EPISODE 1512

[INTRODUCTION]

[00:00:00] ANNOUNCER: This episode is hosted by Sean Falconer. Sean has been an academic founder and Googler. He has published works covering a wide range of topics from information visualization to quantum computing. Currently, Sean is Head of Developer Relations and Product Marketing at Skyflow, and host of the podcast Partially Redacted, a podcast about privacy and security engineering.

Today, we spoke with Daniel Situnayake of Edge Impulse. We discussed cloud-based dev environments, cloud-based IDEs, infrastructure as code, dev containers and live collaboration.

[INTERVIEW]

[00:00:39] SF: Dan, welcome to the show.

[00:00:41] DS: Hey, thank you, Sean. It's great to see you.

[00:00:43] SF: Yes. I'm really excited to have you on. I was fortunate enough to work with you for a little while during our time at Google. But, I guess, before we dive too deep into this really fascinating field of machine learning on edge devices, can you start by introducing yourself, I guess, to the listeners and give a little bit of background on how you ended up where you are today?

[00:01:04] DS: Yeah, so I mean, I never know where to start with this. I've done such a bunch of random, weird stuff, and it's all sort of led me in this direction to working with machine learning on embedded devices. But what I do right now is I'm the head of ML at a company called Edge Impulse, and we build tools that make it possible for developers to work with machine learning for embedded devices. This is a field that hasn't been around very long. So, I was very fortunate to sort of be in the right place at the right time for all of this to come to fruition.

My previous role, I was working on the TensorFlow Lite team at Google, and helping launch a product called TensorFlow Lite for microcontrollers, which is basically a set of tools for taking deep learning

models, and deploying them to the smallest types of embedded devices, which are microcontrollers, which basically processes, tiny little computers that you can write low level code for. And we're trying to figure out how do you let people take these deep learning models, which are typically pretty big and hefty, and shrink them down so they run on these little constrained devices. It turns out, it's really, really hard to do that, and since those early days, there's a bunch of tooling that's grown to try and make it easier to achieve this kind of thing. That's the thing that I get really excited about. I think everybody has a moment in their career where they stumble onto working on something and they realize, "Oh, I found it. This is my thing." And for me, that was it.

[00:02:38] SF: Your thing is taking machine learning and applying it to this tiny little edge devices, little IoT devices and stuff. It's a real niche, I think, you've carved out for yourself.

[00:02:49] DS: It's interesting, because, it's like a niche, but it's a big niche, right? If you think about what are embedded devices, these are the things that occupy our world, more than any other kind of computer, they're inside of all sorts of things from like your household appliances through to the insides of a car. The machinery in factories, pretty much everything we produce these days that uses electricity, has some kind of embedded device as a component. And so, what is kind of amazing is that it gives us this canvas of the entire world to start deploying machine learning to. So, rather than machine learning being this thing that lives on big servers or on notebooks that analysts are running, it becomes this tool that's directly in contact with the real world in a very fundamental way, and that's what is really exciting to me.

[00:03:47] SF: Yeah. So, you've been in this space for a number of years between TensorFlow Lite, and you've written about tiny ML, and you have an upcoming book, *AI at the Edge*. So, you're not like only an expert, but you're kind of an early believer of this, like, the future of this technology. How did you get interested in the space to begin with? And then what was sort of the key insight that made you I guess fall in love with it?

[00:04:15] DS: Yeah, it's kind of been a long journey. So, I started out, I did a degree in computer networks and security back in many years ago in the United Kingdom, a University called BCU. And that was sort of a real hodgepodge of loads of different topics. And one of the topics I was working on was around auto ID technologies. So, these are technologies that are used to keep track of things in the physical world with a computer. Everything from barcoding, and RFID, which was like a hot new

thing back then, through to computer vision, which was very primitive back then compared to what we have now with deep learning, where we've got these like mobile phone apps that can transform your faced into a dog or whatever. Back then, it was like, “Oh, we can just about recognize where somebody's eyes are in a photo.”

So, I always found that stuff really interesting. And then, over the course of my career, the technology that has enabled the type of stuff that we're doing today, with deep learning models that can do really amazing things with vision has just gradually matured, right? It's sort of went through these research stages and got into production, and we got to the point that we've got these big server-side models that can do really incredible things with vision and trying to understand real world images, for example, or processing other kinds of data and figuring out what's going on.

As that technology has grown and become easier to use, so has embedded. And so, embedded tooling used to be very, very difficult to work with, used to be super niche. And over time, the tooling has evolved to make it easier and easier for any engineer to work with. What that's allowed to happen is that the dots have sort of been connected between these two disparate worlds, of machine learning and embedded engineering.

So, there was this moment where I was working on a startup that I founded years ago, where we were basically developing technology for farming insects as a source of protein. I don't want to go into that, because I could spend the whole podcast talking about all that crazy stuff. But one of the things we had to do is understand. We've got these sensors in an insect farm that are telling us what's going on. And we need to be able to interpret this information to give us actual understanding of what's happening on the farm. We've got raw sensor data, and we need to transform that into something that we can use in some if statements so that we know, if this is happening, we need to wake up the team, and they need to come into the farm and fix this problem. Because otherwise, we're going to have a lot of unhappy insects.

Working on that stuff, it kind of brought together these two veins and allowed me to see that there's some potential there. And then when I was working at Google, I was just very fortunate like Google, one of the cool things about working there is there are all these different teams working on crazy things, and you can get exposed to stuff that you just happen to walk by in a corridor, and I came across the team

that was working on this union of machine learning and embedded and immediately realized, “Wow, this is my thing.” So, kind of like there.

[00:07:36] SF: Yeah, it's amazing. I relate to what you're saying about how I think historically, this idea of like building on embedded systems is something that's like procedure is very hard, is niche, is something that even going back to when I did my computer science education many years ago at this point, I wasn't really exposed to that. I remember I had a professor approached me about the idea of doing my master's degree working on embedded systems. I was like, “God, no, thank you.” But I think the industry has progressed a lot since then. There's been a lot of development that makes it I think, more accessible and seem more sort of tangible to actually creating things that people are actually going to utilize versus what I was experiencing back then that felt highly niche or very theoretical.

[00:08:21] DS: Yeah, it's interesting. It's always been really important, embedded engineering, but it's gradually gotten more accessible. I wouldn't call myself an embedded engineer. I'm not a real embedded engineer. I've been working with embedded systems on and off throughout my career, and I sort of know my way around. But the types of skill sets, that people who are spending every day in the trenches with embedded engineering have, are just mind blowing, really, really cool, fascinating stuff. But it's typically been a little bit more difficult for the average engineer who today, I guess, the average engineer is probably someone who works with web application development, I would say, or like back end for web applications. Now, we've got to a point where it kind of is a bridgeable gap.

But actually, what we do at Edge Impulse is more about enabling embedded engineers, of whom there are millions of people, enabling those people to access machine learning, which is an even more fringe thing than embedded engineering ever was, right? This is something that's come out of research labs in the last decade. The type of deep learning that is widespread in production use now is something that almost didn't exist a short while ago. So, it's even more of a tricky thing.

[00:09:41] SF: Yeah, I guess like why machine learning for these types of devices, these edge devices? What type of I guess use cases are people solving using AI on the edge device that they weren't able to do previously?

[00:09:56] DS: Yeah. So, there's a really nice way to think about this is that we've got this edge device. So, you know what is an edge device, first of all? It’s a device at the edge of the network. So, it's

something that's like a node out there in the world where things are happening. It could be an IoT device with some sensors on. It could be like a gateway device that's working with a few IoT devices to give them connectivity. In more tangible terms, it could be something in your car, it could be your smartwatch that you wear when you go running. It could be a weather sensor on a building. It could be a machine in a factory that's got some sensors hooked up that can monitor it.

So, all these devices are out there pretty pervasively in our world these days, that are kind of living this IoT dream where they're collecting data. And then they're looking at that data a little bit and sending it off to a server, maybe. We've got some insight into what's going on in our world and the big vision in IoT is that if you instrument everything, you can then make business decisions based on it, or you can build smarter products. But the problem is, the thing that wasn't really anticipated is the sheer volume of data that comes from the sensors that are out in the field.

Imagine you've got a single device, it might have multiple sensors on, each sensor is like capturing data at thousands of samples per second, all of that data is on a device, which is maybe running on a battery, or it has limited connectivity, and it's very difficult to send even remotely representative sample of that data upstream. So, you really have the situation where we've got all this potential, all this data that's describing what's going on in the world, but we can't really do anything with it, because it's trapped on these devices. And previously, all the devices could do, they don't have enough memory to store much of the data. If you've got a tiny little microcontroller with a couple of 100 kilobytes of RAM, if you're lucky, you're not going to be able to store a load of data and then do analysis on it.

So really, what's going on is we've got some maybe very, very simple logic that's doing like a moving average filter. It's like, “Ooh, if temperature gets above this, then send a message to the server.” And that's about all we can do. That's really a shame, because the data we have captures so much nuance about the world, but the utility of these little signals we're getting from is not that high. So, the thing that's really, really cool is, if you can take a machine learning model, and put that on one of these edge devices, then the machine learning model essentially, is a little bit of software, which has been trained to represent some human insight. So, it's like you've taken a little bit of human insight or human intelligence, and captured it in this program through some kind of method. And then you're able to deploy that program down onto one of these devices, and it can suddenly start to make sense of this data that's flowing in. It can turn us high frequency stream of sensor data into a series maybe of

meaningful events that tell you about what's happening in the system. And then you can send those events upstream and act on them. Or you can use them to initiate some kind of local action.

Maybe we've got an industrial machine that can detect when something's going wrong when something's deviating from a normal state, and it can shut itself down to stop any damage happening. Or maybe you've got a security system, which is able to not just detect if there's motion going on in a room. But if there's a person walking around between certain hours, and then it can wake someone up to go investigate, there's just so many different applications where if you put some intelligence down on the device, you get a load of things that are feasible, that weren't feasible before. And it covers everything from improved privacy where there are so many applications where you don't want to be sending raw data from the field up into some mysterious cloud environment. Home security is a good example. Like, I wouldn't want to have a camera in my room, in my house, watching me all the time and streaming the data up to some cloud company. But I'd be quite happy to have a little camera that's able to say whether a person is in the room or not. And if there's a person in the room while I'm on vacation, then maybe that's not good.

[00:14:37] SF: You mentioned a lot of really interesting use cases. I think in this idea of bringing intelligence to these edge devices. I think a lot of the like recent excitement in development that people had around machine learning and AI has primarily come from training these really big models with massive amounts of information data, but utilizing the scale of the cloud, and a lot of techniques like neural networks and so forth have existed for a long time. But it's like the scalable training that's made, what was impractical, now practical for performance perspective. With machine learning on edge devices, where, as you mentioned, they could be super low power and not have a lot of resources, you're essentially operating in a completely different world from these massive models running on the cloud. What were the technical innovations in the space that have made it so that companies can actually start to invest in this type of technology?

[00:15:28] DS: Yeah, really good question. So, there's a whole set of tools that have made it possible to actually get these models running on device. One of the kind of most important is just the runtime or the compiler that you use to take a model that you've trained somewhere, and get it to run on an embedded device. So, some of these tools like TensorFlow Lite and microcontrollers are basically allows you to take a model that is trained in a pretty typical way on cloud system or on a development workstation, and puts it into a form which is optimized to run on an embedded device. It has an

interpreter, which is nice and lightweight. It doesn't have lots of dependencies, like the typical Python libraries might. It's just nice and portable and simple and it can run these things on device.

But natively, it runs them kind of slowly. The thing with machine learning models is they have a lot of repetitive math like big matrix multiplications and those little microcontrollers are not necessarily particularly fast. I mean, saying that, one of the things that's led to this technology kind of bursting out right now is that over the last few years, we've got these really cheap microcontrollers, cost a couple of dollars and are more powerful than the computers that I used to play video games on when I was a kid. So now, that we've got this hardware that can potentially run these things, even then, it's a little bit slow, by itself. One of the things that's happened is that the companies that develop these pieces of silicon have identified that it's really important to be able to do this type of computation and they've come up with libraries that make use of optimizations on the hardware level, to implement some of the fundamental building blocks of the mathematics for deep learning.

So, we've got these tools that can run a model, those are then connected with these libraries that can make use of the special features of the hardware in order to run them quickly. And suddenly, you're able to do computer vision, with a deep learning model on a little tiny microcontroller that uses barely any power. So, there are also some optimizations and interesting approaches that you can use to transform a model, again, kind of mathematically so that it's more efficient to run. So, things like quantization, which is a way of basically just reducing the size of the numbers that you're dealing with, and allowing you to kind of do less work, but get the same result.

[00:18:10] SF: So, it sounds like there's these hardware optimizations that have been built in, and then it sounds like also that, a lot of the trainings are actually happening on the device. Are the devices primarily performing just inference operations?

[00:18:22] DS: Yeah, that's a very important point. So, what we've seen is that training on device is totally possible, it's just more math. There's no reason technically why you wouldn't be able to do it. But the limiting factor really is the availability of labeled data. So, when you're training a machine learning model, typically you're taking a dataset and you're marking different samples in that dataset with some metadata about what they represent and that metadata is called a label.

For example, the thing everybody always says is like, training a model to identify a cat versus a dog. That's kind of boring, actually. Let's talk about something embedded specific. Maybe we're training a model to identify whether I'm running or doing push-ups, and it's going to go in a smartwatch. So, I've got a dataset of lots of captures of maybe accelerometer data recorded from people who are doing push-ups or are running. What I need to do in order to train a model is label that data with the activity that the person was performing. So, that's very easy for me to do. Well, it's not always easy, but that's something I can do if I've got this dataset, I can sit down, ask the people what their activity was, and label it all. But if we're on device, there's no sort of third party there who's applying the labels. Sometimes you can figure out what's going on from situational cues, but if you can do that already, you probably don't need another machine learning algorithm to help you understand what's going on, right? So, you It's actually quite hard to get data and know what the data means when we're out in the field. And so that means that training in the field is pretty challenging, because you're just lacking stuff to train with. And while there are some ways around this, and there are definitely some situations where you can infer what's going on from some other cue, the vast majority of the time, we're talking about machine learning in an embedded context. We're talking about doing inference on the embedded device, but we're probably going to be training the model somewhere else, and that's going to change and evolve over time. I'm sure we'll see more training on device as the tooling improves, and people figure out what kind of use cases it works in. But for now, the vast majority of the time, we're talking about inference on device.

[00:20:49] SF: You mentioned that there's these techniques, for example, like reducing the size of the inputs. So, is the process of like building the inputs for the machine learning, like building the model, different? Is that feature engineering process different for using things like real time signals versus, traditional ways of building machine learning models?

[00:21:15] DS: Yeah, I'm glad you brought that up, because that's one of the things that makes embedded machine learning really interesting, really unique, and it's sort of at least opened my mind up to this whole universe that I hadn't previously been that aware of, of digital signal processing. So, people have been trying to interpret real time streaming data on small electronic devices for a long time, and the set of tools that is used to do that, for example, to take an audio signal and extract some information from it is called digital signals processing. It's this amazing field that's been around for a long time. There are a lot of really, really cool algorithms that are like fascinating to read about and understand and play with, that can help take this really kind of big, high frequency messy data, and

extract the pertinent features from it, in order to make decisions. Historically, this has all been done just so that you can write some sort of heuristic logic or conditional logic to understand what's going on.

So, maybe you've got like a clap detector where you can go, “clap, clap”, and the light switches on and off. You could use digital signal processing to kind of – if you figure out like, what frequency is that clap, and you look for audio that's in that frequency only and filter the rest out, then you can use that to gate whether the light turns on and off. So, all of this same technology, we have available now to feed into machine learning models. What we can do is rather than like feed raw audio into a machine learning model, which is kind of problematic in a lot of ways, because raw audio, it's very sort of big. You've got like a high frequency you need in order to capture all of the audible frequencies. You got to be capturing audio samples at a certain frequency. And you end up with a lot of data. If you feed a lot of data into a deep learning model, you need a big deep learning model, typically. The model itself needs to be quite large, or at least, you need to do a fair amount of computation in order to process these samples as they come in.

So, what we're able to do instead of that, is throw a battle tested, like highly researched digital signal processing algorithm in front of the deep learning network, and let that process the audio, and out of that, we get a kind of like distilled, simmered down version of the audio with only the information that we care about. So, if we're dealing with speech, maybe we can tune our signal processing algorithm to only pick up frequencies that are in human voices, because we don't need any of the rest of that information. And we can also sort of reduce the resolution and time of that data, so that it's only enough resolution to recognize the types of sounds we make during speech. We don't need to go any more fine grained than that, because we're dealing with speech. Suddenly you've taken this big huge input and boiled it down into something much more manageable, that we can feed into a deep learning model, and it makes the models life easier, because it doesn't have to figure out that stuff on its own. They can, right? Deep learning models are really good at that. A deep learning model, you can think of as a universal function approximator, so it can learn in theory to take any inputs and give any output. But it's better if you can do that stuff in a really easy, well-known way using kind of battle tested engineering tools, and then give you a model less to worry about. That way you have more control as well. Because I can say, “Hey, I want to ignore signals in this frequency, maybe there's a lot of noise going on in a certain frequency.”

At the DSP stage, I can just decide I'm going to throw that frequency information away before it even reaches the model, and it's just saving time and allowing me to have a bit more decision-making control over what's going on.

[00:25:28] SF: We talked about a lot of this, like optimization work that people kind of have to be thinking about, and I imagine it's a little bit like, you know, the constraints of doing game programming in the 1980s versus what people have at their fingertips today. So, is there like new skills that people have to learn that coming from, I imagine, embedded systems programmers already have some of these like optimization tricks and thought processes at their disposal, but coming from maybe traditional machine learning world, do they need to kind of get up to speed on some of this different way of thinking versus like, "Hey, I can just scale endlessly, horizontally and vertically within the cloud and build whatever I want?"

[00:26:10] DS: Yeah, definitely. I think one of the key things for me is this idea that you need to decide what does good enough mean, because in the machine learning research world, and in the production world, really, if we're honest, there's just this kind of hardcore pursuit of metrics and performance above everything else. And this idea that what we need to do is eke out every last fraction of a percentage point of performance against a benchmark dataset to prove that our model works well. The thing is, the way that we do that typically is by pouring processing power and memory at the problem. So, we're now, we've got these models that we're using for language processing, which are vast, vast things that can't even run on a single computer, right? They can only run in a distributed setup with like a ton of GPUs. That isn't going to fly when we're talking about little tiny, embedded devices.

What we've got to do instead of thinking about, like, how can I get the absolute best accuracy at doing this, it's still important to think about accuracy and get the best bang for your buck, with performance. But we also need to think about, if I'm designing this application, what's the minimum signal that I could get from this system that will help this application be better? So, for example, imagine we've got a machine in a factory that we're wanting to understand when it's about to fail, and if we can catch a failure before it happens, that's awesome. We're going to save a bunch of money because some parts of it won't get destroyed when it fails. Do we need to catch the failure 100% of the time? Maybe not, because even if we catch the failure, 20% of the time, we're still saving money. So even if we can build something that doesn't give perfect results, but is good enough to move the needle a bit, it can be a really valid application for this technology.

Thinking in that type of mindset of like, “Okay, I'm building this product”, if I just knew a little bit more about one little thing, what could I do? Sometimes the answer is, “Wow, you can actually make that work so much better, even with that little bit of information.” Maybe you're not throwing everything at the machine learning model. Maybe you've got a perfectly good heuristic algorithm that you've written using your domain expertise, that expresses, “Hey, if this happens, we'll do this. If that happens, we'll do that.” But maybe there's like a kind of few corner cases that that doesn't catch, and maybe you've got this 20% of corner cases, that might be the thing that the machine learning model can help with. And by training a machine learning model on those particular corner cases, it can give you a little bit more insight that allows you to capture those and do a good job of resolving them.

So, it's all about thinking like, how can this add to what I'm doing and how can I create signals for myself that are beneficial without necessarily thinking, this is like a magic wand, I'm going to wave and it will do everything for me and figure out what's going on.

[00:29:24] SF: Right. I imagine a lot of use case dependent, going back to your example earlier, where you talked about classifying push-ups versus, I forget about the example is, sit-ups, let’s setups and then. If my whoop misclassifies my sit-ups as push-ups one time out of 10, it's not that big a deal. I'll be okay. There's the George Box quote of like all models are wrong, but some models are useful and it's probably as applicable to this idea of AI on edge devices as anything, really, in the space.

[00:29:56] DS: Exactly, yeah. And I think this philosophy obviously, applies across machine learning. It's not just something that's specific to edge devices. But it's really important on edge devices, because often, you're constrained by maybe how much memory you've got, or the latency you can put up with. So, for example, for vision, typically, if you're doing something with vision using a deep learning model, the bigger your model is, the more accurate it's going to be, up to a certain point where you get diminishing returns. But the point of diminishing returns is a lot bigger than the size of memory that's available on a typical embedded device.

So, at a certain point, you've got to decide, “Hey, I've got a cut off here, I'm only able to use like 100 kilobytes of memory. I still need some space on this device to fit the rest of my program.” So, what can you do with 100 kilobytes? What's the best you can do? Maybe you can only get 60% accuracy on your task versus 80% with a bigger model. That can't be the end of the world, if that happens. It's got to be,

“Hey, this is still an ingredient that can help make my application better, even if it doesn't solve all my problems.”

[00:31:14] SF: So, I want to start to talk a little bit about Edge Impulse. Back when I was doing machine learning projects in university, you had to know a lot about how all this works, like data cleaning, which algorithms they use, how to read the output. And the tooling was super rudimentary, I don't know if you've ever used Weka, but they were obviously built by an academic with no intention of commercial use. So, I guess, like, how has this changed? And what does Edge Impulse kind of – or where's it fit into this like tool chain?

[00:31:43] DS: Yeah, I love that you use that idea of this stuff was built by researchers and academics and never intended for production use, because that's really the thing that drove the founding of Edge Impulse was like a bunch of people trying to build cool stuff that is all possible in theory, and running into obstacles, because the tooling is just so difficult to work with. And it's not the fault of the people who made the tools. It's just that those tools were designed to prove something in a research context, they weren't designed for real world use.

So, what we do at Edge Impulse is come at this from the point of view of like, “Hey, if you're an embedded engineer, and you're trying to add a product feature that does something really cool, and you think machine learning might be able to help with that, how do we give you the tools to prove out and figure out whether machine learning can help, and then go all the way through the process end to end, and do a really, really great job of training the best model in combination with signal processing, that you can testing it out. So, you fully understand as an engineer, how it works and what are the parameters within which it does a good job. And then how do you get that integrated into your application that's going to be running on an embedded device?

So, we really tried to look at this through the lens of like someone who's an embedded expert, someone who works in embedded development, but isn't necessarily a machine learning expert. Although we want to be really useful for people who have machine learning expertise as well. I think of Edge Impulse as an end to end platform, and there are a few competitors out there as well. It's a space with a lot of innovation. But it's really a platform that's designed to help you throughout this entire thing that we call the machine learning workflow, that starts from when you're thinking about what you want to build, and

maybe knocking together some simple prototypes, and goes with you all the way through the process of training models, deploying them to device and then making sure that they work well.

[00:33:56] SF: Can you walk me through the process of how do I build essentially like the to-do equivalent of ML on an edge device using Edge Impulse?

[00:34:07] DS: Yeah. Well, we got this really nice feature that basically, because this is such a common thing that someone who just wants to give this technology a spin. So, one of the things you can do with Edge Impulse is like treat your phone as the edge device. If anyone's familiar with WebAssembly, we figured out this way to use WebAssembly to deploy a model to your phone, super quick, just so that you can test it out and see how it works. So, the most basic end to end thing might be, imagine I want to do this push-up versus sit-up detector, right? So, I can hold my phone in my hand. Actually, to begin with Edge Impulse, it's like a set of tools based around a web UI called Edge Impulse Studio, which is kind of like a workbench for dealing with data from sensors and training models and deploying them.

So, it also has a bunch of other different parts that can connect to different devices and do different things like collect data. One of these parts is this mobile client that I'm talking about. What you might do is set up a project in Edge Impulse, you then connect it to your mobile phone, you just have to scan a QR code, and then you can capture some data super quick. So, lie down on your mat, and do some sit-ups with your phone in your hand, and then do some push-ups, I guess not with your phone in your hand, because it wouldn't move around very much. We've done stuff like strapping an embedded development boards, to our arm with some bandages, and doing a workout and collecting a dataset based on that.

Once you've got this little dataset, you really only need like a couple of minutes worth of data to begin, if you're doing something with activity classification. You can train a simple model and we kind of worked through this idea of sensible default values. So, if you first arrive at Edge Impulse, you've got this data that you've collected, if you just kind of click through, we'll hopefully have come up with some stuff, which makes sense that will give you a model that works. It might not work super well, but it will work enough that you'll be like, "Wow, that's crazy." And then you start with that, and you go from there. Because machine learning development is the most iterative thing you can possibly imagine. The whole way it works is iterative. The technology itself works through iteration, like the model is trained through iterations of trying to get better and better at modeling the training dataset. And then the workflow is

iterative as well. You're figuring out, “Hey, I tried this DSP algorithm, it works all right. Let me try a different one and see what the difference is. Like is this working any better or has it got worse?” So, what we allow you to do with Edge Impulse is iterate really, really fast, so you're able to test things out super quickly, you're able to make changes and try them and keep track of the changes you've made. And over time, hopefully not even that long, get to something that works really nicely.

[00:37:07] SF: Imagine some of the examples that you gave, like the idea of having a device in a factory that can detect some sort of major problem within the factory or, or even like the idea of some sort of device sitting on the International Space Station that's doing something intelligent, that helps keep astronauts alive or something. How do you bring in, I guess, that domain expertise? Whether it's farming, a factory, or space travel and translate that expertise, I guess, into like an ML model that works for something like an edge device?

[00:37:38] DS: I'm so glad you asked that question. It’s so important. So, there are two sides of development with machine learning, right? One side is the technical part, where you're thinking about, how am I going to build this? What are all the technologies I need to bring in in order to make this happen? But the other part is thinking about your application, thinking about what it actually needs to do, and understanding whether it is succeeding or failing at that. The only way you can do that is with domain expertise. The only way – imagine we're building something to help the International Space Station stay in all of it, we're going to have to have somebody on board who understands that problem that we're trying to solve and understands whether we're doing it right or not. Because otherwise, I think everyone who works in technology has this little bit of a kind of Messiah Complex where it's like, “Oh, if I just came in and whipped out my laptop, I could solve this problem in 10 minutes.”

Well, that isn't true. And where you really realize it isn't true is when you're getting into trying to solve problems in industry that you don't understand, and you think you've got the right idea, but you don't. So, what you need to start with in any project, using machine learning is someone who really knows what they're talking about, and that person is your domain expert. And you're going to need to work with them to shape first of all, the design part of your projects where you're figuring out like, what is this thing going to do? Before you even factor in the technology, what do I actually want to build as an application? If I was going to build something, and the technology would just solve itself, what would I build in order to improve this situation?

And then figuring out what are the goals from there? And what are the kinds of non-goals? What are the things we need to make sure happens? And what are the things that we have to avoid at all costs? So, working through that design process with domain experts is absolutely critical, and even beyond just one domain expert, it's really important to have sort of an advisory board, almost, for your projects where you've got a pool of expertise that you can draw from to identify things you might not have realized. So, I guess a question you might ask is, why is this so important in machine learning, and we don't necessarily do this with other types of thing. And the thing is, first of all, we probably should do this with other types of thing.

But secondly, machine learning basically automates the creation of software to some degree. We take data, and we run it through this process and a program comes out, which makes decisions. And that doesn't necessarily, as an engineer, have to be at any point within that process where you understand what's going on and that's really, really dangerous. If we're sitting down writing an algorithm to make a decision, then we have to know what we're doing, right? You have to know what the algorithm is doing. We have to understand what are the specific inputs and outputs that should map up, and we can even write some unit tests to prove that it works, to prove that we wrote the code correctly. So, it's a little bit of a self-limiting thing. Whereas with machine learning, it's easy for anyone to go in and hoover up some data, and then train this model and not really know what it's doing, and not really know why it's making those decisions. If somebody comes and asks, you've got no answer for them. So, that's why it's so important to do a good job of this. And that the main interface beyond the design and planning stages that involve domain expertise, the main interface that exists between domain expertise, and your system is the dataset.

The dataset is where we capture the insights of a human expert in the form of a dataset, and then we use that to train a model. So, it's absolutely critical that you do a good job of capturing that insight and understanding the boundaries where beyond which that insight doesn't apply. And making sure that you're capturing a dataset that's like representative of real-world conditions, and is diverse enough to capture all of the kinds of crazy things that you can see in reality, and that's really where you need the guidance of a domain expert, because how do you know what to capture if you're not an expert? You just don't.

So, what we've done, hopefully, is build some tools that embedded engineers work with domain experts on solving problems using machine learning. And the cool thing about embedded engineers is a lot of

the time they have really strong domain expertise about areas where they're solving problems. So, it's quite a cool combination of roles.

[00:42:29] SF: Yeah, it's amazing. I think it's like a really important thing to bring together is that domain expertise, whatever you're trying to build for, with the expertise of the embedded engineer with expertise, essentially, in machine learning as well. Because there's been a number of high-profile machine learning models that have resulted in things that are very biased, because the inputs weren't particularly weren't good, they weren't designed with this diversity in mind, and you end up with something that's not going to really serve the larger population that's trying to use it.

So, as we start to wrap up, I did want to touch on some of the evolution of the engineering team at Edge Impulse. So, you've been there as a founding engineer, and now you're head of machine learning, how has like building a product like Edge Impulse, how's it different than say something like a traditional web based, SaaS based engineering organization?

[00:43:21] DS: Yeah. I mean, it's an interesting question. So, at our heart, we have SaaS products. We're building this set of tools that exist online that let you train models and deploy them to embedded devices, amongst many other things. But the real difference that we have is that we've got these two extra pieces of like domain knowledge, basically. One of them is on the embedded side. We have a really big, really strong team of hardcore, embedded and DSP engineers, who understand what it's like to be in the trenches, building embedded applications, and understand the fine nuance of designing signal processing pipelines for making sense of digital signals. And then on the other hand, we have machine learning team who's able to bring insights in, like, how do you train models, not just deep learning models, but classical ML models? And also, along with that, like, how do you interpret the information about these models? How do you understand whether a model is performing well or not? And how do you understand whether your application is effective? What are some of the problems you can run into that might not be intuitively obvious, which, that's the big risk with machine learning that you can run into problems and not even understand that you've got one or understand why it's happened.

So, what we're trying to do is bring together all of these kinds of disciplines under one roof, and have really good interplay between the teams. The thing we're even talking about this afternoon, one of the things we've noticed that the company is like everyone on the machine learning team wants to learn

more embedded stuff. Everyone on the embedded team wants to learn more ML stuff. So, we've got a lot of people who are very curious about the world and about their different engineering disciplines. And we're trying to do our best cross pollinating, because we're sitting in this space where it's like a new intersection of things that haven't touched each other before. And there's so much cool stuff that we can do there. By building this, basically, like diverse organization of people with different backgrounds, we're able to identify what are the promising things that we can do. We found that it's just been incredible. We've been able to come up with ideas and build features around them that we've not seen anyone do ever in the world. We just come up with this stuff internally, and built it and it works really well, because we're in this unknown area where new things are touching. For me, that's the most exciting place you could hope to be at.

[00:45:56] SF: Yeah, that's incredible. I think it's a really good sign of the company's culture that the embedded engineers are wanting to learn more about machine learning, and machine learning engineers are wanting to know more about embedded, because it's a team sport to make a company successful. Is there anything else you would like to share with the audience before we wrap up?

[00:46:16] DS: Yeah, I mean, I'd love to talk a bit about the book. So, I guess one thing I'd like to share is that we've got a new book coming out. My coauthor, Jenny Plunkett, and I are currently going crazy getting past the finish line on getting the book ready for publishing. The name of the book is *AI at the Edge*. It's really sort of designed to be this introduction to the whole field of artificial intelligence on edge devices, for pretty much anyone with a technical background. So, it doesn't matter whether you're an embedded engineer, or an ML person, or a PM, or an engineering manager, or someone who's interested in founding a company, or even someone with domain expertise in a completely unrelated topic that they think might be interesting to apply ML to.

The goal is that it gives you sort of this A to Z lay of the land. So, you understand what all of the moving parts of this world are, and you can start to make progress on whatever project you want to build. It goes through everything from understanding all the different technologies, understanding what are the different embedded devices you can use, what are the different algorithms you can use, through the whole workflow that we suggest that helps you with a sort of eye towards ML fairness, and building applications that work really well, guides you through the process of training models, testing them, deploying them. And we even have a section in there about how do you build a team to work on this stuff, right? It's such a new field. How do you assemble a team of people who can execute in this

space? And we're drawing on not just our insight from being around at Edge Impulse, and Google and so on, but we're also looking at all of these people that we know in our networks, and what kind of experiences have they had with this technology, and we tried to draw this really broad kind of summation of all of this hard-earned knowledge that exists so far about the field.

I'm pretty excited about it. It's going through the review process at the moment and even the feedback we're getting from reviewers is just adding so much to the kind of the hard-won insights that are in there. So, it's been this amazing team effort by a lot of different people, but very excited to get that out there soon.

[00:48:43] SF: Yeah, it's really exciting and I've already pre ordered my copy. So, I'm excited about that.

[00:48:46] DS: Oh, thank you.

[00:48:48] SF: Yeah. So, thanks so much for coming on the show and sharing the background on machine learning for edge devices. I think we got into a lot of different topics on this, and it's always fun to talk to you. I always learned something new. You make me feel a little bit bad about myself because you have way cooler hair and a better accent. But I want to thank you so much for coming on, and we'll make sure to include a link to the book in the show notes as well.

[00:49:11] DS: Oh, thank you so much, Sean. It's always fantastic to catch up with you as well. Yeah, thank you so much for having me on and thanks to all the listeners. If anybody's interested and wants to chat about this stuff, reach out. I'm on Twitter and always happy to chat.

[00:49:19] SF: Awesome. Thanks.

[00:49:20] DS: Cheers.

[END]