EPISODE 1381

[INTRODUCTION]

[00:00:01] KP: Neural networks, in particular deep, neural networks, have revolutionized machine learning. Researchers and companies have pushed on the efficiency of every aspect of the machine learning life cycle. The impact of trained models is particularly significant for computer vision, and in turn, for autonomous driving and security systems.

In this episode, I interview Forrest Iandola, who's the Head of Perception at Anduril Industries. We discuss their Lattice solution, which can plug into a variety of devices, such as ground vehicles, security cameras and drones. It does artificial intelligence and sensor fusion out of box. We discuss field deployments of these systems and how to run machine learning on the edge.

[INTERVIEW]

[00:00:44] KP: Forest, welcome to Software Engineering Daily.

[00:00:46] FI: Glad to be here, Kyle.

[00:00:49] KP: Before we get into your current role, I'd like to know a little bit about your history. Can you tell me how you got started in software in general?

[00:00:55] FI: Yeah, I've been programming – Definitely there are people who have been programming since younger than me, but I started programming at 15. I did an internship at a physics lab called Fermilab, where I got to play around with software to run particle collider infrastructure. And it just kind of snowballed from there. I got into simulating physics problems with lots of computers. And learning how to use lots of computers at large scale made it easy to make the leap into machine learning and figure out how to make machine learning run fast.

And once I get into machine learning, went to grad school and developed this neural net called SqueezeNet, which kind of helped get the research community going in terms of catalyzing

interest on efficient neural networks. So making neural nets run fast. And from there, we took neural nets that run fast and efficiently to the automotive industry. So I co-founded the startup called DeepScale where we made neural networks run efficiently for automotive applications. Sold that to Tesla, and worked at Tesla for a while. And then I started looking for other applications of robotics. And I found Anduril, where I work now.

[00:02:01] KP: So SqueezeNet, does that improve the efficiency and speed of the training, the inference, or both?

[00:02:07] FI: So I worked on all of the above. SqueezeNet, in particular, the idea was to make – Well, actually it's kind of interesting story. So we started out aiming to actually speed up training. So my PhD advisor and I had made the system called Fire Cafe, which allows you to train neural networks on hundreds of GPUs. Instead of taking weeks to train, it would take hours to train. And this is in 2015. So this was pretty cutting edge at the time.

But I was still pretty impatient. So I started looking at how to change the neural net design itself to train faster, and realized that if you had fewer weights in the model, that the model would be smaller and it would take less time to communicate the updates across computers. And then you could have training run faster on lots of computers. We got it down to like three hours with this neural net that we called SqueezeNet.

And our goal was originally just to speed up training. It was actually my lab mate in grad school, Matt Moskowitz, who tried putting SqueezeNet on a cellphone. And he found it was really, really nice, because it took hardly any space on the phone itself. If you were to put SqueezeNet in a app, like for an app update, it's only a few megabytes. And then the really exciting part was the energy efficiency. So your battery actually lasts longer using SqueezeNet because it doesn't have to move the weights or parameters around the memory as much.

So we actually started out trying to solve one problem of making training run faster, and we accidentally discovered a way of making things more efficient on smartphones. And now there's a whole community around tiny neural networks for smartphones. You might have heard of MobileNet, or EfficientNet, or some of the more recent ones.

[00:03:40] KP: That opens up a lot of opportunities in my mind. Do you have any key things you've noticed that have happened on mobile devices that wouldn't have happened without a technology like that?

[00:03:50] FI: I think so. I mean, you look at things like Instagram, or even Facebook itself. There's a ton of smarts going on, in many cases, on your mobile device itself. Ranging from understanding who's in the picture. To filters that stylize the image or make it look better. And these are all things that you could, in theory, run this in the cloud and have it be less efficient. But running on the device makes the whole thing very snappy. You can edit in real time potentially and take a picture and immediately have different options from a neural network for how you want to stylize the picture.

[00:04:25] KP: And broadly speaking, is everyone in automated vehicles, and robotics, and things like that computing at the edge? Or is there reason to take a more client server architecture.

[00:04:36] FI: I think it varies. So in the autonomous driving space, I think a lot of stuff happens at the edge because it's a real-time system. I mean, if you're to go through a tunnel, for instance, and lose your internet connectivity, you don't want to lose your ability to interpret the environment through your neural networks and your perception systems. So I think on-device is important there.

I think in things like security cameras it's much more diverse. There are companies that do mostly analysis in the cloud. They're ones that are mostly on the edge. And I think it really varies what you're trying to do. If you look at what we're doing at Anduril, it's a mix. I think we're skewed a little bit more towards the edge. So we put Nvidia GPUs, pretty small ones, on most of our products. It's something that we've talked about in a number of different venues publicly that we rely on GPUs in the products.

And one thing you'll find is that it's much, much more usable when the neural net and the perception system can immediately give you an answer and then have, if the product is a drone, it can immediately change directions. Or if the product is a surveillance tower, it can immediately decide, "Oh, I need to follow that object over there with my camera."

[00:05:48] KP: SO you mentioned the drones and century towers that are some of Anduril's products. Can you tell me a little bit about how those get deployed? Who buys them and what do they use them for?

[00:05:58] FI: So our biggest customers tend to be different groups within governments. So we have publicly announced various customer engagements with different parts of the US government and the United Kingdom government. And there are other things that you may find out in the future about us working with other governments. If you want to talk later, we can also talk about the ways we might be able to relate to or serve more commercial non-government interests. Those are kind of the typical kind of core customers that we aim to serve. And we tend to solve very challenging problems.

So the customers have a lot of money. They also have a lot of hard problems. And the way we think about it is how can we build a core product that can be adapted to solve any number of hard problems? And so the core product really is a system called Lattice, which incorporates some pretty impressive capabilities around computer networking and AI. And Lattice also plugs into different physical devices. So it can plug into security cameras. It can plug into drones. It can plug into ground vehicles. And it can do AI and sensor fusion within each of those devices as well as across them.

And sort of where this data all flows tends to be to operators who make decisions. So you might think about sometime in the future, science fiction AI that decides what to do in the context of defense or field operations. But today, it's really operators making decisions. And Lattice aggregating this information from all these different devices and giving you the most realistic possible picture of what's going on out there so you can decide what to do.

[00:07:39] KP: So Lattice deploys to a number of places. Is it a software application? Is it a hardware attachment? What is Lattice?

[00:07:47] FI: Yeah. I think it's a deliberately broad term. So I think maybe it's easier to explain by pointing to some of the things it encapsulates. So it encapsulates an app that runs on your computer, or on your smartphone, or even on a specialized – We've shown versions of it

running in VR. And so there's the user interface side. And that connects to data services. So those data services, depending on the situation, can be configured to route through the cloud or not. So if you're in a Internet-denied situation or just an area with poor Internet, you can do everything locally. So you can have each of your individual hardware devices can run the – What would you call it? Like the endpoint version of Lattice, where they're using the AI capabilities, and they're using the networking capabilities, and the sensor fusion capabilities to make sense of a bunch of data. And then Lattice also has a networking system inside of it called Flux. And Flux has high-security. Flux has the ability to plug in different assets. You don't have to type in IP addresses or anything. And you can also add users in whatever configuration you like. And so you route all this data through Flux, and it can go through cloud systems for additional processing and storage. The storage can also be local in the field. And then that information ultimately ends up on your computer screen so that you can make decisions.

[00:09:15] KP: Could you walk through maybe a use case and a setup? I could see someone who, as you described, is in the field without Internet and power. Or maybe there's a reason they need to kind of air gap their stuff. Or maybe there's a cloud solution. There's a variety here. How does one go about getting started?

[00:09:31] FI: Yeah. Let me go over an example application that actually has been deployed to customers in all three of those scenarios of ranging from extremely air-gapped and isolated all the way over to heavily cloud connected. And the use case I have in mind is a counter drone system. Or as people in the business call it, a counter UAS system. And so what this system is responsible for is following. so imagine you've got a place maybe in a fairly challenging or hostile environment, and you've got some perimeter that you want to secure. And what can happen is people will bring in drones. So anybody can go on Amazon and buy an inexpensive drone and fly it where it's not supposed to go. And the joke is you spend hundreds of dollars buying a drone and flying it where it's not supposed to go. And then the person you're going after with this drone has to spend way more money to address the problem on their end. Anyway, so the drones could come in and just be trying to snoop on you. They could be doing worse. They could bring in explosives.

So what our counter drone system does is it finds drones in the sky. And to do that, we use a variety of sensors. We use radars. We use cameras. We use radio frequency sensors that can

sniff some of the information about how the controller or the remote operator on kind of the other team is controlling their drone. We find those drones in the sky. And then depending how the drone's moving, the operator can take a look at the video feed that we have from our cameras and our other data of what the drone's doing. And an option they have is to launch a drone that we make called an Anvil, which flies very, very fast. It goes out and gets underneath the drone from the opposing team, if you will. Flies straight up and crashes into it. And just before it crashes, you have the option to to disable it or call it off and say, "No. That's actually not what I want to go after." And there's a video feed on Anvil the entire time. So all this stuff works sort of because of the backbone of Lattice. So the users who are seeing the data on where the drones are and deciding to control Anvil, that's all the Lattice user interface that they're using. And the ability to talk to all these devices is also via the Flux networking aspect of Lattice.

[00:11:46] KP: So the networking is a challenge in and of itself I would imagine. I think maybe we'll come back and unpack it. Are you also providing the intelligence to that drone that takes in its sensors? Figures out how to get positioned below the adversary and then execute the command?

[00:12:00] FI: Absolutely. So we're in this case thinking of it as basically a full stack system. There are some cases that we can get into later where we can plug in sensors and devices that the customer already has. Kind of the basic kind of drone system, we sell the hardware. A lot of it was developed by us at Anduril. The software was developed by us. And the sensor fusion, the AI systems were also developed by us. So we're doing our own computer vision.

In fact, one thing that's really nice on the computer vision side is the ability to iteratively improve the computer vision model. And this is a theme that you see in a lot of the public information on self-driving car development as well. If you look at stuff from Tesla or Porsche, they talk about this a lot, of you want to put the best computer vision model that you have into the field. Run it on your cameras. But as you start to see mistakes, maybe it occasionally misclassifies a bird as a drone, or there's a rock on the ground that looks a bit drone-like. You can absorb that data. In our case, Flux makes it exceedingly easy to get data back to a larger computer where we can train a new model and we can very quickly put together a new computer vision model that resolves those cases and ship it back into the field. And these are the kinds of things that are

much, much easier to do in the context of a full stack solution than it would be in a more piecemeal solution where we would buy computer vision from somebody else.

[00:13:22] KP: Yeah. And this way you also have a controlled stack. The sensors that are in the device, the camera and all that. So it seems like you're not susceptible to the variants that different manufacturers can have. Is that a part of the strategy? Or could you easily adopt to varying hardware configurations?

[00:13:39] FI: It's somewhere in the middle. I think there are lots of things people say about it's hard to put new hardware in, or it's plug and play. I mean, across the industry, people have different opinions. I think what we've seen is hardware selection is important, right? So when choosing a camera or a radar, it's really useful to put a bunch of different products side by side that we could choose to adopt and run tests in the field where we fly drones or do whatever we want to do to kind of get a feel for how it's really performing in practice.

And occasionally, we'll get into a case where we don't have quite the sensor that we would like. And so we'll work with a vendor to customize it or we'll dig into the details there. Once we have a sensor selected, we also do a lot of work to refine the software that relates to that sensor. So for instance, when you're working with cameras, we work a lot on, "Well, how are we pointing that camera? How are we controlling the zoom? How are we controlling the focus? How are we pointing the camera at an object that the radar has already found?" And maybe working around some of the quirks or the errors that the radar introduces into the system. So these are the kinds of hardware wear things that we tend to do.

And plugging in a new sensor is something that we can do. Naturally, we often have to go through these same steps of characterizing where the sensor struggles. Tuning the software. Maybe modifying the hardware of the sensor just to improve performance. But all these things are possible. And one nice thing is when you're working with a defense where budgets are quite large and the interest the customer has to solve the problems is often very high, we can often afford to make these customizations and put in the effort and it still can be something that makes sense for us financially.

[00:15:20] KP: So building a system that can take out a drone that's invaded some private space where it shouldn't be, I'm guessing with the Flux networking, you can get a full suite of telemetry on an actual demonstration. So that's great training data. But, surely, you can't use real-world training data for all of your training. How do you go about coming up with a model?

[00:15:41] FI: One of the things that we're able to do is internally do our own testing and development. So we have a couple of test sites in California. One in San Clemente, California. One in Apple Valley, California. And those test sites, we have a full suite of our hardware and software set up and we can fly drones and develop. So just as you were saying, we're able to do a lot of pretty realistic testing right from the beginning of developing a new capability. But then once it goes out to customers, we don't stop there. We're able to customize further.

So let's say you have one customer that's in a heavily wooded area. Another that's more in an open desert area. One that's in snow. Depending on the data sharing agreements, we might be able to merge all that together and take the data from those different deployed customer instances and combine them. Or if we're a little bit less lucky with it, what we may end up doing is customer A will have our data set plus data that we collect from their instances of these devices. And customer B will do the same thing. And we can often work our way out of these problematic cases by doing that. Not just by retraining the computer vision model, which we certainly do, but also by looking at how we're adjusting the parameters in the sensor fusion, improving the object tracking, improving the correlation across different sensors as well.

[00:17:02] KP: So deep neural networks are susceptible to adversarial attacks. The famous textbook case is a image of a gibbon or a panda that is correctly identified. And then you add a little noise. And suddenly the network thinks it's a school bus or something like that. Do you have to worry about adversarial attacks?

[00:17:20] FI: I think we probably do. It's not just adversarial attacks like someone trying to pull the wool over our eyes by disguising a certain device to look differently. Although that certainly can be the case. But it's also just the world changes over time. So new drones are developed. Bird migration patterns shift. Airports get put up. And sometimes there can be more commercial flights than there used to be. And the world's always changing. So I think adjusting to these changes is really important.

And one thing that we've been able to do is to have our engineers pretty in the loop with a lot of the customer deployments. Looking at what are the new issues that are starting to crop-up? So certain new kinds of drones emerging, we can spot that pretty quickly. And sometimes the system performs great against those. But if it doesn't, we can work to resolve that. I think in terms of somebody actually like putting a sticker on the drone to make us think it's something else, I haven't seen a lot of that yet, but I'm kind of ready for that when it starts to happen.

[00:18:19] KP: Well, let's get more into the Flux networking. I don't know if every listener is going to be familiar with the idea of mesh networks and things like that. Just to set the stage, could you frame some of the challenges you have around getting information moved from here to there?

[00:18:32] FI: So maybe a scenario that people who kind of follow the news and technology, but don't know as much about we do might relate to, is the self-driving car case. So if you ever look at news or talks on self-driving cars, one thing that comes up a lot is we should be able to have all the cars on the road, the self-driving ones and the normal ones too, all be communicating with each other about what's their position. Where do they want to go next? What obstacles are they seeing? Talk to the stop lights. Talk to the toll plazas, all these things.

And the reality is most this doesn't really happen in practice yet. It's not that the technology is impossible by any means. It's more cost money and also requires some agreement on protocols. So getting everyone to adopt the same thing is hard. And getting people to retrofit their old cars and stuff with this networking stuff is probably a non-starter.

But if you look at the kind of things we're doing where either we have control over the hardware or working with customers who are helping us to use hardware that they already have, we're able to do a lot in terms of taking large numbers of different sensors, and drones, and device and plugging them together. So I actually think that the stuff we're doing in Anduril may be kind of a thing that solves some of the basic problems and helps us learn some of the basic lessons that just like how other technologies have migrated from defense to a consumer, like GPS, that mesh networking in terms of large-scale deployments with lots of different devices maybe something that starts with what we're doing and then works its way out to consumers.

[00:20:09] KP: To what degree are your drones automated?

[00:20:12] FI: So it's a bit case by case. So if you look at Anvil, the drone that we have to go after other drones, the way it works is the operator is completely in the loop. So it wouldn't make sense for the operator to have a joystick or be pressing up, down, left, right buttons on their keyboard. Because if the speeds where Anvil is going, they just couldn't respawn fast enough. But the compromise that we have is the operator decides when it's time to launch Anvil, if at all. The operator can watch a video feed the whole time the Anvil is flying out to the drone that it's supposed to connect with. And all the way up until the very end, the operator can decide to cancel that Anvil mission. And so the level of autonomy there is if you put it in self-driving car terms, it's kind of like a level two self-driving car where the system can do stuff on its own, but it requires human supervision. And that's where it is today.

I think another type of drone that we make is called Ghost. And so Ghost is optimized more for intelligence reconnaissance and surveillance use cases. And so, on Ghost, instead of looking up in other drone, it's looking down at the ground with its camera. And in Ghost, there are various types of autonomous situations that you can set up. You can have one Ghost. You can select an area where it can go fly around and maybe look for a certain target that you've decided to find. I mean, just find all the cars in this sparsely populated area would be an example use case that it can handle on its own. And there's also a whole aspect with Ghost and some of the other drones that we plug into that the customers have, where we can task several drones to work together to go look for something in an area. And they pretty much on their own divide up the problem and decide which drone should go where.

[00:22:03] KP: So if I have a fleet of drones that's out doing some, I don't know, mapping recognizance exercise, I'm potentially getting a live feed. But it occurs to me that could be interrupted. And maybe the device needs to hold some local buffering or things like that. What are some of the challenges you face in a live scenario getting data pushed back or pulled in all the right directions?

[00:22:24] FI: Yeah. So one thing behind the scenes that's going on with a lot of this Flux stuff is using multiple types of networks. So depending on the situation, the drone, or surveillance

camera, or vehicle, or whatever may have any number of different types of network technologies. If you're very lucky, it might just be plugged into the Internet. And that's pretty rare. It may have 4G. It may have satellite Internet. It may have local mesh networking, which is pretty common in our systems. And it comes to local mesh networking, as you say, it's often the case that for a while it will work great. Then, for example, the drone will fly too far away and it has to just reduce its frame rate that's sending data back or just log data locally. These are all things that we tend to be able to account for. And it's honestly a lot of lines of code to put all this stuff together.

One nice thing is once you've done this once, the code tends to translate pretty well. So Anduril bought a company this year called Area-I, which makes a family of drones called Altius. And we were able to take a lot of the code that we've developed on the networking side, the computer vision side, the autonomy side from Ghost and move that over to Altius. And that worked well. And we've also worked with customers who bring their own hardware and move some of this technology over there, and it tends to work pretty well. Obviously, there's some tuning. But it's much, much nicer than having to do all this over again from scratch for each kind of drone or device.

[00:23:56] KP: So a device can always have an onboard hard drive. Although that's something you pay a penalty for when you have to put it up in the air and expend fuel to keep it up there and all that. But that can certainly help where you put some data locally before it's transmitted. Do you have to face tradeoffs about like we need a bigger hard drive because we think it's going to be out of reach longer? Or maybe devices can return to base because they feel their buffers too full. Is that a big challenge that has to be solved in the protocol level?

[00:24:23] FI: I think the hard drive one, it turns out, is fairly easy. But I can give you an example of a harder one. But first, why is the hard drive one easy? So you can buy an NVME SSD drive for very cheap now. And it has hundreds of gigabytes on it. So that's a pretty good situation. I mean, hundreds of gigabytes of video data is actually a long time. But a place where we do have to face tradeoffs is it's surprisingly hard to fit a lot of sensors on a drone. Like it would be great if you could have several kinds of cameras, radar, maybe radio frequency sensors. But you often have to be very lean in terms of your payload.

And in fact, one thing that I've sometimes seen happen around here at Anduril is you'll have multiple different drones maybe even flying in the same area, but we'll put different sensors on different drones. Because as you say, there's issues with how much energy you can store in the batteries and how heavy those sensors are. So a classic example would be you have one set of drones that are running cameras and the others are running radars. And the radars fly around looking for moving objects and then say, "Oh, hey, I found something moving on the ground." And then that alerts one of the camera drones to come over and take a more detailed look in terms of the pixel space.

[00:25:37] KP: So when you have autonomous devices like drones, especially more than one of them, I guess they could be totally decentralized, or they could follow some centralized policy. What sort of ways do you approach them all existing on the same playing field?

[00:25:51] FI: I think one thing that we've done kind of from the ground up is to assume that there will be some customers who need to run purely air-gapped. So what that means is you have to have some sort of peer-to-peer mechanism for different drones and other sensors to coordinate. You can't just wait until you have a chance to talk to the cloud. And so building it that way obviously has been harder. But once you have that, it becomes pretty straightforward to add more devices. And they tend to slot in more neatly when the assumption is that it can be peer-to-peer or centralized and that it's really kind of in the config files where you decide as opposed to being a total rewrite to be able to do the peer-to-peer capabilities.

[00:26:35] KP: You'd mention devices that have access to potentially multiple networks. Maybe it's Wi-Fi, 5G, and satellite, or some combination of those and others. Are you trying to then utilize and, I guess, switch across what's effective at the time?

[00:26:49] FI: That would be really good. I think there probably are situations like that. I think what I see more commonly is the case where there will be devices kind of daisy chained together. There will be one spot where there's really good Internet access. And then the others will form a mesh network that can slurp up some of that Internet access on that place where it's working well. I think that's more common.

And one nice thing about being able to do this is it makes it easier to get data back in challenging scenarios. So you can imagine, some of these devices are in really remote locations. And it's difficult for the operator to get eyes on what's going on. It's difficult to get data back that you could use for training data. And having this capability makes it significantly easier. And you'd think that this whole thing of getting data back and training could be a long process. I mean, when I was at Tesla, they tend to release a new version of autopilot maybe every month, maybe every few months. At Anduril, I've seen people turn this around in as little as a day.

So there was an example recently where we deployed the counter drone system to a new area. They had a type of street light of all things that had an odd shape. And our computer vision model thought some of those street lights were drones, and would just constantly be looking at the street lights of scanning for drones. And so we're able to get a bunch of images of those street lights back and retrain the model, do a bunch of experiments offline to make sure we haven't broken anything else by making that change, and ship the new model to customers. And the whole thing took less than a day. And, yeah, having the Internet networking capability to do that is really empowering.

[00:28:34] KP: Do you end up shipping those new models to just the areas that have the unusual street lights? Or does that just go out globally?

[00:28:42] FI: The street light case, I believe that streetlight data went into models that run all over the place now. So that's one nice thing about the situation we're in. Sometimes people who have companies that work with the government end up doing one thing for one customer, one thing for a different customer. Ends up being very little reuse of technology. I think we've been really fortunate to find a lot of opportunities where we can build something to solve a hard problem. And then one of the customers who's using that product or that piece of technology says, "Hey, I've got an even slightly harder problem for you." And we solved that one. And then all the earlier customers get the benefit of us having solved that. And we're able to push a new release to the whole set of people who are using that particular product.

[00:29:27] KP: Could you elaborate further on what you mean by sensor fusion?

[00:29:31] FI: So one helpful way to break this up would be sensor fusion within a device and sensor fusion across devices. And I guess by a device, I mean, maybe it's a few pieces of hardware, but they're all kind of concentrated together and maybe they're plugged into each other. And so sensor fusion within a device could be in the counter drone system. You have a radar, which the radar we have is actually quite powerful and it can see many kilometers away in all directions. And based on where the radar finds objects moving in the sky, we then point the camera there and get, basically, images on that object.

And so pointing the camera at the radar sounds like it should be easy, right? And sometimes, luckily enough, it is. But if you think about it, you're trying to see an object several kilometers away. That's not something that the computer vision world or the camera world is really optimized for, right? You think of cameras and computer vision, maybe you have a cellphone, or maybe you have a self-driving car prototype with a camera that sees a few hundred meters, right? We're looking kilometers away. So there's a couple of problems.

One is the ability to focus the camera is much more challenging at that range just because of how optics work. And so we have to put in a lot of effort to get autofocus working well. And then another challenge is the radar sort of error bars around where the radar things the object is especially when you're in complicated terrain with all kinds of echoes and things on the radar. Those error bars may be larger than the field of view of your camera, which means when you point the camera at the radar, there's no guarantee that you're going to find anything.

And so the trick we found is you kind of subscribe the camera to where the radar thinks the object is and then you search around it. So you add an offset. You say, "Maybe for a second I'm going to look above where the radar thinks it is by a little bit. And then I'll look to the left." And you kind of jump around until you find the object. But it's these kinds of details that I think often are kind of where a lot of the time goes and working on sensor fusion within the device. And I guess I'll see if you have any questions on that. We can talk about sensor fusion across devices too if you want.

[00:31:41] KP: Yeah. So that's a very interesting coordination problem in a way. So you had to kind of develop this local search for the camera knowing that it should just be pointing in the right direction, but also coordinate when radar found something. It's time to allocate those

cameras. Is that all part of just the Lattice brain? Or does one have to kind of create that for the use cases?

[00:32:02] FI: So that's something that we can – Given that we have that, we can now use it anywhere. So a lot of the code that runs to control the camera allocation, and the computer vision, and so forth is general. So you could imagine on a drone saying, "I'm subscribing my drone camera to a radar," that maybe is on another drone or maybe it's on the ground, "and I'm going to use that same camera search algorithm from the perspective of the drone." That tends to work with pretty minimal effort in terms of porting it. And then of course you have to tune the small constants of, "Well, how big is my search radius?" and these questions.

[00:32:36] KP: I guess my next question is about the extent and breadth of machine learning. Clearly, you use it for computer vision. In theory, it could also be used for decision theoretic problems and planning and things like that. Is that a component of what you do as well?

[00:32:49] FI: I would say, on the machine learning side, as you say, computer vision is probably the biggest application. Another area where we've put a lot of effort into machine learning is classifying objects based on their motion. So you can imagine, if you're classifying human-made air vehicles, like drones and airplanes, versus birds, if you look at the data, what you'll find is human-made air vehicles in most scenarios tend to fly in a fairly straight line. The altitude doesn't jump around too much. It'll have gradual changes. Whereas birds tend to fly these loopy, whimsical patterns. They find a mouse on the ground and they're suddenly diving down and they're coming back up.

So just based on the shape of how the object has moved recently, we can actually use machine learning to infer something about what kind of object it is. Of course, it's not perfect, and it's something that you don't want to rely purely on without other sensors. But it's a good start. And that's something that we use to help decide which radar object to point the camera at next. Is it more of a loopy thing? It's more likely to be a bird. Okay, we'll de-prioritize that. Or is it more of a straight line? Okay, that's probably something we should look at sooner. So that's kind of examples within a device.

There's also across devices. So you can imagine you're following a drone or other air vehicle hundreds of miles. So one thing there is you want to have different sensors along the ground, some which are sensors that we installed. Some are ones that customers have from other sources. And we can hand off. As the object flies through the sky from one to the next and say, "Oh, this sensor and that sensor, each are locally doing object tracking. But we discover they're talking about the same object." And that correlation is something that so far has been more rule-based. But I think machine learning is a great thing to potentially use. In terms of the transition to more machine learning, I think one thing that's been a theme in the self-driving space where I used to work is just gradually replacing or augmenting rule-based things with machine learning. Once you hit some wall of, "I keep adding more rules, but the rules just kind of contradict each other." Or, "How am I going to manage thousands of rules?" That often is kind of the spark of where you start trying out machine learning. And by that point, you probably have lots of data.

So I think if we talk in six months or a year, you're going to see each time there's more pieces that we've converted into either fully machine learning problems or, let's say, machine learning assisted problems.

[00:35:18] KP: What are some of the challenges that you're tackling on that front?

[00:35:21] FI: So I'm very interested. I'm not going to claim to have the answer in tracking based on machine learning, or more intensively using machine learning and tracking. And in computer vision-based tracking, or self-driving car tracking, or maybe you're relying on lidar , and radar, and cameras, it can work pretty well because you can label lots of data, right?

The challenge with tracking with machine learning and what we do is, with the object so far away that the camera, as you remember, isn't always pointing at all the objects at once. It can just look at one thing at a time. You don't necessarily know how that object moved sort of in terms of ground truth. We can get some amount of data where we put GPS trackers on objects and have sort of brown truth of where they flew. But we don't have the level of truth that you'd have if you're looking at close range objects. You've got lots of cameras on them and lidar and everything. So I think we're reading these papers from the self-driving community on tracking and motion planning with machine learning, and we're trying to figure out how that adapts to our case. We're looking at much more sparse far away kinds of data.

[00:36:30] KP: Yeah, you have less obstacles, but an extra dimension almost to deal with.

[00:36:34] FI: Exactly. I think we have much to learn from them. I also think the technology we're doing may benefit self-driving people in certain ways. We already talked about the mesh networking in self-driving scenario. Another one is imagine if you could see kilometers down the road how fast you could drive. Obviously, you have to work out a lot of other problems for self-driving. Self-driving is not a solved problem yet. But if you had all the kinks worked out and you could see a kilometer or multiple kilometers down the road with the kind of thing that we do, autobahn speeds would be pretty achievable with self-driving cars.

[00:37:07] KP: Yeah, that's pretty incredible. Well, when it comes to, let's say, anti-drone technology, the drone you want to get rid of that shouldn't be there, that's controlled by an operator potentially. And if they have good sights on it, they could be doing some avoidance. What happens then? Do you have an algorithmic approach for avoidance? Or does someone have to grab a joystick on your end?

[00:37:28] FI: So the nice thing is if you look at how Anvil works, until Anvil gets very close to the target, it's actually relying primarily on the sensors that are way back on the ground from where it launched, or other sensors around the area that we've placed in strategic locations. And so imagine an Anvil is flying towards a drone that's wiggling around and it's raising and lowering its altitude, it's moving left and right. Well, anvil knows the latest of where it's moved because it subscribed to all these sensors over a wireless network. So as long as we, from the ground stations, can find the object, Anvil can just keep adjusting its route to get closer to the object that that's looking for.

[00:38:14] KP: Got it. Yeah. When you think about pushing the limits of where the software can go, what are you bumping into? Is it compute, training data, something else?

[00:38:24] FI: Those things are certainly challenges. I think I've been really heartened by all the work on hardware for computing neural networks and other computationally-intensive problems. I mean, not just Nvidia, but there are lots of companies developing neural network accelerators. And it's going to be really exciting to see how that develops. I think in terms of software, I mean,

the researchers who are giving away great neural network designs and great software like PyTorch for training and deploying neural networks. That's all fabulous. That makes my life so much easier.

I think some of the challenges are actually a bit tame or not specific to machine learning. So one example would be build systems. So every company has this problem of you start out with a few people. It's a startup. Kind of just walk around the room and talk to people about how everything should fit together. But I think at last count we have 800 people in the company, and we have millions of lines of code, and everybody is pounding on this code trying to make it better and changing things all the time. And how do you just compile your C++ code every time and have it not be too painful? Like some of these just practical things that especially in Software Engineering Daily type podcasts are kind of probably a commonplace. No different just because we're using machine learning.

[00:39:46] KP: Yeah, makes sense. Can you talk a little bit about the tech stack you guys use?

[00:39:50] FI: So I'm most familiar with the tech stack within machine learning lands. We already covered a little bit about the computer networking and some of the hardware. So maybe it'd be good to do a little bit of a dive into how our machine learning system works. So kind of the flow is we get data in from the field and we can choose what data to ingest. We have lots of it on disk. And then we have people label it. So there certainly are approaches with synthetic data or other kinds of data. But, basically, even the biggest proponents of synthetic data. Say, if you can get real data, that's going to be better, is sort of what I've seen.

So we get lots of real data. We label it. We've built our own labeling tools in-house and we've gotten some really bright people. We've recently got someone who worked on labeling tools at, I believe, it was Uber. We have people who are veterans of startups that have worked on ML end-to-end. So really great people working on that. And so we've really been working to tune the efficiency of how do we get to the point where a person can be very, very productive in terms of labeling data?

And we've been creative of finding ways to have the computer and the human work together on labeling to speed that up. And then once we have labeled data and we've organized it into data

sets, the training, as an engineer, you have a couple of options. If you're doing something more experimental, you can just get a GPU computer and train something. We have lots of GPUs. If you're doing something more production-oriented though, we have put a lot of effort into making training neural networks reproducible.

So the situation that we're aiming for is you have basically your neural net that you're training. You have a way of measuring the neural net's accuracy. And you have a system for deploying the neural net. And each of those is its own Docker container with the config file. And so the idea is you first train your neural net. You subscribe it to the data sets that you want to use. You can study the results, evaluate it, deploy it. But if you get hit by a bus or you're just busy, someone else can come in and pick up exactly where you left off, and everything should still work. So the idea is when we get new data in that might be trying to solve a problem we're seeing in the field, pretty much anybody on the team who's kind of just been briefed on how this works can pick it up and work on it. And we're even trying to go so far as to have people who aren't necessarily engineers pick this up and do the data science side where they look at the failures and they get more data and they retrain.

And then when it comes to deploying the neural networks on the security cameras, or towers, or drones, or what have you, we have a model deployment system that exports the model, puts it into – Typically we use either Torch Script, which is sort of a compiled version of PyTorch, or we use TensorRT, which is a library from Nvidia, to actually run the model on images on the device. And that's sort of the workflow. And then you get data off the device and the whole thing repeats itself.

[00:43:01] KP: In terms of deployment, when you hit a use case like that interesting lamp that we discussed earlier, it makes sense you're going to roll out a fix. When you don't have some immediate impending need like that, are you still consistently releasing models like to, I don't know, avoid drift or things like that? Or is it something you want to lock down?

[00:43:20] FI: Yeah. We're always trying to make better models even if we're in a situation where nobody's screaming enough to fix it the next day or anything like that. So the idea, as you say, is there's going to be drift. There's going to be new cases. We'd love to just incorporate that data. The other thing is we try to be pretty clear about what are the scenarios where we're

struggling. So the streetlights were one scenario. But there are always more. We have a pretty big backlog of them.

So one that we solved recently was windmills. So we went to a site that had lots of windmills. And you might say, "Well, why does that matter?" Well, it turns out radar really likes windmills. It thinks windmills actually are moving objects because it's spinning. And so the radar finds it. The camera points at it. And it does have sort of a tail fin to it. And the rotor, you could think of it as maybe kind of like a propeller. And so we kept calling that a drone from our model. And it wasn't like a huge issue. It didn't need to be solved the next day, but it was annoying. And we got more data. And we also collected what we call a unit test set, where it's like some data that we have it's just windmills or just those difficult street lights. Or we have these separate different cases so we can see how we're performing over time on all these challenging cases. Anyway, we did that. We fixed it. And so there's always a backlog of stuff like that. So if there's something really urgent, we tend to just pick up some of the ones where we know we're sub-optimal and dig into them.

[00:44:52] KP: Are there any common false positives like that, like mylar balloons or Chinese lanterns? Anything you really hate?

[00:45:00] FI: So I think the relative size of drones is really interesting. So one of the most popular drones, or at least a popular drone, is called a Phantom 4 made by the company DJI. I was like, "Why is this thing so hard to detect?" I mean, it's tiny. I think it's like a third of a meter wide. So like a foot wide. And it turns out – And people ask me like, "Why is this DJI thing kind of challenging?" We can do it pretty well, but it took a lot of work to good at it.

And then like an airplane will just lazily fly by. It'll be like 30 kilometers away. And we'll have no problem picking that up. And people are like, "Forrest, what's wrong with you? I mean, like this DJI thing is not that far away." But it turns out, a Boeing 737 is a hundred times wider than the DJI Phantom 4 drone. And so the relative size is really interesting. I mean, in theory, on a clear day, we could see airplanes 30, 50, maybe more kilometers away. But these little drones, you got to zoom way in. You got to do this whole camera search. You got to do all this stuff. So, yeah, I suspect though people are going to probably keep making smaller drones. So we're just going tohave to keep working harder.

[00:46:05] KP: Well, are there any interesting things on the roadmap you're able to talk about?

[00:46:09] FI: So we already talked a little bit earlier in the podcast. But one thing that we're working on/is on the roadmap is a whole family of systems called Altius. And since we acquired Area-I, which originally developed Altius, they're already plugged into Lattice. So a little bit of background about Altius. These are what are called air-launched effects. So they can also launch from the ground, but the really cool party trick is you can be flying an airplane and shoot out these Altius devices then they can start flying themselves. And the Altius devices, they have cameras. They have other technologies on board. And you can pretty efficiently send them out to go collect a lot of information and scan a wide area with just launching them from one plane.

And I think this whole Altius thing, I mean, they've only been part of Anduril for a few months. And we've already integrated all this. We've helped the Area-I team work with them to improve a lot of aspects of the system. This is the kind of thing we want to do more of in terms of taking existing technologies, bundling it and plugging it into Lattice, and then doing a lot of field testing, a lot of iteration to make sure the thing really works as expected.

[00:47:22] KP: Are you guys hiring?

[00:47:24] FI: We are hiring a lot. So we've grown substantially. So I've been here for about a year. And I think we went from about 300 people when I started to 800 people now. And we are hiring in Orange County, California, Seattle, D.C, and Boston. And, on my team, we're looking especially for engineers and people who have worked in industry on challenging problems. We definitely have places for people with AI expertise or object tracking or sensor fusion expertise. But we also just like hiring smart people who can dig into problems from first principles and solve them. And I think that's true across the company as well. We have lots of different niches for just smart people who want to move fast and solve challenging problems.

[00:48:15] KP: Very cool. And anywhere people can follow you or the company online?

[00:48:20] FI: So our Twitter is @anduriltech. And our website is anduril.com/

[00:48:26] KP: Awesome. We'll put some links in the show notes. Forrest, thanks again for taking the time to come on software engineering daily.

[00:48:28] FI: Thanks again for having me on the show, Kyle. I really enjoyed it.

[END]