SDS PODCAST

EPISODE 972:

IN CASE YOU MISSED IT IN FEBRUARY 2026

Jon Krohn:	00:00	This is episode number 972 our ICYMI in February episode. Welcome back to the SuperDataScience podcast. I'm your host, Jon Krohn. This is an in case you missed it episode that highlights the best parts of conversations we had on the show over the past month. I'm taking my first clip from episode number 965. It was a technical interview with Will Falcon, co-founder and CEO of Lightning AI. In this excerpt, Will explains how he converted his wildly successful open source project PyTorch Lightning into a startup with over $500 million in annually recurring revenue. You're in this great ecosystem, both with this commercial relationship with Facebook, but also all the open source stuff that's encouraged by Facebook, by NYU. What were the conversations that led you to funding a company? Was it because you'd already done it with NextGenvest?
Will Falcon:	00:53	So I did not want to start a company, first of all. I think if you've done startups, they're hard and I just come from a grind. And so I was very happy to become an academic for a bit, read. I mean, it was amazing. I was reading books and doing research and I love science. I just love thinking about things, going into really hard impossible problems and going as deep as I can. So if everyone's thinking ever, should I do a PhD or not? That's what I would ask you. Do you love science so much that that's what you want to do? Because if all you're doing is for a career step, you're going to hate your life two years into it. But I loved it. It was amazing. And so I did not want to leave. I mean, the problem is PyTorch Lighting took off and I wasn't trying to make that a thing.
	01:36	It just took off and people started using it. And so then VCs started pinging me. They started emailing me being like, "Hey, we see your open source project. It's working, this and that. " And I was like, "Hey, I'm good. I just sold a company. I just want to focus on yourself." And they're

Jon Krohn:	01:51	Like, "I'm more interested."
Will Falcon:	01:52	Exactly.
	01:53	And now at the time I think about it, all these things have names today, but what I was doing back then was pre-training world models. Today that has a name. Back then we didn't call it that. But I was training me pre-training models on four, 8,000 GPUs, single person, doing all the things that the pre-training folks are doing at OpenAI and whatever. How should you scale the thing? What gradient should you use?What's the distributed strategy? How do you prevent faults? And what if the loss function is this? So most of our research at Fair, at least for me, was more theoretical. So it's more like how do you change the math function? So every model has a loss function that optimizes something. And so it's more like, how do you change the math function? Do you add a regularizer here? Do you use this one or this one?
	02:35	And so I think what I learned was cool and I have lost that skill a little bit, I think. I need to get it back is when you leave undergrad and you've done math, you're really good at reading math. It'd be like learning a language and you're really good at hearing it, but speaking it is something you have to practice. And speaking it is more like, let's say you wanted me to build a model to model this world here that we're in right now, I'd have to figure out how to model the wall and what does gravity look like and how does a lighting hit and this and that. And that's a math equation. So being able to translate that to math, that was a skill that we developed a lot. And that was what FAIR was really good at.
Jon Krohn:	03:12	Oh, really?
Will Falcon:	03:13	Yeah.

Jon Krohn: 03:13 That is a cool skill.

Will Falcon: 03:15 Exactly. And so then that's what we train models with. It wasn't, "Oh, here's a model, train it. " No, it's like, what does the training algorithm itself look like? And if you want to have better representations of the world, how do you accomplish that? Do you make things come together or separate? Do you embed images or not? And if so, how do you do it? Do you use this type of embedding or this type of function and what's a similarity between them? And what if you add a penalizer and regularizer and this? So most of pre-training today is actually not that. It's more like the engineering of pre-training. We were not just doing that, but we were also doing the math and that's fair.That's why fair was what it is. These are math people who are also really good engineers and they can solve these problems together.

03:53 And so that was called pre-training. And so yeah, I mean, I think the VCs were kind of onto something, I guess, very early and they asked me to come in for meetings and then I flew out on a Thursday and then by Monday I had 10 term sheets and it's like I kind of got out of hand. And so I was like, "Well, I guess I'm starting a company." And so I called my advisors and I was like, "Hey guys, so I got all this money that they're offering me. Should I do this? " And they're like, "Well, your research is going super slow, so probably." No, I mean, I've been telling them, I was like, "Hey, listen." Because I kept getting dragged into coding every night. And so I was like, research, run, submit my training runs, okay, it's running. And then I'm like, okay, go merge this pull request from PyTorch Lighting.

04:34 And then it was stuff that I wasn't doing. People were like, "Oh, can I have this function for R&N the way you do a loss function for an RN backprop through time." I was like, "I'm not using backprop through time, but not have

to implement it for you. " So I started working for people. I was like, "This is terrible." I almost shut down Python Sliding in September. I was like, "This is distracting." And so my advisors were already kind of like, "Hey, your research is a bit slow." And I was transparent about it. I was like, "Hey guys, I just keep getting dragged into this.

Jon Krohn: 05:01 " We're going to keep on looking into training algorithms and all that essential pre-training work we do with an excerpt from my interview with Tom Griffiths in episode number 969. I asked Tom, who's professor of both computer science and psychology at Princeton, as well as author of the bestselling new book, The Laws of Thought, how he might adapt our understanding of human intelligence to guide designs for AI systems. You frame human intelligence as rational adaptation under tight limits. So explaining biases, heuristics and few shot learning, so learning from a few examples, which is what LLMs don't do, as optimal use of limited time, limited computational resources and limited bandwidth for communication. So it seems like that ... I didn't even really get to a question there, but the question that I was going to have, you kind of already started now getting into, but maybe with that context you can develop on your answer even further.

05:52 The question is, how should those constraints guide the design of the AI systems that we now build?

Tom Griffiths: 05:58 I think one thing is it's not necessarily the case that they have to. So I would think about those constraints as a good way of characterizing what the difference is between human minds and our AI systems, right? So human minds solve a set of computational problems that are characterized by these constraints. We live only a few decades, give or take. We have to do all the things we're going to do using just a couple pounds of neurons, right? And we can only communicate with one another by

making weird honking noises or wiggling our fingers on a keyboard, right? Those are the kinds of constraints that humans operate under. And so as a consequence, we have to be able to learn from small amounts of data because we're just not going to live long enough to see lots of data. It's not like AlphaZero AlphaGo, which is trained on many human lifetimes of Go Games.

07:00 You can't do that. Human GoPlayers are going to be limited by what is possible to learn in a human lifetime. You have to be able to recognize the structure and problems so you can use previous solutions that you found or come up with good sort of efficient strategies for solving those problems using the limited computational resources you have. So unlike Deep Blue, a human chess player can't search 100,000 positions per second, they can search much less than that and they have to focus on finding the good ones to search. And you can't overcome those constraints by being able to transfer the data that we've experienced or pool our computational resources directly because we have to operate under those bandwidth limits. And so we have to come up with good conventions for allowing us to do things like work together and to pull data, things like writing language in the first place, but writing science, starting companies, right?

07:59 These are all mechanisms that we can have for being able to coordinate the resources that are available to us. And so that set of things really characterizes in some sense what it is to be a human as an intelligent system. And our AI systems are not subject to any of those constraints, right? They can learn from many human lifetimes. They can add more compute when more compute is needed. They can transfer state or share the data that they train from. All of those things are much more straightforward for your AI systems. So one consequence of that is that you might expect that the AI systems are just going to be

a different kind of intelligence from human intelligence, right? Because they're not operating under the same constraints, they're not solving the same computational problems. They're going to be able to solve problems potentially really well.

08:43 And in some cases, they're going to solve those problems better than humans, but they're not necessarily going to solve them in ways that are recognizable to humans or ways that are the same as the solutions that humans find. And so that sets up sort of two possibilities. One possibility is we say," That's fine. We just want to engineer the best systems we can. We should just keep on doing what we're doing, keep going in that direction, and we're going to get this kind of divergence between what human intelligence is like and what AI is like, and that's okay. "The other approach you could think about is one of saying," Oh, well, actually we want some of those attributes of human intelligence to be inside our AI systems. In particular, it might help us make AI systems that just make more sense to us in the kinds of solutions that they find.

09:27 And so if we want to do that, then we really need to understand the things that are going on on the human side and think about how it is that we transfer those over. And so we talked about metal learning is a trick for building systems that have inductive biases that are maybe more like humans. That's certainly an avenue that you can go down if you want to do that, but engineering inductive bias is hard. Scaling is much easier than engineering inductive bias. And that's the reason why we're in the current sort of moment that we're in AI is that that's a technique that sort of works and it's an engineering problem. We know how to solve it, but engineering inductive bias really requires having an understanding of the problems that we're solving and what reasonable inductive biases for those problems are.

And it's something that people are still trying to figure out how to do well.

Jon Krohn: 10:11 I'm excited to see how the complexities of inductive bias based on human experience actually get implemented into AI models. I don't expect it will be long before we start to see real development. In my next clip, I'm turning to episode number 963. In it, Antje Barth, a member of technical staff at Amazon's AGI Labs, came to the podcast to tell us about their latest product, Nova Act, and what it could do for AI developers. You drive the vision, strategy, execution for how developers engage with Amazon's next generation AI products. And the latest exciting AI product you're promoting is something called Nova Act. So tell us about Nova Act.

Antje Barth: 10:50 Yes. So Nova Act is a service that we just recently launched. We had a research preview going on since March last year and are super excited that this past December at our AWS reinvent conference, we launched this as a GI service. Nova Act helps you to build UI automation tasks at scale very reliably and helps you to really kind of start prototyping and putting it in production really fast. So you can get started, which I love as a developer really fast in a playground experience and then iterate on it, deploy it, and once you're ready, push it onto the AWS side and run it there reliably and safely at scale.

Jon Krohn: 11:34 Really cool. And it's free to get started, right?

Antje Barth: 11:36 Absolutely. So one of the key things, again, I'm excited as a developer is we want to make it really easy for you out there to get started. In this industry, we know the speed to delivery, speed to shipping is kind of the mode. So you don't want to spend too much time spending up the right environments and infrastructure and integrations. You really just want to validate your ideas. A lot of startups,

you just want to go build the ideas you have, validate them really quick, iterate on them. So you can get started really quick in our playground experience to especially do that. And then you iterate there and then you can move into the next step. For example, if you want to customize more in IDE environments. So we really want to keep also the surface area really kind of where you're doing your day-to-day jobs as a developer.

Jon Krohn: 12:25 Really interesting. Would you be able to walk us through the typical user journey through a Nova Act experience? Obviously, I'll have a link to Nova Act in the show notes for this episode so people can go there and describe to them what it would be like to us to experience it as we go there for the first time and we're playing around in the playground all the way through to deploying.

Antje Barth: 12:47 Absolutely. So you can go to nova.amazon.com/act and then there's the playground experience. There's also links to all the dev tooling we're offering, IDE extensions, SDK downloads. But really kind of the first step is you go into the playground free of charge, you don't need any AWS account or anything. So you just go in there, you provide a website. So for example, you're going to, let's say a booking website or a specific event, maybe sign up site, right? Conference season is starting soon here in the Bay Area and everywhere. So you just put in the website and then in natural language, you can decide and put in the actions to take. Let's say I want to sign up to an upcoming meetup, right? So maybe I put in the Luma website and I'm going to say, "Hey, search for a specific meetup. Maybe I want to join an AI performance meetup.

13:41 I look for that. " And then you can also put the actions in to fill out, click the RSVP, click the join it and have it fill out the form for you. And on the same playground, you see an embedded UI, the browser environment. So you don't have to set that up any manual way. So you can see

it right there. So you can observe how Nova Act, how the agent is going to that website. It's performing your actions. You see also the reasoning traces of what it is doing, which is exciting, especially important for developers to be able to debug, to troubleshoot, to really see what's happening there. And you can tweak it if it's not getting it at the first time. So you can optimize your instructions, see it. And then once this is performing well to what you want to achieve, you can then download the script.

14:31 So in the background, it's writing a Python script that captures all of those steps. So you have the ready script. You're using natural language, it converts it to the code for you. And then we have IDE extensions, for example, or just an SDK. You get that into your preferred ID of choice, you import that script, and then you're basically back in your coding world as a developer. So you can customize it, you can tweak it in there. And also the IDE extension has this embedded live preview. So a lot of times when we're building those automation workflows, you have a separate window popping up that shows you the browser. And we got a lot of developer feedback that that's just a little bit too much. We want to stay in the flow when we're building something. So we included that in your IDE. So you can really stay in there, have a unified window with all the troubleshooting, the debugging, the traces.

15:23 So you can keep really close eye, develop, customize. And from the IDE, you can then, if you want to, also connect to your AWS account if you have one, and for example, deploy it there to run it in production.

Jon Krohn: 15:35 Nice. So that sounds like a really easy way for somebody to be experimenting with developing an agent because you can go to nova.amazon.com/act and then be able to use your natural language to describe what you'd like your agent to do. Watch it, do that task, and then get the

Python code, use it in whatever environment you're comfortable writing your code in, and then that allows you to easily scale up whatever you're doing. It sounded like there was also, it sounded like you mentioned also being able to productionize on AWS with this solution.

Antje Barth: 16:04 Right. So a core kind of motivation for us, talking to a lot of customers, talking to developers out there, we've seen so many flashy demos, right? You go to any meetup, especially here in the Bay Area, but as you know in other parts of the world in a similar way. And everyone has a flashy demo and all the kind of fun stuff that agents can do. But what we observed and the feedback we received as well from customers is that on average, those agents work maybe 60% of the time. And to be honest, an agent that is reliable 60% is 0% useful, right? Especially if you want to productionize this, you need to really have reliability that you can trust those workflows and that agent to complete that one task. So this was the core motivation for us, kind of really the P0 to make sure Nova Act can reliably 90% and more deliver on those workflow executions.

Jon Krohn: 17:00 Awesome. So how do you do that? How do you get that kind of confidence? How do you go from a 60% reliability to better than that? What kinds of tooling do you have in Nova Act to ensure that?

Antje Barth: 17:12 Yeah. So I want to talk a little bit how we train Nova Act, which is exciting. At least I find it super exciting. So in the past when we trained AI models and things, it's a lot about imitation learning. You collect data and you show it that data. Now, as we're moving from kind of the chat-based conversational models that we all know and then work really well into this space where we're building agents that need to take an action, that doesn't work anymore because the agent is not just predicting a next word, next token anymore, the agent needs to predict the

next action to take. So what we're doing is we're building out those reinforcement learning-based web gyms, as we're calling them. We have actually one in the playground. You can actually play around and observe one. And those web gyms are replicas of very typical UIs, so like maybe a form filling UI, a shopping workflow, et cetera.

18:12 And we're letting the agent train in those web gyms. So imagine hundreds of those gyms, like typical UI design elements, there are typical tasks to do, and then give the agent thousands of tasks and they basically self-play in there. So this is similar in the past, how AI learned to play chess, how it learned to play go. It's really kind of a trial and error approach. So the agent goes in there, tries to do that form filling. It might fail a couple of times, but itself corrects. It does it again and it learns a better way to achieve the task. So this way, the agent understands to reason through this UI and understands how to accomplish the task and helps it also to generalize well, because UIs change. So that's super important. If you're building that agent, yes, you have a specific site, maybe you're in those gyms you're training it on, but then again, you wanted to generalize if the website changes, if the checkout button moves, if the sign-in button moves somewhere else, if it's using different icons, because a lot of UIs are really designed for us humans to navigate.

19:20 And for us, this is a simple task. If you think about maybe you go and write an email, and depending on which email program you're using, which application, sometimes the button says draft or new or create, or it's just a little pen icon to create a new email. For us humans, we learn that. We understand that it's not a hard task for us. But if you're sending an agent to that environment, to that UI, the agent needs to be able to generalize and understand and reason in a similar way, even if some buttons and some tool says, draft, the other

one says create. So this is really exciting. So you're training the models and nobody act is trained like this on those web gyms to then really kind of be able to generalize and navigate those webpages even in real life as they're changing the structure and then the look.

Jon Krohn: 20:09 And in my final clip from last month, we're sticking with this potential of creating self-reliant AI systems. In episode 967, I chatted with Praveen Murugesan, the VP of engineering at Samsara, a publicly listed internet of things company with a billion dollars in annual revenue. In this clip, Praveen fills me in on how quantum physics might be the catalyst for creating AI agents that can operate free from human intervention. Something else, another emerging technology that you've talked about in a blog post we caught is talking about quantum computing. So I wanted to get into that a little bit. In an article you noted, and we'll have a link to this article, it's in something called Digenomica. And so we'll link to that so people can read it in full, but you note that connected operations platforms already do many things that transport re-imagines and that quantum computing may continue to enable this by crunching quote, unimaginable amounts of data.

21:10 So you talk about something that would today take a traditional compute platform, 10 septillion years can be done in minutes with quantum. So yeah, tell us about what quantum might be able to do in your space around routing, predictive maintenance, scheduling. And yeah, go ahead.

Praveen M.: 21:33 Yeah. I think when you really think about it, like a very practical example is like routing and scheduling, you could say. I'll give you an example, which is like pretty much everyone who studied computer science has gone through, like the traveling salesman problem, right? It's an NP hard problem, which means just scales

exponentially in terms of like computational needs for you to solve it. And anybody who's trying to solve that problem effectively uses optimization techniques. So like in a simple sense to say it's like a traveling salesman problem where you're basically saying for routing, like a vehicle has to visit 20 stops. Find me the most optimal way for it to visit the 20 stops. I do believe it takes, there are a little over, I think a lot more than trillion. I think maybe quintendal, I think is the next one, 10 trillion, like number of combinations that exist.

22:28 So I think over time algorithms, we've gotten good at approximation algorithms and then like reducing the search space of that solution space and then like coming up with answers. But I think getting to optimal solutions like for the space of like route planning is like a great opportunity with quantum. Now, we just talked about traveling salesmen in terms of the one individual route problem. Now move that along to think about, "Hey, I have to make 10,000 stops, I have 15 vehicles, find me the most optimal." Yes, you can do that. It's also a very classic engineering problem called like the vehicle routing problem. And again, solutions today uses approximations, not like precision answers, right? Now, let's extend that two steps further, right? Now we've talked about stops. Now between stops, what happens is routes, right? Think about it from a lens of like, I need to go from stop A to stop B, I need to parse a bunch of segments to get there and I have options within those segments.

23:36 Now the problem again multiplied to become something bigger. And let's add one more variance into it. Every time, whenever we all drive, we know that we actually have real time traffic lights. That's real time traffic, very real thing. Now, lots of what we are doing is like these approximation models are like just taking some predictive examples and then like not really using closer to real time

data and then like they're just trying to model, okay, this is what I think likely will happen. And then, right? So with technologies, when computes sort of becomes really easily accessible and like the limits goes off, your ability to do things with much more high precision is possible. Obviously with things like routing, you can get like really hypersize and a lot more efficiency, the gains can be achieved. There's always the argument we can make of like, what is good enough, which I think there's like a point of diminishing returns at some point, but like the possibilities are endless and like the different variables you can react to are pretty much endless.

24:47 So I think that what I generally think will happen is like today, a lot of the technologies we have help you make assisted decisions. And I think that's kind of great for you to get into like a world where truly self-operating systems, I don't think we are anywhere close because like with these type of problems or like the compute capabilities plus the level of access to data to do make real time decisions are like limiting factors in that scenario. All

Jon Krohn: 25:19 Right, that's it for todays. In case you missed it episode, to be sure not to miss any of our exciting upcoming episodes. Subscribe to this podcast if you haven't already, but most importantly, I hope you'll just keep on listening. Until next time, keep on rocking it out there and I'm looking forward to enjoying another round of the Super Data Science podcast with you very soon.