasible to ship all that data back to some cloud and process it. Often, it'd be very valuable to run these things on those devices, and prediction is one part of it, but now that you have a platform that allows you to do a bunch of things, maybe use the existing [inaudible 0:27:48.9] we have, maybe you build your own things custom for what you have and really do all of those in one single platform. I think that's going to be really exciting.

[0:27:57.3] JM: Not just the model of doing computation at the edge, but there's also the middleware device potential where if you have cars driving around that have computers in them or you have drones flying around that have computers in them or you have people's smartphones where maybe they have spare compute cycles and if you had people walking around with their smartphones that had spare compute cycles and you've got — I don't know, a machine learning streetlight that is needing to do some kind of computation, it can maybe offload some of that computation on to nearby cellphones. The model for TensorFlow of being able to distribute computation among different nodes seems like it's well positioned to have that kind of computation distribution potential.

[0:28:50.0] RM: That's right. Those are some really interesting ideas. Yes, having a platform that lets you do things like that and combining this computation and distribution potential to access different kind of computation will definitely enable new interesting ideas and combinations of those.

[0:29:06.2] JM: Yeah. It's one of those things where you can see that future evolving, and I can just imagine, it must be very hard to design, because obviously you don't want to prematurely optimize for that kind of future, but at the same time you do want to leave design space open for that kind of future. Is that something you even think about or do you think it's something where you've designed software for long enough that it becomes intuitive, the kinds of abstractions that you want to build where you don't paint yourself into a corner or unable to take advantage of that kind of future.

[0:29:41.7] RM: Yeah. I think it's a bit of both. Yes, the experience that we or many folks on the team have had designing systems like this definitely helps. We have a lot of really smart engineers who've done — Build amazing systems here before. That definitely helps in designing basic APIs that are more likely to succeed. That said, we don't always get things right.

One of the important things that is in the world of today with changing things and so on and especially with something like TensorFlow is to keep in mind, keep an eye out for what do people want to do with this and learn from that. You don't want to just get stuck that, "Okay. This is our design. This is where we were and we're not going to change it." A good example that I see there is how we switch from just starting with pure graphs to now executing code directly as well. I think adding that new thing was hard. We had to rethink certain things, but now that we've done it and over the last year I think it's really improved the platform a lot and it's more accessible to many more users. You don't want to get stuck with, "Okay. Here's just where it is." Yes, you may have really great designs. Often, we do, but not always. We're still learning, and I think we'll continue to improve.

[0:31:04.9] JM: Another recent development was TensorFlow Hub, and this allows for reusable parts of machine learning models. This sounds very appealing. What is an example of a reusable module?

[0:31:19.6] RM: I can think of a couple of examples. So on the tech side, often you have this idea of embeddings, which are basically representations, vector representations of things like words or sentences, and these are interesting because you might train these ones on, say, some text data that you have and often you don't have to have label data. This is just regular text that you've been trained on.

What they might do often for word embeddings is the way these embeddings work is those embeddings are really — Think of those as points that represent those words in a really high dimensional space where words that mean similar things are closer to one another and words that mean different things are farther away.

These are very useful for lots of different text kind of models for natural language processing models. It turns out you can train them once on a reasonably good carpus and use them for lots

of different things. That's one thing that's I think very, very useful and that's something we offer there.

Another one that we've seen is, often, if you're doing, say, some kind of image classification or something, on a very custom task, you have your own dataset. You might have a really small dataset, say, 100, 200 images across a few different classes. Now, clearly, you can train a really large model just on that small dataset. However, it turns out that we and many others have trained these really good models on super large datas. It's like the image net database, which is about a million images that's available externally.

We've trained models on that dataset and it takes a lot of computing power, like the example that I was talking about recently in terms of low hub. Our best in class model took about, I think, something like 60,000 GPU hours to get that kind of accuracy. Clearly, not everybody can train that model. But what you can do is take that model and really take the features from that model and use that for your task.

We make the core model available and that's accessible to you with the weights from TensorFlow Hub. So while training your custom model on your task, you can just take that and just execute that to get the features and it helps you a lot with your task. You don't really have to go through that huge computational burden, and it's really exciting that you can get started pretty quickly.

[0:33:46.8] JM: When someone posts a model on TensorFlow Hub, do they also have to post the dataset that trained the module that — For example, if I train a word to vec model, if somebody else wants to use that word to vec model, then they could be concerned about the bias associated with how that model was trained if they don't have access to the dataset. Do they users who post modules also have to post the datasets associated with them?

[0:34:20.4] RM: That's not required. Some of these datasets that we use or others use are public datasets that are available and you might find information about those. However, we don't want it a requirement that you need to have a dataset. That said, for the specific issue that you mention where you might have biases, etc., we also have a number of tools to help you understand biases, to analyze the models themselves. For example, there was a specific tool

called TensorFlow model analysis that we just released as part of our overall TensorFlow extended suite of things to provide you the bigger machine learning pipeline itself.

There are a number of other tools as well that we have that help you just understand those models for different kinds of biases and we definitely hope to help, continue to work on that as well.

[SPONSOR MESSAGE]

[0:35:20.6] JM: Users have come to expect real-time. They crave alerts that their payment is received. They crave little cars zooming around on the map. They crave locking their doors at home when they're not at home. There's no need to reinvent the wheel when it comes to making your app real-time. PubNub makes it simple, enabling you to build immersive and interactive experiences on the web, on mobile phones, embedded in the hardware and any other device connected to the internet.

With powerful APIs and a robust global infrastructure, you can stream geo-location data, you can send chat messages, you can turn on sprinklers, or you can rock your baby's crib when they start crying. PubNnub literally powers IoT cribs.

70 SDKs for web, mobile, IoT, and more means that you can start streaming data in real-time without a ton of compatibility headaches, and no need to build your own SDKs from scratch. Lastly, PubNub includes a ton of other real-time features beyond real-time messaging, like presence for online or offline detection, and Access manager to thwart trolls and hackers.

Go to pubnub.com/sedaily to get started. They offer a generous sandbox tier that’s free forever until your app takes off, that is. Pubnub.com/sedaily. That's pubnub.com/sedaily. Thank you, PubNub for being a sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

[0:37:03.5] JM: I want to zoom out to some of the other associated discussion that you mentioned at the beginning of the conversation and some things that were touched on at the

TensorFlow Dev Summit. AutoML is a project that Google is working on to allow people to train custom machine learning models without thinking much about how that training is actually proceeding. Has the — I don't know how much — Do you spend any time on AutoML or are you mostly focused on TensorFlow?

[0:37:36.3] RM: So my focus is primarily TensorFlow. I know a lot of that work. I mean, a lot of those folks sit here as well. I know a lot of those researchers. Yeah, my focus is definitely more on TensorFlow.

[0:37:48.1] JM: Okay. Have there been any learnings from the AutoML team that have impacted how you're designing TensorFlow?

[0:37:56.2] RM: A few things I would say. One, this is another area where, to me, AutoML works is it's trading off more human expertise, more computation. So having TensorFlow really be optimized for these kind of things is really, really important. I mentioned 60,000 GPU hours that was used for training one of the AutoML models, and when you're getting at that scale, if the tool is not optimized, if it doesn't perform, it doesn't take advantage of the machines and the devices as much as it can, then you're really wasting a lot of resources. That's definitely been one.

Another is more — It's been interesting to see how AutoML can optimize for different kinds of environments, one of the things that they were working on was optimizing models for deployment on devices. Even though they are being trained and they're being optimized to get really high accuracy, they're also being optimized for deploying on devices so they into account the kind of computation you can do there or how it's going to take and there's a budget maybe that you have and you want to balance for that. So those were clearly interesting and sort of leveraged all the pieces that we have in TensorFlow as well.

[0:39:14.0] JM: I've heard a lot of emphasis recently on transfer learning. What is transfer learning and why is it so important?

[0:39:22.0] RM: So this is sort of the example that I was talking about for images. Let's say you have an example where you just have a couple of hundred images, say, across 4, 5 classes. Now, if you want to train a really good model for those, it's going to be hard for just these few

images. You're not going to get a very accurate model, if you start from scratch, if you have no idea even if you use the best description of the model that people have had from research.

What people often do for cases like this is exactly what you would get from model hub, TensorFlow Hub, which is where you have the entire model pre-trained on a different dataset, and often it's image net which has a million images or more, and then you've trained it on that it turns out because images often of natural kind of things are similar in natural. Most of that model you can really use for a different kind of application. Even for, say, you had a dataset for identifying five kinds of birds, you can reuse most of that model. You take that as a module, you plug that in and just train a small classified on path. This is called transfer learning basically where you're taking the ideas and learning form one dataset and transferring it to another dataset, which may not necessarily be large enough for this.

[0:40:41.0] JM: Is there any necessary work to make transfer learning a first class citizen in TensorFlow? Do you try to make transfer learning easier to implement?

[0:40:53.0] RM: Yes. We allow you to easily control what you're training, what you're not, easily port models or take parts of modules like these modules with the TensorFlow Hub where you can say, "Okay. I have already this pre-trained model from this other task. I want to take this. I want to just change the stop here, change the classified." All that is really easy. We have tutorials to do that as well. We have examples and so on.

[0:41:17.9] JM: Another development that you mentioned earlier was GANS, the generative adversarial networks. What are the use cases where GANS are useful?

[0:41:27.7] RM: So GANS are interesting for a number of areas. Often, the basic idea is there's one network that's trying to create something, that's trying to generate something, and there's another network that's trying to separate that or discriminate that from whether that's real or not, from real data or not, right? This forces the first one to keep getting better and the second one to keep pushing it harder and harder and so on, and that's what allows it to learn. That's the very basic there.

These can be useful generating different kind of things. Often where we are starting to see them is for generating images where people are generating — There was this interesting work from folks at NVIDIA, I believe, where they train it on lots of celebrity pictures and then let this model generate different kinds of pictures. If you look at those pictures, they look very clear. They look like real people, but they are really just figments of imagination of this model, and they're starting to look pretty realistic. This is a fun example.

I've seen real companies use that for designing new clothes, for coming up with new kinds of patents for clothes, for new kinds of styles. So we're starting to see some real applications of that in the industry as well.

[0:42:50.3] JM: Yeah, I don't know if you've seen, but there are these generated fashion models on Instagram where they're not real people, but they have thousands and thousands of followers and people have created these fake human beings and then they just put some influencer marketing behind it. Like they'll make a fake person and then show the person wearing Nike clothing and sell that to Nike, and they've got 60,000 Instagram followers, and it's just like not a real person, but people are purchasing advertising against this non-real human being. It's kind of a — I don't know. One of these things where it's like, "Wow!" It really feels like you're living in the future when you have 60,000 Instagram followers to an artificial human being on a social network.

[0:43:39.5] RM: Yeah. Those can get tricky as well.

[0:43:43.0] JM: Definitely. How has the TensorFlow model debugging experience advance? Because you have people who train models and then they want to improve those models and maybe they have some issue with the model that keeps coming up, like the model keeps tagging. Whenever it sees a book tags, that book is a computer, because it's also rectangular shaped. You don't exactly know why it's tagging the books as computers. How does the debugging process advanced?

[0:44:14.1] RM: There are a number of things that we've done in this direction, and like we talk about those one by one. One is the debugging from the — Just as you're building a model, you want to understand maybe the model is not working and so on and you want to understand that

better. That's an area we think the whole idea of eager execution, where you're basically writing just Python code. What that enables you to do is to debug with Python tools.

Any tool, any library you have in Python or just your basic Python debugger, you can do whatever you want with it. You can set breakpoints. You can print out stuff. You can visualize stuff using standard Python libraries and so on. That's sort of one direction.

Two, for a similar thing, sometimes you do want the graphs because they can be optimized much better. They can be deployed across the distributed cluster and so on really, really well. Let's say you have that. For that, we have something called a TensorFlow debugger, which now we recently announced this visual integration with TensorBoard as well so you can actually — It's like your debugging environment that you might get an idea and it looks really cool and it's great to debug. Those are from early programming debugging style things.

Some of the other things you mentioned were around, okay, understanding a model or understanding a specific mistake, let's say, that the module made. I think from a research perspective, there's been a lot of progress interpreting these models. So for a specific example, if it makes a decision or a classification, there are these tools that can help you really try and understand what parts of the inputs did they come from.

For example, for an image, and you mentioned a book, it's classified as something else. You can see what part of that original image made the classified, made that model think that that was the case. Then you can use some of this information. You can also look at different parts of your model itself and you can use all that information to learn and think about ideas to improve that as well.

[0:46:18.9] JM: I was at the TensorFlow Dev Summit in 2017. I was unable to make it this year. At the 2017 event, there were presentations of applications for detecting skin cancer and diabetic retinopathy with machine learning models, but these applications were only being used in laboratory environments, and I saw these and I got excited because I was thinking, "Oh! Over the next year, we're going to be using our smartphone to scan something on our skin and detect if it's a freckle or if it's high risk and if we need to go into the doctor's office," but it's been a year

and I haven't gotten a smartphone application that does that. My sense is that that's because there is a barrier to getting these tools approved.

What are the barriers to getting these kinds of low hanging fruit medical applications of machine learning, these self-served consumer applications. Why aren't we seeing more of these yet?

[0:47:24.1] RM: I think from a medicine perspective, there are a number of things, right? There are the basic health kind of tools and whether it's your Fitbit app or your Google Health app and so on. The things that runs on your phone and tracks your progress, whether you're running enough or not and things like that. Those can definitely have machine learning today and do the basics for you today.

But beyond that, if you really want to build an application that really complements your doctor or helps your doctor make decisions, you need to be much more rigorous and thorough about that. There is a good reason why medicine today — If you want to introduce a new medicine or a new technique, it needs to go through a process to get approved and the reasons are you don't want to get something out there which will make wrong decisions or which will lead doctors to make wrong decisions. You really want to be very, very careful in that.

So that process takes a while. There are a number of applications like the one you mentioned for skin cancer and others that are starting to make their way through that process, and overtime I think we'll just start, we'll see more of these getting deployed across the board, not just in the US but outside the US as well.

[0:48:40.1] JM: Something I saw from Jeff Dean's talk at the TensorFlow Dev Summit this year that was unbelievable to me was that the same images that were used to predict diabetic retinopathy, just these images of people's eyes, they could also be used to predict age and gender. You have this labeled dataset of eyes, just images of close up eyes and you think, "Okay. Great. We've got a labeled dataset of eyes." How useful can that be? Certainly, you can detect diabetic retinopathy with it, but the fact that you can predict age and gender is remarkable, and I think there's also — I think there's been work around predicting risk of heart disease just from looking at eyes. Are there other instances you've seen where these datasets have produced — I guess there's probably high fidelity, unstructured data that has a lot of detail

that is hard to interpret to humans. That's probably the class of dataset we're talking about, but have you seen any other unexpected predictions that have blown you away?

[0:49:54.7] RM: This one definitely beats everything else I've seen. I think — You mentioned unstructured dataset, and that's definitely true. In this case, most people have no idea you could do that. As you saw in Jeff's talk, the folks who are doing this were also extremely surprised. This was more just a test and just a process that we were trying to go through to help somebody learn about how this whole thing works.

For other areas, I can see more of this happening as you find these models are incredibly good at finding fairly complex patterns of things. So as we provide different kinds of data, I expect to see more of these. That said, no, I haven't seen anything as amazing as this.

[0:50:39.3] JM: I wonder what the other datasets that will be worth exploring for this kind of thing. I can imagine pictures of planets, and maybe if you have some idea of the chemical composition of the atmosphere in those planets, maybe you can figure out more about the physics associated with those chemicals that are in those atmospheres, or something like that. I'm trying to think of something that's analogous to these close up images of eyeballs. Nothing's coming to mind, but — I don't know. Maybe you can let me know if you think of anything.

[0:51:12.8] RM: Yeah. One of the things that the reason we could actually show that it seems to be related and you could predict is that the dataset had the values from the people as well. The dataset had the gender value and so on. You could take that out for making predictions or for learning and so on. You could learn from that and make the prediction.

For a lot of the things like the example you said, the hard thing would be you need a dataset which has those values first where we have strong confidence because of how that thinks and so on. That said, yeah, it will be interesting. I honestly can't think of any crazy things like this. Yeah, I'm sure we'll see some.

[0:51:51.3] JM: Well, this is what makes me really excited and optimistic about the Google project baseline. This is something I'd really like to do a show about. I don't know if there's

anybody out there that is in the Google baseline project, but I think this is like where 10,000 people are getting tracked for like 10 years at close fidelity, pretty exciting.

What are some other simple but powerful applications that we're going to eventually have by just combining machine learning and the smartphone?

[0:52:19.9] RM: The most simple things that we see, of course, are you have the phone with you. It has things like camera and audio and you use them for interesting things. I'll give an example of something that's deployed today. So if you buy a Pixel 2, then the camera on that device when you take a picture of a person and so on in portrait mode, it basically separates the foreground from the background for you and it basically show — It has this effect, it's called a bokeh effect which basically fuzzes the background essentially and makes the person pop out really well. This was all done using machine learning on the phone, using existing stuff on the device.

I think the interesting thing on phones and other devices, what's the information that goes into there? What's the information that this device is getting? In the case of phone, it has all kinds of sensors. The most commons one we think off are the audio, the speaker, the vision sensor, the camera, and I think the combination of those is going to be very interesting in terms of what you do with that on the device itself.

[0:53:31.6] JM: All right. Last year the focus was on usability for the TensorFlow team. What's the focus this year?

[0:53:38.1] RM: I think it doesn't let up. I would say a couple of things. I think performance is where we started is a big thing. Performance and usability continued to be important for us. I think last year we had said we see TensorFlow as fast. It's flexible. We want to be able to do amazing things and production ready, so you can deploy these things into real-world applications as well.

So I think from the perspective of where we are, I think it's important for us to make sure that people can really use this in real-world things. So we highlighted a number of things that were made with TensorFlow and people are doing some interesting things. We would love to see

more of that. We would like to see more integrations with all kinds of real-world applications that people use.

I think we'll do lots of different things to really bring this to more people, and making it easy is just one. Another example is the JavaScript thing, which I think enables a new developer community of a different developer community necessarily potentially from a Python community which is excited about this and can do lots of interesting things.

[0:54:50.5] JM: yeah, and we're going to do a show in the near future about TensorFlow.js. We'll be doing a much deeper dive there. Rajat, I want to thank you for coming on the show. It's been great talking to you once again.

[0:54:59.4] RM: Thanks, Jeff. It's always a pleasure being here.

[END OF INTERVIEW]

[0:55:05.0] JM: GoCD is a continuous delivery tool created by ThoughtWorks. It's open source and free to use, and GoCD has all the features you need for continuous delivery. Model your deployment pipelines without installing any plug-ins. Use the value stream map to visualize your end-to-end workflow, and if you use Kubernetes, GoCD is a natural fit to add continuous delivery to your project.

With GoCD running on Kubernetes, you define your build workflow and let GoCD provision and scale your infrastructure on-the-fly. GoCD agents use Kubernetes to scale as needed. Check out gocd.org/sedaily and learn about how you can get started. GoCD was built with the learnings of the ThoughtWorks engineering team who have talked about building the product in previous episodes of Software Engineering Daily, and it's great to see the continued progress on GoCD with the new Kubernetes integrations. You can check it out for yourself at gocd.org/sedaily.

Thank you so much to ThoughtWorks for being a longtime sponsor of Software Engineering Daily. We are proud to have ThoughtWorks and GoCD as sponsors of the show.

[END]

© 2018 Software Engineering Daily