EPISODE 1076

[INTRODUCTION]

[00:00:00] JM: Devices on the edge are becoming more useful with improvements in the machine learning ecosystem. TensorFlow Lite allows machine learning models to run on microcontrollers and other devices that only have kilobytes of memory. Microcontrollers are very low-cost, tiny computational devices. They're cheap and they're everywhere. The low-energy embedded systems community and the machine learning community have come together with a collaborative effort called TinyML. TinyML represents the improvements of microcontrollers, lighter weight frameworks and better deployment mechanisms, as well as greater power efficiency.

Zach Shelby is the CEO of EdgeImpulse, a company that makes a platform called Edge Impulse Studio. Edge Impulse Studio provides a UI for data collection, training and device management. As someone creating a platform for edge machine learning usability, Zach was a great person to talk to about the state of edge machine learning and his work building a company in the space.

We are always looking for new show ideas. If you have a show idea, you can go to softwaredaily.com and create a quick posting about that show idea, whether it's a project that you're working on or the company that you work at. We are always looking for new stories to cover.

[SPONSOR MESSAGE]

[00:01:30] JM: If you have a DevOps story to tell, GitLab would like to hear it virtually. The 2020 GitLab Commit User Conference will be held as a virtual event on Wednesday, August 26th, and the call for proposals, or CFP, window is open through June 5th. Maybe you've managed a culture change, or discovered a DevOps solution that you want to share, or you've dramatically reduced your release time. Whatever your story is, you can share it at Commit. Go to softwareengineeringdaily.com/commitcfp, and you can also register for the virtual GitLab Commit user conference at softwareengineeringdaily.com/gitlab commit. That's softwareengineeringdaily.com/gitlabcommit to register for the event and softwareengineeringdaily.com/commitcfp to submit your call for proposal submission through June 5th. The CFP will close on June 5th. So if you have an idea, share it before then.

[INTERVIEW]

[00:02:44] JM: Zach Shelby, welcome to the show.

[00:02:45] ZS: Great, Jeff. Really glad to be here. Exciting times in the industry especially with technology that enables remote things to happen.

[00:02:54] JM: Sure is. You work on Edge Impulse, which is a platform for machine learning, embedded device machine learning. Explain what Edge Impulse is.

[00:03:06] ZS: We started working on machine learning on really embedded compute about three years ago while I was at Arm. At Arm, we started to see this trend where Morse Law had caught up with edge compute, and what that means is that we started to pack more and more small transistors into these little microcontroller and lower power CPU-based devices. All the devices you don't see in the world, electric meters, industrial sensors, wearable health monitoring devices, and what that means as Morse Law caught up is that we can pack a lot more math compute power per watt per dollar into devices. Suddenly, it became really efficient to run extremely complex math in real-time on any of these devices, whereas radio and moving data to the cloud still costs, right? There is a cost of energy and there's a cost of all these data sitting in the cloud.

We identified this opportunity to start to really go and maximize the amount of processing we're doing at the edge. We started working on some cool open source projects, this project called micro-tensor, that tries to map TensorFlow models and TensorFlow ML flows down to microcontroller devices. We've built a nice following with that and ended up merging projects with TensorFlow. Micro, as it's known today, the TensorFlow Lite Micro project. We got some great experience building the underlying tooling to make machine learning possible on these devices.

But what we learned was really important, that just providing some command line tooling is like making a compiler work for software. It's a very small piece of the whole development flow to make software developers successful. We realized that, actually, for the big problem for making developers successful with machine learning on these devices, not just the compiler, it's the whole flow of data, data sets, models, training and testing models and then deploying models. That whole flow of DevOps is a really big problem. It's even bigger when you get into edge devices. With my cofounder, Jan, we decided to build Edge Impulse to solve that, that lifecycle problem for developers working on any kind of edge device.

[00:05:36] JM: The lifecycle of a developer who is building machine learning models for edge devices.

[00:05:43] ZS: That's right, and it works with real-time device data streams, right? The edge of our infrastructure produces a huge amount of data. Sensors, audio, images and video, all the way to like device logs, network logs, security logs, SCADA systems and industrial. All these systems produce very large amounts of real-time data streams, and machine learning allows you to work on those data streams like in stream right where they happen or very close to where they happen. So you could deploy machine learning right on the device that has the sensor or the data stream so it never has to go anywhere before you know something more useful out of it, or you could put it in networks, so in a router, or in a gateway, maybe an edge server. Edge, is a telco network. To process that stream in real-time and reduce the amount of information you have from raw data to answer. Instead of pushing just like big audio streams, you could say this audio showing there's a broken motor in this factory, and you just send that one bit of broken motor rather than complex audio streams.

[00:06:58] JM: Tell me about the way that devices, embedded devices are managed and indexed by a company. If I've got a bunch of devices throughout my factory, how am I even accessing and managing these devices much less doing continuous deployment or machine learning model training and deployment and testing across these hardware devices?

[00:07:29] ZS: That's a good question. A long time ago, 20, 25 years ago, all of these systems were very proprietary. You had proprietary control busses and very specialized kind of software tools per vendor. It used to be that Honeywell had to be involved with maintaining any device in like an industrial setting. But the internet of things changed that. We spent the last 20 years upgrading all of our infrastructure for devices so that they speak IP. They're all on IP stacks these days. So you have an IP address for the device. You have a secure connection, HTTPS style, TLS encrypted connection, and then you have cloud services that manage communications with these devices. We have a lot of IoT systems for all the major cloud vendors from Arm, which allow you to speak to any of these devices and perform firmware updates. All of these devices these days are firmware update compatible [inaudible 00:08:29].

You can push in your firmware down a device easily, and that's the really embedded end of the spectrum. When you get into CPU-based devices, like routers and gateways, then we have you in more sophisticated management, because they are embedded in Linux machines. All the same things we do to manage like an edge server that runs Linux, we can do the same things with an embedded Linux router.

We can push new software down to the device. We can manage it, and that enables us to deploy machine learning just like we would any other software package. Think of machine learning experience just as a software library. You push in data in, you get answer out.

[00:09:11] JM: The edge devices, what's the compute power for these small devices? Do you have enough processor power and memory to run the kinds of workloads that you might want to with big machine learning jobs?

[00:09:34] ZS: That's the amazing thing that's happened with the technology. There' two big trends. There's a whole new field of machine learning math called TinyML, and the TinyML movement was started by Google, Arm, Qualcom, Microsoft, all the big players coming together to make use of the best machine learning math optimization. So, optimized architectures, very small neural networks, optimal signal processing, quantization. There're a lot of tools to make machine learning very small integration. That's one thing that happened. That started last spring, and there's a TinyML foundation that we help support to spread the word on that.

But also, 32-bit MCU in the CPUs got a lot better, very, very fast. Arm and all the Silicon vendors around the Arm architecture are pushing new computer parts every year and they're getting faster and faster. For example, on a typical 32-bit microcontroller today, we can push 40 million math operations a second. That's on a $2 part. The amount of math that can be crunched on these devices in 32-bit floating-point, right? This was real math and the kind of workloads that we need for machine learning. We can do a lot in software.

I'll give you a few really practical examples. Right from our tool chain, we can do complex vibration analysis. So you take three axis accelerometers. That's a motion unit. You can find in your phone or in most wearable devices, and in real-time we can go processes those three axes that are pretty high-frequency vibration data. That entire machine learning processing cycle takes 1 millisecond of math time for two seconds of data. The amount of RAM and flash used is under 4K. That's a really small algorithm. That's kind of the extreme end of efficiency that we get to. No special acceleration or anything. That's just software, C++ running on the MCU.

Now if we go higher into the kind of application spaces, we get into audio and things that look like audio, like radar can look a lot like audio, like a LiDAR scanner for example. Hence, processing audio, we push one second of audio in and we do in an MFCC spectrogram and then we start applying machine learning to the spectrogram. That takes 14 milliseconds of inference time for one second of audio. Extremely real-time, it's very operation, and it's about 10K of RAM and flash. That's today's numbers out of our system. Those are very kind of MCU real-time applications possible today.

When you start to get into images and still images, this gets slightly bigger, but that's still within the scope of microcontrollers. Processing a still image for anomaly detection or classification can take some hundreds of milliseconds, up to a second, depending on how complex it is, how high of a resolution image it is, then you're talking about under 100 K of flash and RAM for optimized models. That's still within the scope of these like really expensive microcontrollers.

Where it starts to be a little bit too processor intensive for the MCUs is when you get into real-time data, because you have many frames per second that you have to process, and those loads, we push on to CPUs. Eembedded Linux routers, for example, CPU-based edge devices. We can do real-time video processing. Tracking objects across video or real-time tracking, or classification of images in video. You can do that on inexpensive CPU-based devices. Cortex A, embedded Linux routers today, without any acceleration.

But what you’re going to see happen is that we have a whole roadmap of interesting neural network acceleration hardware parts coming. Arms announced a new Arm ethos neural network acceleration unit that Silicon vendors will start to include in their off-the-shelf processors. We have vector extensions that Arm announced recently to be able to do more complex vector math at the hardware level. Then we have a whole bunch of other stuff, like today we announced DSP acceleration support without EDA computer, which is a really small low-power processor. So we can offload a lot of the map on a DSP. Instead of calculating a spectrogram in software, we can just say, “Hey, DSP. Calculate the spectrogram for me and then come back to me when you're done with it.” That helps us go a lot more efficiently.

As we see this acceleration hit the market, we’ll be able to push, for example, video down to an MCU, and we’ll be able to push images down to an even smaller, lower power part. So it just lets us do more machine learning with less compute.

[00:14:40] JM: What about TinyML specifically? You'd discuss TinyML as this project that is shared by multiple different large companies with the goal of making machine learning on small devices more accessible or more workable. Tell me about TinyML. What does it actually encompass?

[00:15:04] ZS: Yeah. We started TinyML really is a movement, an industry movement that’s shared by everyone last spring. That was started by Pete Warden from the Google TensorFlow team and a bunch of colleagues across the industry. What TinyML is, is it’s really a neutral movement, an information movement, and it's a foundation. What we formed was a nonprofit foundation to help spread the word about the technology. In general, TinyML is any machine learning that's targeting kind of sub-milliwatt low power devices, right? Let's enable machine learning workloads at low power on cheap hardware, and there is not just large companies, there are a lot of startups. We’re a member of TinyML Foundation. We help support and sponsor the foundation as a startup, and then there's a lot of research groups involved. A lot of the research communities involved, sharing their knowledge on the latest in machine learning technologies. Silicon vendors talking about the newest acceleration, and users even have started to join. Then what we do is we do talks every two weeks. There’s a TinyML talk on technology every two weeks all around the year. Then we have a summit every spring where we get together in the whole community. I'd say it's about 2000 people right now globally that are kind of members of the meet-ups and come to [inaudible 00:16:28].

[SPONSOR MESSAGE]

[00:16:37] JM: Triplebyte is a platform for finding a great job faster. Triplebyte works with more than 400 tech companies, including Coinbase, Zooks, Dropbox and Facebook. Triplebyte is focused on matching high-quality engineers with great jobs. We know that the economic downturn has caused some significant upheaval for current computer science students and new grads and many students don't have the internships or the engineering roles that they'd expected, and you can learn more about these opportunities by going to triplebyte.com/sedaily to sign up for the webinar that Triplebyte is hosting May 28th at 5 PM. You can go to that webinar to find out more about how to prepare yourself for a job, prepare yourself for an interview. Triplebyte.com/sedaily, and during that webinar, Triplebyte will interview Tito Carriero, a current chief product development officer at Segment, as well as former early engineers of both Facebook and Dropbox, and you'll get to learn more about how to get in the door at these opportunities for internships and jobs. You can also go to Triplebyte to find resources on engineering assessments, mentorship and other materials. You can sign up for all these at triplebyte.com/sedaily.

Thank you to Triplebyte for being a sponsor.

[INTERVIEW CONTINUED]

[00:18:10] JM: The machine learning models that are getting deployed to these devices, you said you wanted to assist with the deployment process, that lifecycle. What are the pain points in that lifecycle and what are you focused on fixing with Edge Impulse?

[00:18:32] ZS: Yeah. It's interesting. I'm a software guy and a communications Internet expert. I came at this with kind of my embedded software engineering backgrounds, and same with my cofounder Jan. We thought, "Well, what are the things we have to do in software, in DevOps, to really be successful software deployments at scale targeting these edge devices?" So we started looking at, “Well, what's the equivalent in machine learning?” There is a bunch of pain points that are really, really hard to deal with once you go past a single model.

If you're building a single model with a single kind of static dataset that's been already cleaned up for you, you can kind of hand-build this stuff, like we used to hand-build software in kind of the 1990s. You’d have everything on your own machine, in text files, you’d use VI to edit your files and then you had kind of a manually-maintained make file, and that was it on your machine. If you ever move this to another machine, it would probably fail to build.

ML is kind of at that state with data scientists. You can’t build everything on your own machine with your own Python installation, and you have your dataset there in a he file or in Python in your notebook and you build this thing and it works once on your machine. That's kind of the state of machine learning development flows today. It's very manual. It’s very kind of one developer, one machine.

The biggest pain points that we start to see is we’re enabling developers to do this much more broadly, is dataset collection to begin with. How the heck do you collect sensor data into a dataset and make sure that dataset is useful for use in machine learning? That's a really hard problem, because the data that we collect is different. In the Internet of things, we try to reduce the amount of data. That means we filter and we threshold everything.

When you filter and threshold, that data is no longer really useful for a machine learning. What we need with machine learning is very high-quality raw data and as much data as possible, and we need to associate labels with data. You actually need to know when this data sample of one second happened, Zach was running? These accelerometer values are me running, or an audio in nature conservation. We have projects going on in bird sound detection, animal detection. This sounds sample, at this time, that's a bird of this type singing, or this is an animal, an elephant making the sound, and we want to detect that. You always have to have some kind of a label associated with it, and that has to be part of data package.

Then there are things around secure data traceability. I want to be sure, when I push the data sample, that it’s securely signed, and I know the version number of the hardware and the firmware that was used to collect that data. All that has to be in a form that you can’t change it later, because if you start manipulating data later, you don't know really where it came from anymore. It could be a risk to your model work.

This data collection right from the source of the data and then organizing that in a way that it's really useful, it and can be maintained. That's a big pain point. We spent a lot of time producing really nice to use tools for developers just to get access to their own data, and it becomes valuable, right? As you build up your datasets as a developer or as a company, that's a super valuable asset for you. We're encouraging people to get really good at that.

Then as data gets really big, as you collect lots of these datasets, like we have one customer right now working on some wearable health data, like one measurement campaign was 3 terabytes in data. This is just sensor data. This isn't a big data. Sensor data, one measurement campaign, 3 terabytes. We do a lot of then cloud integrations for that kind of stuff or we directly integrate with buckets and we do queries on those buckets to go narrow down the data you want for a specific ML algorithm, because you just can't work with 3 terabytes at the same time. It's just too much. A lot of tooling around datasets.

But once you have a dataset, you also need to be able to apply that dataset to algorithms. The neat thing about machine learning is that these algorithms are like – Think of them as like templates. They're like software templates. They don't do anything on their own, but they're an architecture for data. Once you push your training data through these templates of algorithms, they become trained working machine learning. They're like the templates that learned what to do for this type of data.

They specialize themselves in the data that you're pushing at them. So you need a whole architecture for hosting those algorithms, right? Which algorithms do you want to choose for this use case you're working on? Say, animal sound detection and acoustics. You need to window that data. You need to extract your spectrograms. You might filter out certain frequencies that don't mean anything for you. Then you have to start matching the spectrograms for different data signatures. Well, how many classes of data do you want to look for? Do you want to just identify one bird or three or four different bird sounds at the same time? That affects the complexity of the algorithms.

Hosting these algorithms, making it more like a plug-and-play GitHub repositories, way to think of it, is something then we had to spend a lot of time on. Finding ways to make his extensible, because we can provide great templates, but you might be working on a very special kind of project where you need to customize something. Well, we need to make that easily customizable. We need to host those algorithms for you.

Making algorithms accessible for people and making them easily trainable with your dataset, we spent a lot of time on, because that's hard. Finally, testing, right? Testing these algorithms is really important in the cloud, and we do all these things in the cloud for the developer, because there's so much compute and so much kind of complex tooling involved. It's really hard to install on a single machine and it's hard to access cost with many people. We do this kind of visual testing in the cloud.

We stream sensor data to the cloud. We test how does it work? How well does it work and why does it work or not work for the developer? You really have to see what's happening, because these models are fairly complex. If you put this algorithm immediately to an edge device, it's almost impossible to see what's going wrong. It's a little box without a screen with an audio mic in it. You can't really see why it's not working, and it's very hard to debug as a developer. We do all these visual testing in the cloud and we do a lot of unit testing in the cloud. Because we have a powerful machine, we can go through lots of text sequences at the same time.

Finally, code generation is super important for this developers that, today, building this stuff down to a device was like a manual process, "Okay. Let's take his big machine learning model from the cloud world and let's try to compress it down, quantize it, do a bunch of optimization and squeeze it down to a device." We've taken a very different approach, which is let's make really efficient algorithms to begin with and then lets auto generate optimize code for different Silicon targets. If we choose an Arm MCU, Cortex-M something, we can optimize for that and give very kind of compressed C++ math.

If we're targeting a CPU, we can take advantage of that. If we're targeting something larger, we can generate web assembly on the fly. Very kind of optimized JavaScript that can run on anything, and that generation of the actual inference that you go deploy to a device, that's a really important phase. Try to keep it independent of the algorithm so you could deploy this wherever you want. Those are a few of the pain points that developers like start to see really early on when they work on multiple algorithms.

[00:26:56] JM: I'm starting to see a really complicated set of problems that you have to attack in going to market. You think about the different plethora of types of organizations that would have devices that they would want to equip with machine learning. Think about sensors at shipyards, sensors in hospitals, sensors in the doorway in a restaurant, maybe sensors that are machine learning model that I might want to deploy to a security camera. You're trying to have some pre-baked models or some pre-baked algorithms that can be deployed to these different use cases if I understand correctly?

[00:27:46] ZS: No. Actually, we don't believe all that much in prebaked algorithms, and the reason is that developers who work at these domains, people that developed those products, they have very specialized sensor things that they're trying to detect. They're domain experts. They have access to test data from those cases and they need to develop algorithms that are specialized for them.

What we do, our philosophy, is we help those developers gather the data that they need for their domain-specific application and choose models of algorithms that are applicable for their problem. But once you take model template times their own data and you make those work together, those algorithms become specialized for that security camera or that shipyard detection algorithm, and that's the beauty of ML, right? We use data to train software to do something very specific.

It's not about the specific algorithms. It's about the infrastructure for these developers to be able to develop the algorithms and their purposes. There are brilliant people, right? The developers and engineers are already involved with edge devices, are brilliant engineers and software developers with their domain expertise. What they need is tooling to take advantage of the new technology.

[00:29:07] JM: Got it. Help clarify for me. If I've already got TensorFlow that I can train TensorFlow models and deploy those models to edge devices, what other kinds of tooling do I need?

[00:29:22] ZS: Yes. TensorFlow doesn't really do that today. That's the tricky thing. When you use TensorFlow, you're using it for a data science flow, which means you're working on a neural network for a very specific kind of classification task. What you get out of TensorFlow is still a fairly large model. TensorFlow doesn't solve the rest of this tooling problem that we've been talking about so far, which is dataset collection and dataset management. It doesn't solve libraries of algorithms and then really applying those datasets to lots of different algorithms. It doesn't solve the signal processing, preprocessing problems. TensorFlow assumed that you’ve done all that stuff somewhere else. You’ve sampled your data. You’ve windowed it. You’ve done preprocessing and as soon as that’s done somewhere.

Then it doesn't really solve the problem of deploying this algorithm to devices, right? We've created some nice tooling for like math to connect TensorFlow Lite to a microcontroller, some of the math operations, but it's a very manual process over there. It’s really hard, right? Even as an expert machine learning data scientist, it’s really hard to kind of take machine learning big models and really apply them to real-time data, sensor streams, audio.

One of our engineers, Daniel Situnayake, helped write a book on this. He helped develop this TensorFlow and wrote a book for O'Reilly on this. What he found is that this is super difficult engineering to go do this stuff. It's great to know how to do that, but most developers won't be able to become that specialized, become data science experts, machine learning algorithm experts and embedded experts. It’s just too deep of a stack.

[00:31:13] JM: Can you take me through a use case of somebody that you're working with and how they’re deploying machine learning models at the edge? What the lifecycle looks like? What their pain points are?

[00:31:27] ZS: Yeah. We could take – For example, we’re just getting ready to talk more deeply about a project one of our users worked on, one of our open source community users. We do free developer accounts for everybody. This developer, Kartik Thakore, he’s a data scientist at a health document company.

In his spare time, he wanted to work on cough detection using audio for COVID prevention. How much cough do we hear in an area? It could be in a whole. It could be in a public space. It could be in waiting room in a hospital. What's our risk level for potential transmission of flu in general?

Kartik wanted to work on this. What he did is he first wanted to find a suitable device that could allow him to put this kind of audio detection in a space. He found this Arduino Nano BLE board. A little microcontroller baseboard from Arduino that has an audio mic and a suitable processor with Bluetooth low energy all built-in, and this is like a $30 device. So it’s really cheap. He found that device. He started working with Edge Impulse as a tool to collect the data and use the algorithm libraries that are there to train those algorithms to detect cough.

What he did as he started out first. His first pain point was, “Well, how do we collect the dataset?” What he did is he used this little device to come collect some audio, but he also used a mobile phone to collect audio because, the mobile phone is an amazing sensor. You can collect audio very easily using a mobile phone. We have a couple different features that allow you to collect data directly from embedded devices, audio samples, but also we have a feature to use the mobile phone as a sensor.

Using the mobile phone as a sensor is actually really easy. When we add a device for data collection, we have the options to add a mobile phone as a device and we generate a QR code on the screen. You just take a photo of the QR code with your phone and that opens up a browser app on your phone connected to the cloud ingestion infrastructure, which turns the phone into a secure sensor.

Now, Kartik was able to use the phone to collect cough samples and stream them directly into the dataset tooling in the cloud. You could collect samples of different types of coughing. He combined some additional cough samples that he could find from the public Internet. Uploaded those to the same dataset, and he started to create a really good dataset around coughs, different types of coughs from different people and non-coughing events, sneezing events. Different types of background noises that you hear in public spaces to build up a dataset for a cough detection. That was first pain point with this kind of dataset maintenance.

The second thing he did then is he started to choose suitable audio algorithms out of our library. He took an MFCC signal processing block and spectrograms for coughs are no different. He set a window size that was appropriate for the length of a he typical cough. Coughs are pretty short in duration, and then you move that window over the audio to try to detect that event. Then he chose a neural network, and what happens with the neural networks is we have audio-based neural networks for classification. He chose an audio model out of our neural networks setup, which is good at detecting audio patterns over one window of audio.

His audio model is detecting cough event or no cough event. Keep it very simple to get statistics about background noises. With a couple weeks of work collecting a little bit more sensor data. Cough data, tweaking the parameters of the models a little bit, he was able to get a reasonable well-performing machine learning, tiny machine learning model for cough detection that was small enough to deploy in this type of device. He's been testing that with the community working on some different projects. We're actually working on blog to go and expose that dataset and show a tutorial on how to do this in a way that anyone can do it.

We recently launched with Hackster, a global challenge for COVID prevention techniques. Not things like ventilators, but really like down to earth, how can we help prevent and know how to prevent the spread of these kinds of diseases? In particular technology, that can be transferred to developing countries. So it has to be cheap. It has to be simple. It has to be open source that a developing country's local UNICEF or UN development agency can help transfer that knowledge of how to manufacture this thing. It's a local teams of volunteers, engineers that can go and put this into use. That's the kind of work we're encouraging people to apply, apply these kind of projects to this UN program.

[00:36:42] JM: Again, Edge Impulse in this workflow where Kartik wanted to set up a system for recording and detecting coughs. Where did Edge Impulse fit into that workforce? I understand that he bought this commodity hardware chip and he was setting up his phone to record the audio. But where was Edge Impulse in this workflow?

[00:37:06] ZS: All of it. From the very beginning of collecting sensor data, we have an SDK that goes here, and that SDK enables real-time data collection to the cloud interaction.

[00:37:18] JM: Got it. The SDK system, the chip, and do you manage the data actual datasets? Like you set up S3 buckets and stuff and just dump the –

[00:37:32] ZS: All cloud-automated. We also have the web application for the mobile phone and the generation of the QR code I mentioned. That's done in our cloud tool for enabling that data collection from phone in real-time. That's all our stuff. The visual management of the datasets, that's part of the tool, as well as the libraries of algorithms graphically, the training processes, the testing and the code generation. The deployment of the actual algorithm back to this device or back to the phone, we support web assemblies. You can generate something that gets deployed right through the phone. That's all part of the Edge Impulse developer platform, which is a cloud developer environment. You sign up for an account, you get a free account and then you can go and collect datasets, manage those datasets, train algorithms out of our libraries of algorithms or create your own and then do that testing in deployment phase. That's all part of the platform.

[00:38:35] JM: Are you thinking of it as building infrastructure and workflows for individual use cases like related to audio recording? Like you have a certain workflow for audio recording and then you have a certain workflow for temperature detection. Do you have these different domains that you're focused on?

[00:38:54] ZS: Yeah. We call those wizard templates. That's something we're just working on releasing, which is kind of out-of-the-box wizards when you get started where we ask you, "What kind of project are you wanting to work on? What type of sensor data do you have and what kind of pattern are you trying to detect?

We are right now working on templates for motion. Different types of accelerometer, gyroscope-based applications where we're detecting human motion or machine vibrations and machine motion. They're not really different from a machine learning point of view. Once you choose that from the workflow, what you'll get is a template where we choose the correct sets of algorithms for you out-of-the-box. We might have some specific settings set for that algorithm that work better. Then we'll ask you whether you want to start collecting your own dataset from scratch or do you want to use one of our existing datasets to start with. We have example datasets for that.

Different types of use of vibration and accelerometers are super, super common across almost all application. That's one we're focusing a lot on. The other one is audio patterns, and there's two different types. We have human speech keywords as a template where we detect things like hello or open window, any kind of short command. That's the type of template that we're working on. The other one is background noises, things like coughs or a machine breaking, for example, a siren. That's the type of background noise we can detect. Yes, giving a template for that, what's the best set up? Here is an example dataset that you can start with, or you can collect your own data for your own dataset.

We're just working on image-based technology. The ability to have templates for image anomaly detection, image classification and different types of camera boards where you easily collect those images for a dataset. Yeah, and lots more coming. There's more exotic applications that as we see more of those, like electric current pattern detection. You got current clamps that go around your current lines. Your electric lines going into a house or into a machine, you can detect patterns in that. It's not all that different than vibration analysis. Just a different type of sensor input. Yeah, templates are great way to get people started.

[00:41:16] JM: When you see all these – Like a new sensor like that, electrical current measurement, and you see the software platforms develop. I mean, because you are already developing the software platforms, making them more usable. I imagine, you have some pretty amazing ideas about what the future might hold for if we could just magically flick a switch and have sensors put on to everything and intelligent developers doing interesting things with those sensors. I'm sure you have all kinds of ideas about how that could be useful and how that could make our world much more productive. Unfortunately, we won't see that with the Toronto Smart City, the Google Toronto Smart City. I guess that's been canceled for now. But tell me about what's your vision for like the bright future of the world once we have that kind of mass deployment assuming we would actually have the developer uptake in all these different sensors and application development?

[00:42:23] ZS: The beautiful thing about – We have lots of wild ideas just between our team, but the beautiful thing about running a developer platform where we let people sign up for free and do things, is that we get to learn from them what they're working on. They come and talk to us and tell us and we help them along.

We've got already over a thousand developers using this, and the kind of things they've told us they're actually working on and showed us are amazing in itself. Great examples of this are predictive maintenance and industry, where we're able to train an algorithm to detect when a very specific machine is starting to break well ahead of time with a really cheap kind of like stick-on sensor. That's really amazing, because you can go detect the types of things that could shut down old factories for months. Now, with COVID and social distancing, that's getting even harder, right? Because we have a lot of machinery, a lot of like industrial complexes that are up and running but with very small amounts of staff.

Monitoring those kind of things, we've seen a lot of applications in. We've seen a lot going on in nature conservation too. A big chunk of our users are doing different types of nature conservation projects and things that people are starting to work on are, for example, detecting populations of species through field cameras. Like a camera you put in to watch wildlife, it gets triggered by motion. But identifying endangered species immediately on the camera, which is usually a really hard process to manually.

We have a new project kicking off around thermal detection of elephants in field cameras. Looking for the numbers of elephants that are being seen through thermal cameras in the field and differentiating the elephants from people, which could be an indication of poaching, versus livestock. Doing that in a very small battery-powered device in the field is another project.

We have another project that users are working on around dolphin audio detection in oceans. Detecting when there is the presence of dolphins through a machine learning in real-time, and that's an amazing project. We had people working on agriculture. One really interesting project from a developer out of Egypt is a product that detects the presence of pests in palm trees through vibration. Putting a sensor right into the core of a palm tree that has a very accurate vibration sensor can actually detect these pests, because they act a little bit like termites. They start doing things inside the tree that you detect through the patterns of the vibration and you use machine learning on the platform to do that. Just fascinating applications that seem a little science fiction, but they're actually being done right now by real developers.

Another great one we had just hit Twitter the other day, one of our developers use the gas analysis sensor. A very cheap, 35 bucks, on the embedded microcontroller to detect the very, very accurate gaseous patterns of different types of alcohol, and he did this just because that's the chemical he had around home, which is just bottles of booze. He could detect literally different types of rum from the gaseous signatures using machine learning, different types of whiskey from the gaseous signatures. That's something you can use for different types of chemicals, different types of household items to tell the difference between them just using a sensor and machine learning. That's another fascinating application that we saw just recently from the developer community, and then lots of people working on advanced health monitoring.

Really interesting application we've seen come across the board is analyzing people's pulse signatures using optical sensors, really cheap ones, to detect how people are sleeping. Do they have the flu? Is an elderly person stressed about something in panic? You can match that would audio and skin temperature, skin conductance, motion to detect a lot of different things on how someone's doing. Is there potential safety issue or health issue? Do they need help? Lots of applications in that space.

[SPONSOR MESSAGE]

[00:46:49] JM: The Intrazone is a biweekly conversation and interview podcast about SharePoint OneDrive and related technology within Microsoft 365. Get highlights on usage, adaption and how SharePoint works for you. Subscribe to The Intrazone today. That's The Intrazone, I-N-T-R-A-Z-O-N-E. It's all about show SharePoint fits into your everyday work life. The goal being to more easily share and manage content, knowledge and applications and to empower teamwork throughout your organization with the technology that you already have. Subscribe to The Intrazone wherever you get your podcasts. Each episode of The Intrazone covers important topics and information about Microsoft SharePoint, OneDrive and related technology within Microsoft 365, including segments on news and announcements and topics of the week. There're guest perspectives from various users and partners and customers and there is upcoming events and conferences and workshops. You can find all of this on The Intrazone podcast, I-N-T-R-A-Z-O-N-E.

[INTERVIEW CONTINUED]

[00:48:09] JM: You mentioned that there's a lot of work to be done around large datasets. You have these sensors and you have them potentially picking up really high-volumes of continuous data and needed to do things with that data. What kinds of infrastructure have you had to build to manage – I don't know, buffering, and batching, and compression or just tell me about some of the data management issues?

[00:48:38] ZS: Yeah. It's amazing platform that you have to build to do all these. On the very edge, it's interesting, because you need to give the developers tooling to package up this data to begin with. That was a problem we had to solve early on, because today's data collection formats really aren't designed for small devices with small amounts of storage space, nor for machine learning data.

For that, we developed a new JWT-based JSON format. It's a JavaScript web token or a JSON web token, which is like a sign token in JSON that's specifically for machine learning on data samples. You have to be able to compress that in a very small way. So we applied CBOR, which is a really cool binary object notation for JSON. So you can basically compress JSON down to binary tokens. We use that to make sure we can package data up securely and properly from any edge device, and that could be right from the sensor, from the mobile phone, from your file system. It doesn't matter. That's one of these pieces of infrastructure we had to build early on.

The other thing we had to do is build a global ingestion infrastructure, so something that hundreds of thousands and millions of users and enterprises all around the world can throw data at us, because we're getting these terabytes of data streams from these users. We have a very large AWS-based infrastructure for ingesting data in real-time and making sure that we know that we really received every secure bit of this data from lots of different file formats around the world. The large data ingestion infrastructure for these dataset creation techniques.

Then being able to work with large amounts of data is an interesting data engineering problem. We have users working with lots of smaller projects that we have to create data live in, and then we have very large datasets from our large users. We have kind of multiple layers of infrastructure where we have buckets that contain very long-term large datasets, and we even work with customers with their own buckets, right? So they actually own the data right in their accounts.

Being able to do that and having the tooling to work with that is a really interesting big data problem. So we have things like query tools where we can query down into large amounts of data to narrow it down to what we want, and we have what we call transformation. We do parquet-based container transformations.

Let’s say you have a large dataset, you want to multiply a bunch of preprocessing to that data so that you can use it in machine learning projects. We have a special way of maintaining containers with transformations of Python inside that we can apply to lots of files in parallel. If we’re processing thousands of files, we can apply that in parallel. Something that might take a week takes hours, right? That’s stuff that’s really hard to do in your own machine or with off-the-shelf tools.

Then we have to work live with this stuff. So we have a lot of interesting database technology to go from like frozen datasets to live datasets, because in the tooling, we have to do a lot of graphical stuff. We’re right away showing what the data looks like. What is the preprocessing look like when you’ve applied signal processing? What are the features look like, and that stuff that has to happen in the UX immediately? There are a lot of live database techniques, even taking like pre-compress samples of data so that we make the UX work nicely. That's another layer of stuff we do with the dataset. Then there's a lot with just data set management, right? Graphically showing how many samples of different labels do you have? Are they different? Do you want to move some of your training data to your test data? Do you want to move it back? Just managing this training dataset versus test dataset graphically for users is important. Because as you're working on your project, it’s iterative. You collect some data, you apply to algorithms, you test it, and then you go collect some more data to make it better. This is an iterative process that you work on as a developer. Just like as a software developer, you write some code, you test it and you try it. You write a little bit more code. In ML, you’re doing the same thing, but with data.

[00:52:52] JM: The actual deployment of the models, the continuous deployment of the models, testing the inference to make sure that the inference is still good after you've updated your model. Can you tell me a little bit about the workflow for testing and deployment?

[00:53:09] ZS: Yeah. We started with datasets and we have a lot of this tooling we just talked about to visualize the datasets to see what they are and do you have enough of different kinds of data and then collect more of that data necessary. But then what you need to do is you need to apply that dataset to algorithms.

The other really interesting thing we do is when we host these algorithms in the cloud, you can think of that hosting kind of like GitHub host code repositories for you. We host algorithm repositories for the user. We have blocks of algorithms that you can chain together to process data. You start from windowing into signal processing and into different types of machine learning blocks, and those are libraries. You can choose new algorithms, change those algorithms, get templates for different problems.

But what you then need to do is you need to actually apply this dataset to the algorithms. What we end up doing is hosting a very large Kubernetes compute cluster with a lot of Python machine learning data science tooling that are hosted in the cloud for the user. That lets us do things like generate all the features from a dataset and show them right away graphically, which is something that we do. It lets us train the neural networks in the cloud rather than having to do this manually in a tool on your machine. We can automatically go and to take that dataset and apply it to a neural network and train a neural network and immediately see what the results are graphically. That's something that we end up having to host in the infrastructure, and we have to do things like show performance as well. Something nice when we are hosting this is that we can apply special tools that we've developed to analyze the complexity of the math in the algorithms and show the developer right away how fast will this be on an embedded device of target type A, like an MCU, or what if it's a CPU? How fast will this be? So we can right away show what we call on-device performance where we can show the amount of inference time and the flash and RAM before we actually move the algorithm to the device. It's really useful to know ahead of time so you're not trying to squeeze a big model into a small device.

Then you asked about testing. What we start with testing is what we call live classification, and this is where we stream sensor data from a device to the cloud and then show visually what's happening, and does it work, because it's hard to debug to stuff on device.

For example, I can have existing samples that I've streamed to the cloud earlier. I can replay those. So I can do like a classification replay live in the cloud. Then we visually show what happened here, right? For every window of data that came in from your sensors, how many events did you see? Was this classified as a certain kind of motion? What does this look like visually in the feature space? Is this data similar to the training data you've seen before? That helps explain why it worked or why it didn't work. Did we see anomalies, right? Is this data very different from the data that we've seen in the past?

Showing all these graphically is important. Then it helps us decide what to do next, "Oh! That worked really well. Great! Let's keep this as a test case," or that didn't work very well. If we run an anomaly through device, that anomaly might actually be data that we want to to detect. We just haven't done training on that yet.

For example, if I have an anomaly, I might want to push this back in the my training set. Being able to do that graphically and see that, "All right. This is something that was misclassified, but I want it to be classified. Let's move it to the training set," and those will retrain our algorithms so that next time this will work. That's a really typical development flow.

Then we implement unit testing really carefully so that as you do test cases towards your algorithm, you store those test cases and you mark an expected outcome to it. Not all that different to how we do unit testing for software. You have expected outcomes. You have test cases. What we do is we automate the running of large numbers of test cases for the developer so you can continuously look at, as you improve your algorithm, have you broken your test cases? If an algorithm detected a certain movement before, you want it to continue to work for that even as you improve it. That's a tricky thing with machine learning and the tools today don't really do unit testing for you. We're trying to bring that unit test methodology into machine learning. Running 40 different test cases in parallel is a lot of compute. It's nice when we can spin up 40 different containers in Kubernetes and do that automatically and then show the results.

Then finally we just generate code, right? When we deploy to a target, we choose the target like we want a C++ library. What we do is we go and run all of our tooling to compile our signal processing down, compress our neural networks, cross-optimize. What you end up getting is a C++ library that source code. It's just the math, right? Data in, probabilities out. We do the same thing with web assembly as a package with source code, or if we know the target you're working with one, it's one of our official targets like a specific dev board or a specific router, we can even go and build a binary. You basically get a full binary with that inference built in. You just drag and drop to the device and it starts running. That's a special case when we know more about device. Most typically when you're going into like production deployment, you're doing C++. You just build that into your firmware project, CICD tools, firmware update like you normally would.

[00:59:11] JM: I know we got to wrap up. I just want to say for the listeners, Zach just showed me the platform, which is a pretty amazing set of tools just so you can write your machine learning algorithm into the platform. You can train the model. You can configure everything. It's everything in a single platform and being able to select the libraries and the deployment, the code generation. It's all pretty impressive. It looks pretty convenient for somebody who just wants to simplify their workflow to the extent that we can simplify machine learning workflows today.

[00:59:52] ZS: Exactly. Making sure that developers are successful with this, because this really hard stuff. The chances for failure in trying to do machine learning on a small devices is really high. Try to make this just tractable for developers. I think it's really important that we are designing really great ML ops under the hood. Something we're careful of is putting really proper ML ops machinery under this so that as people go to production and at scale, for products, we have APIs that you could drive to automate the entire process through a normal CICD process. Everything that we do, we have an ML ops philosophy under the hood as people go to production in a large scale.

[01:00:33] JM: Okay. Well, Zach, thanks for coming on the show. It's been really great talking to you.

[01:00:37] ZS: Thanks, Jeff. No. This is great. Awesome questions. Look forward to chatting more.

[END OF INTERVIEW]

[01:00:50] JM: Over the last few months, I've started hearing about Retool. Every business needs internal tools, but if we’re being honest, I don't know of many engineers who really enjoy building internal tools. It can be hard to get engineering resources to build back-office applications and it’s definitely hard to get engineers excited about maintaining those back-office applications. Companies like a Doordash, and Brex, and Amazon use Retool to build custom internal tools faster.

The idea is that internal tools mostly look the same. They're made out of tables, and dropdowns, and buttons, and text inputs. Retool gives you a drag-and-drop interface so engineers can build these internal UIs in hours, not days, and they can spend more time building features that customers will see. Retool connects to any database and API. For example, if you are pulling data from Postgres, you just write a SQL query. You drag a table on to the canvas.

If you want to try out Retool, you can go to retool.com/sedaily. That's R-E-T-O-O-L.com/sedaily, and you can even host Retool on-premise if you want to keep it ultra-secure. I've heard a lot of good things about Retool from engineers who I respect. So check it out at retool.com/sedaily.

[END]