EPISODE 308

[INTRODUCTION]

[0:00:00.3] JM: Machine learning frameworks like Torch and Tensorflow have made the job of a machine learning engineer much easier. Machine learning is still hard. Debugging a machine learning model is a slow, messy process. A bug in a machine learning model does not always mean a complete failure. Your model can continue to deliver usable results even in the presence of a mistaken implementation.

Perhaps you made a mistake when cleaning your data, leading to an incorrectly trained model. It's a general rule in computer science that partial failures are harder to fix than complete failures. In this episode, Zayd Enam describes the different dimensions on which a machine learning model can develop an error. Zayd is a machine learning researcher at the Stanford A.I Lab. So I also asked him about A.I. risk, job displacement, and academia versus industry.

[SPONSOR MESSAGE]

[0:01:02.3] JM: Simplify continuous delivery with GoCD, the on-premise, open source, continuous delivery tool by ThoughtWorks. With GoCD, you can easily model complex deployment workflows using pipelines, and you can visualize them end to end with its value stream map. You get complete visibility into and control of your company's deployments.

At gocd.io/sedaily, you can find out how to bring continuous delivery to your teams. Say goodbye to deployment panic and hello to consistent, predictable deliveries. Visit gocd.io/sedaily to learn more about GoCD. Commercial support and enterprise add-ons, including disaster recovery, are available.

Thank you to GoCD and thank you to ThoughtWorks. I'm a huge fan of ThoughtWorks and their products including GoCD, and we're fans of continuous delivery. Check out gocd.io.sedaily.

[INTERVIEW]

[0:02:14.4] JM: Zayd Enam is a machine learning researcher at the Stanford A.I Lab. Zayd, welcome to Software Engineering Daily.

[0:02:21.0] ZE: Thanks for having me, Jeff.

[0:02:22.4] JM: Today we're talking about why machine learning is hard and in what ways it is hard.

[0:02:26.9] ZE: Yeah.

[0:02:27.4] JM: Let's start more optimistically. What are the fundamental trends that have led us into such massive growth in terms of the adoption and the viability of machine learning today?

[0:02:39.9] ZE: Increasingly, we're seeing the value of machine learning algorithms as we've been able to build computers and processors that can process large amounts of data. We're seeing increased value in this idea of collecting, or having the computer figure out what to do from large sets of data. Identifying patterns in large sets of data as opposed to codifying individual roles in order to be able to do a task.

[0:03:06.5] JM: There's also, obviously, lots of new frameworks, and cloud services that make this a lot easier, and we can get into that stuff. To get to some of your editorial that you wrote about why machine learning is hard — it's a very interesting article that I'll put in the show notes — you write about the difference between software development and machine learning. Could you outline some of the points that you make that lead you to that conclusion?

[0:03:33.2] ZE: I think, fundamentally, the distinction between software development and machine learning is that you have increased dimensions of complexity that you deal with. The way I think of a complexity is complexity doesn't scale linearly with every dimension. It scales in the size of geometric volume.

With software development and with a lot of different software development, you have increased complexity because of different dimensions. If you looking at just basic software development compare it to basic machine learning, in software development, what I sort of see as you're

writing a simple program, a standard simple Python program, maybe a Script, maybe you're writing something simple on your computer, you associate an algorithmic complexity that you're dealing with than an implementation complexity.

The algorithmic complexity you're dealing with is like the actual algorithm that you're writing for your data structure, the actual way you're processing your data. The implementation complexity is the actual implementation of that algorithm in your code. Each one of these dimensions is prone to a bug. They're prone to sort of you not thinking through correctly what the right way to do this, the implementation thing. Maybe you're not using the right API calls, or you're not using the right data structures for whatever library you're using. There are all kinds of little things on the algorithm side, you're not thinking through the algorithm. I have an example of recursive algorithm that I'm just not thinking through the base code correctly.

Those are a lot of different ways that you can get them wrong. With machine learning, even the most basic machine learning, you would still have those two issues, where you're still having issues which are algorithm and you're having issues which are implementation. All of a sudden, you have two new dimensions to deal with. These two new dimensions are the complexity of your model. The complexity of your model is, basically, you're trying to model something about the world. You're trying to model whether you think it's going to rain tomorrow or not, and that's based on a lot of different features and the prediction can be based on a lot of different things.

You also have complexity in your data, and your data complexity is this complexity of often times a data that we get to be able to train our machine learning algorithms, is really messy. It's not in a clean format. It's not in a format that we can easily sort of plug into a model and get it to train and work well. You have to do a lot of preprocessing of this data and you have to do a lot of munging and these kinds of things.

Even for the most basic things, when you add these two dimensions of complexity, it becomes sort of a lot more of sort of volume in your head. The way I think about this is it's like development volume, or complexity volume. As a developer, you can only sort of fit a certain amount of volume in your head. I can, while I'm writing code, or while I'm sort of tweaking an algorithm, or start trying out a new research experiment, I can only deal with a certain fixed amount of volume.

In order for me to be able to deal with that kind of volume, I'll try to completely not think about the implementation and just like take that as a given and then just think about how I'm going to sort of fix the model, or I'll try to not think about the data and then fix one particular dimension and then try to think about the other three dimensions. It becomes a challenge of managing this volume in your head.

[0:06:41.5] JM: I think you're getting at a number of things. You're getting at the problem of the data munging process where you've got to clean your data and you can make a mistake in the data cleaning process and that can lead to erroneous inputs. You also have this partial failure scenario where you can have a model that does a pretty good job, but it's not optimal. Maybe you're at a local minima and you could have all these problems that are almost subjective. It's like you can have a model that partially works.

Machine learning, it's a hard problem because getting algorithms and models to work well is not like just standing up a web app and having some transactional service available. You have to apply creativity, experimentation, and tenacity. Machine learning is it's a lot less straightforward than other types of software development.

You describe an intuition that a machine learning program has to develop. How does that intuition manifest, and why is it a prerequisite for being a machine learning developer?

[0:07:51.0] ZE: You hit the nail on the head and it's exactly, I think, for the reason that you described, and the reason is that machine learning, it's hard to know when you — You don't have quite the same binary measure of success. It's not like you've immediately succeeded. There are a lot of little signals that you need to sort of get back from your algorithm and in the process of development. These little signals guide you in the direction of what you should be doing.

You don't know quite a priority what the right model is for your data. You don't know quite a priority what the right processing is for your data. You can't sit back and sort of write down a set of rules that this is exactly what I need to do. When you're training these systems, especially with these really, as you get to larger scale deep learning systems, there's certain rules of

thumb, but there's always something that doesn't work. Something that doesn't work is often because there's a lot of different reasons why something doesn't work, and these systems, they don't exactly tell you why it's not working.

When you have a lot of different dimensions along where something doesn't work, you need to be able to develop this intuition for, "Okay. I'm seeing this particular behavior that this model is telling me, or this algorithm is telling me. I need to tie it back in to what I think, of all the three or four dimensions I'm thinking about right now, where is this problem?"

[0:09:03.3] JM: Yeah. "The difficulty is that machine learning is a fundamentally hard debugging problem." This is a quote from your blog post, Why Machine Learning is Hard. What makes machine learning code so much harder to debug the normal code? What's so peculiar about it?

[0:09:22.8] ZE: In machine learning, or with machine learning code, the complexity often is when you're debugging this code, you're not just debugging the deterministic algorithm. You're not debugging, as I earlier talked about, let's say I'm implementing a recursive algorithm. I have the actual recursive, the algorithmic part of the recursive algorithm, and the implementation part. I know in my head this is exactly what the algorithm should do.

With machine learning code, you have a particular algorithm that trains your model. That might be a gradient descent based method, that might be a way of learning a decision tree, some kind of algorithm. You also have implementation, which is writing down the actual code for that particular algorithm.

You still have that same level of complexity in those two aspects. Now, on top of that, there are these other dimensions of complexity, which is the model and data that come into play with both the algorithm and with that implementation. When you combine all those together, it becomes much more complicated to try to debug something when things aren't going right and things aren't working perfectly.

[0:10:30.5] JM: As you write, there is a number of different axes that a bug can manifest within in this machine learning pipeline. Describe those other axes is more detail, because in normal programming, if I'm just building a Ruby on Rails web app, it's often a little more binary. You've

got a bug and that bug manifests an external error and it's very clear cut that's an error. These axes that you described in machine learning add a much more gradient scenario for how errors can manifest.

[0:11:08.0] ZE: Yeah. In a sense with the two additional dimensions with sort of issues with the model, issues at the actual data, it's often combinations between an implementation issue and like a data issue that will sometimes give you huge problems. If there's like some kind of particular characteristic about your data, or your model doesn't like data in a certain format, because, for example, with earlier forms of unsupervised deep learning, it was very common to preprocess that data not just doing simple mean subtraction, these kinds of things, but people would do whitening and all these other transformation on the data in order to get it to get these actual early deep learning models to work well.

There are all kinds of little bugs that could show up in that process. You're not quite whitening your data correctly. Your model doesn't like this form of whitening. It would prefer this thing — There are like some other issue with your data where — For example, with images, I was dealing with a dataset years ago where I was trying particular models on these big dataset natural images. There were issues with the actual camera processing, but the actual camera that was processing these natural images — The sensor that captured these images, it resulted in a format that when you whiten the data, it became unstable for this deep learning algorithm to train on — For this unsupervised deep learning algorithm to train on, and so you run into these kinds of little issues where your implementation, your model, your algorithm, and your data need to come together, just align perfectly in order for something to work perfectly.

[0:12:39.6] JM: Yeah. Although all hope is not lost, because as you say, luckily for machine learning, we also have more signals to figure out what went wrong. What so you mean by this? How does that signaling from the machine learning output lead to us improving our models?

[0:13:05.5] ZE: With machine learning algorithms, one thing that you have, the data and the model give you additional — They give you additional signals to try to figure out where something is wrong, or where you can improve things. It's not like they sort of throw you into these extra dimensions of complexity without giving you extra signals and extra ways to cope with that complexity.

One of the nice things about machine learning is that you can have the model tell you — At the end of the day, it tries to give you some kind of answer. Whether that's like a classification label, or rational label, it's trying to give you some kind of answer. You can just basically give it a lot of different examples and just have it see what is the output? What is the answer that you'd give for these examples?

That is a really powerful way of trying to understand where things are going wrong. It's a basic way, but it's also very powerful way, where you just look at a lot of different inputs and you see where is the system failing? Where are things going wrong? That's a great place to start.

You can then look at statistics of these kinds of — For example, what are your statistics on your training and test site in terms of are you overfeeding the model? These kinds of individual statistics are really powerful, but then, additionally, these are all at the end of the train model, while the model is training, it gives you upward signals in terms of its loss function, how fast the loss function is decreasing as it plateaued. It gives you this individual — These numbers that you have to sort of keep track of in order to be able to determine where the problem is, but it has all these different outputs.

[0:14:39.9] JM: What's interesting about this conversation is that when you're saying machine learning is hard, I think machine learning is one of the these really strong breakthrough innovations where it's like the iOS platform where people all of a sudden had this incredible platform to start making mobile apps and it unlocked this vast amount of innovation from developers everywhere. You could same thing about cloud computing, you could say the same thing about Ruby on Rails.

Up until I read your article, my impression was like, "This is what — Machine learning is the same thing. Okay, we've got Theano, we've got Torch. These things make it super easy to just get started with machine learning, and you get some homeruns and then — " Are you saying that's true, you do get homeruns, but you're going to hit some diminishing returns quite quickly, because you're going to start to hit these debugging problems that are really going to slow down your progress once you get started, or are you saying that even just to get started, even just to

standup a model and have it be producing reliable positive results, you're going to need to do some hard work?

[0:15:49.5] ZE: We've had tremendous success with frameworks like Theno, Torch, and TensorFlow. The reason is because the development community has sort of realized that the complexity of machine learning, there's multiple dimensions of complexity you have to deal with that when you take away implementation complexity — What is the advantage of TensorFlow, Theano, and Torch? Is that you no longer have to worry about — A few years ago, every time you sort of coded up on your network, you'd have to worry about the most computations that you were doing on a GPU. You'd be implementing the actual CUDA calls to do a matrix operation on the GPU and you'd have to worry about the complexity, "Oh man! Did I implement that correctly? Did I have the right implementation with this operation? Am I copying memory from my host to GPU correctly? Is the memory access being done correctly?" You'd be dealing with all those valuable issues alongside all the other issues that you have to deal with from machine learning.

What these amazing frameworks have done is that they've struck away all these complexities in a very sort of very robust strong framework. You no longer, in a sense, for implementation, you have — A lot of times people are doing something new, so they have to worry about the implementation of the new aspect that they're doing. For a lot of the most basic operations, that part of the volume of complexity has been sort of taken away, because you're not worrying about that in terms of — or you're not thinking about that sort of volume of complexity while you're implementing your model. You're free to sort of experiment in other dimensions, along the model dimension, or the data dimension, or exploring new algorithm dimensions.

That's a really powerful advancement by these frameworks. That's what's made it possible for us to make so much rapid progress. I do think that if people were still sort of writing their CUDA kernels for training neural networks, the field would be many, many years behind what it is right now.

[SPONSOR BREAK]

[0:17:44.4] JM: When you are continuously deploying software, you need to know how your code changes affect user traffic around the world. Apica System helps companies with their end-user experience, focusing on availability and performance. Test, monitor, and optimize your applications with Apica System. With Apica Zebra Tester, Apica Load Test, and Apica Synthetic, you can ensure that your apps and APIs work for all your users at any time around the world.

Apica Zebra Tester provides local load testing for individuals, small teams, and enterprise DevOps teams to get started quickly and scale load testing as your needs evolve. Apica Laod Test ensures that your app can serve traffic even under high load. Apica Synthetic sends traffic to your website and your API endpoints from more than 80 different countries, ensuring wide coverage.

Right now, you can go to softwareengineeringdaily.com/apica for a webinar about the real ROI of API testing. You can also find past webinars, just how to optimize websites for fast load time. Go to softwareengineeringdaily.com/apica to find the latest webinars on load testing and lots of other topics, and check out Apica System for testing, monitoring, and optimization.

Thanks again to Apica for being a sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

[0:19:14.4] JM: Now, is there more low-hanging fruit in terms of making the procedure of a machine learning developer more streamlined and more error-free, or machine learning is heard? Is that the epitome of some wall that we've hit on the free lunches, and now it's just going to be — There is going to be a lot of in the trenches work?

[0:19:43.2] ZE: Absolutely. I think there's a lot of work that can still be done to decrease the complexity of sort of building these kinds of systems. What I think here is there are a few fundamental tensions that happen over here. One of the fundamental tensions is that there's always this sort of push for higher performance. One of the biggest ways to get higher performance is to increase the size. Right now, what we've been seeing for a very long time is that the most simplest way and the most reliable way to get better performance on an algorithm is by increasing the size of the dataset. That increases your sort of requirements for

computation. That increases the complexity of the data, and increases the requirements for computation.

Because of this increased requirements, increased push on computation, you sort of get in realms where maybe you can't do your processing on a single node anymore, you need multiple nodes. If we take an example from an algorithm from the 90's, I could just brute force all the parameters for the algorithm right now probably on my laptop and not have to worry, just have a simple auto-script that basically checks everything, does every single possible combination and gives me a reasonable result in a very short amount of time.

But it's usually once you to start pushing on the sort of limits of performance, you have to increasingly move towards more and more, higher and higher compute. Because of that, it naturally becomes more difficult to manage those kind of systems, because now it's no longer easy to run that kind of algorithm multiple times. Say, you wanted to do a hyperparameter search for a new fancy algorithm, if that algorithm by itself is so large and requires so many flops for one run, it's hard to be able to tune those hyperparameters without some kind of better way than just like brute forcing everything.

[0:21:34.6] JM: Can you give me an example of some different bugs that you have encountered yourself in this definition of a bug that you classify in machine learning where it's like you have this broad, gradient, different types of bugs that are more subjective, perhaps. Give me, maybe, an example of one that you've encountered and how you walked through it.

[0:21:57.8] ZE: Sure. I think early on, one of the interesting bugs I spent a little bit of time on was I was — Usually, when you are looking at loss functions, when you're training your algorithm, you expect sort of this really quick decay and it sort of shallows out overtime. The algorithm, it gets really good pretty quickly, and then overtime it's getting better, but it's getting better slowly.

It was just interesting for me to see this loss function where you'd expect getting better and better overtime, it was periodic. In the sense that it would go up and down in a very patterned way. In retrospect, it seems so obvious now there was obviously some kind of issue in the way I was ordering the data when I was training this model, or this particular model with a iterative

algorithms. So it would take a small batch of data, crosses that data, learn from those examples, and then take another batch.

In retrospect, it seems pretty obvious there was some kind of periodicity in the way I was loading up data into these models. It was just like — I had never seen sort of a sine wave for a loss function ever before, and it just like struck me as like really odd. But I've seen this now a few times when I'm helping out other people debug their models, or figure out what's going on where there's an issue where you are not shuffling the data correctly, and results in these periodic loss functions.

That's an example of where ordering of the data really matters and you need to randomize ordering. Otherwise, it learns patterns from the actual ordering of the data as opposed to you don't want it to be learning those patterns. You want it to be sort of getting this data randomly.

[0:23:34.2] JM: What's the tooling around this? Is it more you are just writing stuff and then you see output in a graph somehow and then you say, "Okay, now it's back to the drawing board?"

[0:23:44.9] ZE: Well, that's what it predominantly was a few years ago. You were basically writing everything from all the code to do the operations in the GPUs, to the code, to load up the data, to the code, to plot these lost functions, and plot the statistics of your data, and plot error examples and these kinds of things, which made it such an unwieldy process.

Now, with tools like TensorFlor, and Theano, it becomes — With Theano, you have certain plugins that makes it much easier to process these kinds of errors. What it ends up being now with the modern day framework is that you're writing, basically, summary statistics to a particular web view or some kind of log and you can basically sort of look at this graphically in your browser and it will immediately give you a signal that something is off.

[0:24:35.7] JM: You're describing this delayed debugging cycle that you write about, which aggravates the debugging challenge that's already difficult. Why is that debugging cycle longer? I think by debugging, they mean, typically, somebody developing a web app, they just save their code, refresh the page, notice the next bug, solve it, rinse and repeat. That is not how things proceed in machine learning.

[0:25:04.4] ZE: Yeah, man. I'm so jealous of web developers. I got back into web development, maybe, a few months ago, and it's just so nice. You just write something and your page — Your browser is —

[0:25:17.1] JM: Like painting a picture.

[0:25:17.9] ZE: Yeah, you see it immediately, and you get this immediate visual feedback. You're sort of laying out stuff in CSS and you get this beautiful feedback, immediate visual feedback of what you're doing. If you're doing front-end stuff, you're doing back-end stuff. It's a very quick cycle. The page refreshes and you immediately see what the results of your work were.

I think that's a really powerful — This kind of cycle of the work you put into seeing the immediate result of that, that's a really powerful learning cycle, because, in a sense, it's like this problem of when you're trying to figure out what to do in a situation. If you get an immediate feedback on what exactly — Whether this was a good thing to do or not good thing to do, but that's really a fast way to learn about things, because then you can experiment with a lot of different things and sort of play around with a lot of different knobs and get a good feel for what works and what is the fundamental principle that will push progress forward?

The thing is — What I was talking about before is that when we're pushing for higher performance, it results us in pushing the limits of computation, we have these delays in hours to train these algorithms just because we're trying to hit the fundamental limits of what is the current state of the art for the forms of this algorithm? That process takes hours, if not days, if not weeks to train that algorithm to that level.

What the issue becomes is that you run an experiment to test out a particular hypothesis that you have and you have to sit and wait for that the results of that experiment to come in for days, or hours, or for a long period of time before finally get a learning signal yourself where you can feed that back in and say, "Okay, when I sort of change from this kind of gradient descent method to another gradient descent method, this is the behavior I see, and this is the

performance that we get." That delayed learning cycle sort of really slows down progress in the field. It slows down progress in the field and slows down progress in development.

If you were able to sort of shrink that delay to sort of a much smaller delay, I think we'd see a lot, lot faster progress. In a sense, machine learning as a field, especially since the big players, a lot of industries have industrial players that have come to this space, you'll notice that the field has been doing a lot of, in a sense, hyperparameter optimization with lots and lots of machine learning researchers. There's a particular architecture and there's a lot of people exploring different ways they can twit a knob and they add small little thing to the algorithm in order to improve performance on a benchmarking dataset.

In a sense, this is like an example of — It's the result of the fact that doing these kinds of computations takes such a long time. It was possible for a person to just sort of run all these experiments and get the results for them really quickly, then we'd not spend a year in this process for these multiple research groups doing hyparameter. In sense, what is essential, becomes hyperparameter optimization on an algorithm in order to comeback to this solution where every community decides, "Okay. This is the way that seems like a convolutional neural network should — These are the parameters for a convolutional neural network that seemed to work well in practice."

[0:28:25.7] JM: Are we going to get there at some point? Are we going to shorten those debugging cycles significantly?

[0:28:31.1] ZE: I think this comes back to exact same tension between computation and performance.

[0:28:34.8] JM: Same thing. Okay.

[0:28:35.8] ZE: Yeah.

[0:28:36.1] JM: Right.

[0:28:36.4] ZE: It's like whenever you try to push on performance, you always going to require more from your computation. We get to this level where we're talking, for example, if we wanted to build a human-level intelligence, you can't really shoot for a computer that has exactly the same number of operations as the brain, because even if I wanted to build a computer that had the same number of operations that the brain has in terms of number of neurons, number of action potentials, and synapsis between all the connections between the neurons in the brain, it would still be such a long process to iterate over all the possible ways you could build a brain.

That really, you need to be shooting for a few order of magnitude bigger than the brain, and you need those few orders of magnitude faster or bigger than the brain of computation in order to be able to do this like, in a sense, what becomes a hyperparameter search. Where you have this ability to quickly test out different experiments at the scale of the brain and basically run a lot of them in a reasonable amount of time. Then, you'll be in that process that you'll learn, "Okay. These are the models that work. There are the hyperparameters that work, and this is how we build a brain."

[0:29:44.0] JM: You did study neuroscience. How does that drive your interest in machine learning?

[0:29:52.0] ZE: I actually got fascinated early on by the brain, and it was predominantly the brain that I was super, super fascinated in. Sort of — I found this like motley groups of folks at my undergraduate where they're doing amazing work at, basically, intersection of neuroscience, machine learning, math, and sort of understanding the fundamental principles of computation of the brain, and that sort of what drew me into this interdisciplinary field of trying to understand the actual computational principles of the brain through the tools of mathematics and through the tools of machine learning and statistics.

Increasingly, I think machine learning, or progress in machine learning as a field, progress in machine learning hardware, and progress in machine learning algorithms is increasingly pushing us towards this direction where I don't think we're going to be able to exactly build a brain, but we are sort of unearthing fundamental principles about the brain that seem to be reflected as fundamental principles that are important for intelligence.

[0:30:53.5] JM: Right. I totally agree with you, where, basically, there's some set of satisfactory primitives that you need to build, what is effectively, a brain. What is effectively as something that as as powerful as the human brain. We can get to those — Whatever those axiomatic things are, we can get to something that is equivalent to them through just hacking on machine learning, basically the thing that we're doing right now. That's what's so exciting right now. What's so exciting about all these is like through, basically, brute force, we're just going to get there.

[0:31:30.6] ZE: Yeah, and it's going to be great.

[0:31:33.4] JM: It's going to be great. It's going to be very cool. Okay. This gets us into more interesting conversations, not that machine learning debugging is uninteresting. It gets us into more speculative conversations, perhaps I should say.

There's the neural lace idea that obviously we should be covering, this idea that, basically, in order to — well for whatever reason you want to make a neural lace. It's like the interface between the brain and our computers that Elon Musk suggests building in that Recode interview. He was suggesting it as a response to averting A.I risk, basically, because if you use a human brain as a springboard — If I understand his theory correctly, you use the human brain as a springboard so it sorts of prevents, or keeps our interests aligned with the AI, or something.

[0:32:24.6] ZE: Yeah. I've sort of seen these comments that I think people in the actual field always have a laugh about them. Whenever a comment like that comes up, it just feels like — To me, I think, about this particular thing, it's just like Elon is an expert in particular fields and he's done remarkably well and has a really deep insight in these kinds of things that he's truly excelled at.

Just because one person is an expert, like if you look at Ben Carson, super, super highly specialized neurosurgeon. It doesn't necessarily mean that his economic views are great, or are very sophisticated. I'm not exactly sure about combining about this fear about this A.I. that we have to watch our backs for, watch out for.

[0:33:14.9] JM: You're one of the anti-fearests.

[0:33:16.1] ZE: Yeah, most definitely. I don't think that we have to fear future A.I.

[0:33:21.5] JM: Even if it's tail risk, isn't that like worth considering?

[0:33:27.0] ZE: The really, really probability that we have tail risk, I think we have the potential for a lot of safeguards when we're building these systems, that there's ways to sort of build structures around — Build safeguard instructions around these kinds of systems, and much in the same way that we build structures and systems around humans in order to control their power.

If you look at, say, our new president, we have a government which has a structure system that limits the amount of stuff they can do with immigration, that limits the amount of power he has. There are same ways that developers of A.I. have maybe even more powers of controlling these kinds of things.

[0:34:06.5] JM: I agree with you in theory, but it will be interesting to see what happens in practice, because, basically, the narrative around this right now is you've got the Bostrom people who are like, "Okay, we need to worry that if we optimize a machine learning based factory, maybe, with nano machines that are doing the construction of something to make paperclips, that this machine learning based factory will optimize for making paperclips so much that it turns every human in the world into paperclips, because it's just like, "We got to produces many paperclips as possible at all costs."

The reason this idea is compelling is because we're in this industrial race to have the most sophisticated flexible machine learning. Even if we just look at like Alexa versus Google Home, it's difficult to imagine these companies restraining themselves. I think it's sort of like asking the oil companies to start talking about global warming, or climate change, in more realistic terms. They're not going to do it.

Right now, obviously, we're at a point where there's not really a significant risk, our machine learning systems are still so infantile. But just looking at where we are today and like yeah,

maybe open A.I. will do something, but it's seems like we're going to be very exposed to whatever tail risk there is.

[0:35:32.7] ZE: I think one place where we've seen this for sure is — And I think what happens is when you have economic incentives to optimize a gray area — If you look at Facebook, and Twitter, and social media in 2017, they've become so effective about optimizing their reward cycles about optimizing their —

[0:35:53.3] JM: Engagement.

[0:35:54.2] ZE: Engagement. That they've essentially fractured modern attention spans.

[0:35:59.8] JM: Yeah.

[0:36:00.4] ZE: My generation, generations like — If you look at the attention span for people when you're using social media, it's fractured, your day is fractured, your sort of thinking is fractured. You're always being interrupted. You have to pull away from all that in order to actually do any kind of deep work if you're trying to sort of focus on something. I think that's an example of where — It's an example of this process.

We wouldn't necessarily call it machine learning, but in a sense, it is this iterative process where there was some economic signal, and a lot of engineering effort went into sort of optimizing these applications and these websites to sort of become really, really good at sort of getting engagement, and sort of pushing on, "How we get people to sort of pay attention to us and focus on us?" To this point where it's become a very major part of a person's day to now sort of spend time on social media.

Even if there's like these massive economic incentives and we see these kinds of iterative process where people's attention is being sort of pulled away from them, there is still, in a sense, like fundamental limits to what can be done there. There's still, in a sense, you have this iterative process, and increasingly people are spending more and time. It's not like we're going to come to this end goal where eventually we're going to be spending all of our time on

Facebook, all of our time on social media, or somehow we're like sort of sucked into this entirely.

In a sense, I think the same thing is going to happen with machine learning, is that we're going to have — What the bigger issue is that we're going to have — We have this iterative process right now, we're increasingly realizing that a lot of the work that's being done in the 21st century is increasingly doable by machines, doable by machines plus humans, and we're doing these multiple passes.

We started in industrial evolution with farm work, then we started automating a way a lot of sort of basic repetitive work and we're doing increasingly more and more. We're starting to automate away more knowledge-based work. We're doing these multiple passes, but I think we're not necessarily going to hit this point where everything is going to be automatable.

[SPONSOR BREAK]

[0:38:10.5] JM: Couchbase is a NoSQL database that powers digital businesses. Developers around the world choose Couchbase for its advantages in data model flexibility, elastic scalability, performance, and 24 by 365 availability to both enterprise web, mobile and IoT applications. The Couchbase platform includes Couchbase, Couchbase Lite, which is the first mobile NoSQL database, and Couchbase Sync Gateway.

Couchbase is designed for global deployments with configurable cross data center replication to increase data locality and availability. Running Couchbase in containers on Docker, Kubernetes, Mesos, or Red Had OpenShift is easy, and at developer.couchbase.com, you could find tutorials on how to build-out your Couchbase deployment. All Couchbase products are open source projects.

Couchbase customers include industry leaders, like AOL, Amadeus, AT&T, Cisco, Comcast, Concur, Disney, Dixons, eBay, General Electric, Marriott, Neiman Marcus, PayPal, Ryanair, Rakuten/Viber, Tesco, Verizon, Wells Fargo, as well as hundreds of other household names.

Thanks to Couchbase for being a new sponsor of Software Engineering Daily. We really appreciate it.

[INTERVIEW CONTINUED]

[0:39:39.5] JM: Let's zoom out. You are working in the Stanford A.I. Lab, and you described your interests as being focused on high-risk projects. What is a high-risk project in your mind?

[0:39:54.2] ZE: Fundamentally, a high-risk project is something where it's a high-risk. For me, it's something where you're not — There's a small chance, I think, small chance of things working out, of things sort of coming together and everything aligning together in order for this —

[0:40:10.2] JM: Low probability, high upside.

[0:40:12.0] ZE: Yeah, exactly. It's kind a project where it sounds a little crazy when you talk about it, so you don't necessarily talk about it publicly, but you work on it, and when you something to show for it, then you come back and you say, "That was a crazy idea, but we pulled it off."

[0:40:28.0] JM: Could you give me an example? When you talk to the public, do you speak more modestly, or, "Yeah, I'm optimizing models."

[0:40:35.1] ZE: I'm doing a lot of debugging. There was some recent great work out of the lab. It's on the cover of Nature this week, which was doing skin cancer detection using sort of a real big convolutional neural network. Essentially, what it turns out is that — this is a work from colleagues in my lab. It turns out, if you drain a big neural network on 100,000 images of skin cancer, these are images that are just collected from Google and you just scrape Google for images of skin cancer. You could match performance of these amazing Stanford dermatologists just by having a simple algorithms see lots and lots of examples.

[0:41:12.2] JM: How much of that is the headline? How much of that is it like, "Well, there are these certain situations where, or many situations where the algorithm would have not have

recognized it, because it's some weird edge case." Does that not happen? Are the algorithms good enough at this point where, like, even the edge — Because, I mean, the narrative for a while was, and I remember talking to Oren Etzioni from the A.I. Lab, the Paul Allen's A.I. Lab, and what he said was — and we had a lot of conversations, "Oh, it's like the centaur chest. Basically, you've got the human working together with the computer, and the human can identify edge cases, and the computer can identify things that the algorithm specializes in."

I guess, what I'm wondering is, was the determination from the test that, "Yeah, we really don't need the human anymore."

[0:42:08.2] ZE: No. I don't think that was the goal of the — Sort of at that, and we're definitely not at that state where the human is redundant. I think, fundamentally, what this kind of technology allows — The power of this kind of technology is that it allows the ability to scale sort of the quality of healthcare that we have with sort of the experiments that was done in Stanford dermatologists. You can now scale this across the world in communities that don't have access to, say, an amazing Stanford dermatologist. If you can match a performance, it gives you this ability to scale beyond sort of these restricted communities.

[0:42:45.8] JM: Of course! These medical testing costs are so onerous.

[0:42:49.3] ZE: Yeah, they're very, very expensive. It's because you have sort of a really highly paid doctor, essentially. There's an interesting paper where this fantastic work where they show they you can train a pigeon to detect breast cancer with about 100%, or close to 100% accuracy. You just feed it. You have this reward cycle where you set up where you feed it when it makes a correct answer, and it has to choose whether something is cancerous or not. It's amazing —

[0:43:14.0] JM: You're kidding me.

[0:43:14.7] ZE: No, I'm not.

[0:43:15.3] JM: It sounds like a joke.

[0:43:16.1] ZE: No. It's absolutely true. It's a peer reviewed study. It's actually pretty cool. Fundamentally, what's happening here is that you have, basically, the basic pattern recognition happening here, and you have this really highly trained, amazing doctor who has gone through four years of medical school, has gone through residencies, and all these things, and has a much more broader skill set than just being able to look at a picture of a mole and determine whether it's cancerous or benign.

When you sort of take that sort of really broad skill set and have it do one thing, that one particular narrow thing, it turns out you could take a random grad student and have them look at a lot of images of skin cancer and perform close to that level, just because a lot of that terminology, it turns out you can just take a neural network and train it in a lot of images and it will perform close to that level.

[0:44:05.4] JM: Yeah, okay. Humans needs to specialize more. I got a couple of more questions. Ongoing conversation I've had with some of the guests recently. How we get to general A.I., or what it will look like when we get there, and sort of the dichotomy between narrow A.I. and general A.I., because the narrow A.I., we could classify — I think Google Homes. This is like the perfect example for this. You look at Google Home and like, is it narrow A.I. or is it general A.I.? If it's not — It certainly doesn't seem narrow, it can do tons of things for you, but it certainly doesn't seem general, because general is what we've classified as humanlike intelligence. Do you think this question is even interesting enough to discuss, or have you thought about it at all?

[0:44:45.8] ZE: Yeah. I guess, for me, the perspective on this is that if you look at something like Google, Google Home, or look at Amazon Alexa, the question is sort of, "Do you believe it's on a path towards a general intelligence?" The kinds of things that are working really well in machine learning right now and A.I. right now are predominantly supervised learning on large scale datasets.

That's what's sort of driving a lot of the advancements that we're seeing in products with machine learning products. My personal view — and this is a view that people in the field will have, other people will have opposing view — is that moving along this path is not going to lead to general intelligence. That's why I wouldn't call it general intelligence, because I can't imagine

with sort of order of magnitude is more ensuring effort, sort of moving along this path resulting in a general intelligence.

[0:45:32.8] JM: It's just a smartphone, it's a voice smartphone.

[0:45:35.4] ZE: Yeah. For me, the way I think about this is distinction between narrow A.I. versus general A.I., it's not the current skill set right now, but, instead, if we were to devote orders of magnitude more work to this, would we hit diminishing returns? Supervised learning on large scale datasets is a perfect example of this, where if you look at the performance of these kinds of algorithms, as you increase the dataset size, the performance goes up, but in order for every sort of relative delta improvement and performance, you need orders of magnitude more data.

Initially, a small amount of data will get you really high performance. As you, sort of keep on pushing on the performance metric, you need orders and orders of magnitude more data in order to be able to get same amounts of relative performance increases. It's because of these diminishing returns that you hit this like sort of plateau, where your systems aren't able to generalize in a way that you'd expect a general intelligence do.

[0:46:28.4] JM: That makes sense for me for specific verticals, like voice recognition, or recommending a restaurant, or whatever. But what I wonder is, once we get to the point where it's ensembling these things really well, maybe there's some creative ensembling you can do where you sort of get these more spontaneously generated step changes in how it gets closer to general A.I. Does that sound not possible?

[0:46:56.7] ZE: Even with ensembling, I think you'll see the same sort of behavior, this trend, where —

[0:47:01.6] JM: I'm sorry. Maybe, perhaps I used the word ensembling wrong. What I meant was like the synergies between the different high-level models that it's building, like, "Oh, it can do restaurant recommendations really well, and then it uses that with the improved voice recognition. Maybe that gets us a step change."

[0:47:17.3] ZE: Yeah. The term that you're sort of pointing out, I think it's absolutely — This is key and this is what's going to unlock a lot of really amazing products, a lot of really amazing experiences in machine learning. Where to someone who's just using this system as a user, it might be indistinguishable from an actual person on the other end.

For the average user who's using a kind of system with, as you say, ensembles and multiple components, and it covers pretty much all the head cases, all the major cases, and a few of the tail cases for, like, "This is how a person might use a system." It will be pretty much indistinguishable from a human. But any kind of adversary, or any kind of person who's like prodding the system to like try to determine, "Okay. Is this really human, or is this not human?" If you talk to a human, it's easy to be adversarial and for me to determine whether a system is a computer or a human.

You can move in directions where you increasingly find that because that sort of space that a computer is expected to know, is so large, because of how broadly competent humans are, that it's really easy in this really, really, really large space to find a place where a system will fail. It's like these kinds of individual components, these kind of ensembles might combine together to blanket a space, blanket most of the dimensions of intelligence that we have for humans, but it's obviously really easy to find one particular area as a human to say, like, "Okay. Let me just try to figure out if I talk about this, or word it this particular way," is completely not the way a human will respond.

I think this is a fundamental limitation of the current ways that we're training these kinds of systems. Basically, this is a very long description. What I basically say is that these kind of systems are really, really fantastic for most of the cases that you interact with them. The reason they're not a general intelligence, is because they don't cover all those really, really, really rare edge cases that an adversary might sort of try to poke and prod in order to determine whether a system is a general intelligence or just like a really big algorithm to train on large dataset.

[0:49:09.3] JM: Of course, what's weird is that it's going to cover a superset — Not a superset. It's going to cover — I forget what the set terminology is, but it's going to cover things that a human could not do, but it's still going to be able to be incapable of doing certain things that human can do, which is maybe why Kurt Weill believes it's not going to be exactly resembling a

human, but it's going to have humanlike creativity. Anyway, I don't want to go down Kurt Weill's route.

Just to close off, assuming you've got a little more time to go over, industry versus academia is quite an interesting question these days with every A.I. lab being under threat of being acquired by Uber, or Google, or something. How is research in industry versus academia these days? Are there questions that industry cannot ask, because it is too basically scientific that academia is still is better asking questions about? Or are these basically one and the same, where the industries have gotten so good at long term thinking that they're asking good enough questions that they might as well be academic?

[0:50:13.1] ZE: Yeah. One thing where there's — Within academia, these people who aren't working on really long term projects, which is, I think, fundamentally a mistake. I think Google, and Facebook, and these big industrial labs are really well-suited, because of the kind of resources they have to explore more medium term, large scale, really pushing the limits of computation.

I think we have a few bets for like how to build a general intelligence, but I don't think we necessarily like sort of — We know the path that that will lead to a general intelligence. We have a few bets, like, "Okay. If we move along this direction, we'll sort of build towards a general intelligence, or if we move along this direction, we have the ability to build a general intelligence.

Those are, in a sense, fundamental questions. They're like fundamental, really long term questions for us to be able to answer that will lead to a general intelligence. That's where academia shines. I think, right now, there's a lot of funding to be able to pursue that kind of stuff within industry. You see a lot of labs thinking along those directions.

There's a lot of really, really fantastic work being done there. I think a lot of really great work, because resources give you an ability to run these experiments on a scale that isn't really possible in academia. But there are a lot of fundamental questions and a lot of, in a sense, like experimentation and research is you have to be open to a lot of crazy ideas and you have to explore a lot of crazy ideas.

Fundamentally, as soon as you have the resources at the level of Facebook, or Google, in order to — You have sort of a less of ability to really tackle a crazy idea, because you need to be able to have some kind of progress. You have all these resources and you're sort of pushing forward on something. You become more blind to those really, really out of the box, really crazy ideas, than you are when you have sort of more constrained resources.

I think one other thing that's going to happen is that we see a lot — We're in this sort of heap cycle of funding for these kinds of industrial labs. What my concern is, is that sort of this funding can be — If you look ahead 20 years from now, there will be some really visionary long thinking people, long thinking CEOs still funding these kinds of labs. But the amount of funding that's going to happen for industrial A.I. labs is going to go down.

You have a lot of sort of me-too funding with kind of industrial A.I. labs where a lot of people sort of are funding these kinds of things because everyone else is funding them. In order for sustained long term development, don't think that we're going to necessarily like see this kind of really, really long term investment form industrial labs. As soon as like that sort of economic inventive goes away, there's only going to be a few really stars of excellence in terms of industrial A.I. labs, but I think a lot of it will go down.

[0:52:54.2] JM: I hope not, because this is like the ultimate prize, and it's essentially like — I have trouble fathoming that, because it's like until we get to general — It certainly seems, right now, we have all the tools that we need to build general A.I., or we're getting there. It is within sight. It's crazy. If you are the company that wins that prize, congratulations, that's the last invention we need. You're pretty well-suited.

Anyway, yeah, interesting times. Okay, Zayd, thanks for coming on the show. It's been really great talking to you. I really enjoyed your article, and I look forward — I think it was your only blog post. I look forward to seeing more blog posts from you.

[0:53:35.0] ZE: Yeah, you'll see one very soon.

[0:53:37.0] JM: Okay, great. Zayd, it's been a pleasure.

[0:53:39.5] ZE: All right. It was great.

[END OF INTERVIEW]

[0:53:45.6] JM: Thanks for Symphono for sponsoring Software Engineering Daily. Symphono is a custom engineering shop where senior engineers tackle big tech challenges while learning from each other. Check it out at symphono.com/sedaily.

Thanks again, Symphono.

[END]