EPISODE 1146

[INTRODUCTION]

[00:00:00] JM: Containers and virtual machines are two ways of running virtualized infrastructure. Containers use less resources than VMs, and typically use the Run C open source container runtime. Syxbox is a containerization runtime that allows an alternative to Run C and allows for the deployment of Docker or Kubernetes within a container.

Cesar Talledo is the founder of Nestybox, a company built around the Sysbox runtime. He joins the show to talk about container runtimes and his new company.

[INTERVIEW]

[00:00:37] JM: Brad, welcome to the show.

[00:00:38] B: Thanks for having me.

[00:00:40] JM: You work on Roboflow, and one way to look at Roboflow is as extract transform load, or ETL fork computer vision. So, many people have heard previous episodes about extract transform load. They know that pattern. Explain how that applies to Roboflow.

[00:00:57] B: Yeah. So when we got started with Roboflow, it was because we are building our own augmented reality applications that were powered by computer vision. And we felt this really big frustration around kind of that ETL layer of things, where it seemed like there were purpose built tools for doing things like labeling your images and training your models. But there is this like peace in between, in between labeling and training where everybody basically just has to build their own one-off Python scripts to do all these like menial tasks that aren't really important to the problem.

And so when we're asking around all of our friends like, "Hey, how do you all do this?" It seemed like everybody was just like reinventing the wheel. And so we figured since we were going to have to write these tools internally for ourselves for our own app, we should release those out to

the world. I mean, actually, now we've pivoted the company to be completely developer tools focused primarily on that niche right in between labeling your images and training your model.

[00:01:48] JM: Okay. And so explain what a prototypical workflow would look like for Roboflow.

[00:01:53] B: Yup. So we're like universal conversion tool. You can imagine us as kind of like an open platform that plays nicely with all sorts of other point solutions. So we're pretty agnostic about where you do your labeling and training. We can accept over 34 different annotation formats ranging from like outsourced labeling tools like Scale or Amazon SageMaker. Or if you want to label them yourself, we support all of the major labeling tools. So you label your images and then you get these annotation files out of them. And one of the biggest pain points that you'll feel is that when you export your data from all these tools, they come in format specific to the tool, which are not like compatible with any of the models that you're going to want to use. And so kind of one of our hooks is we can import all those tools and then output to all these other different formats. So that's one of the ways that people find us, is they're looking to convert VOC XML to TF records for TensorFlow.

And so the way that it works is you come into Roboflow, you drop in your images and annotations and we perform a bunch of checks up on them. Help you make sure that that data looks good and is ready to go for model training. We'll help you augment your images, preprocess them. Do all those sorts of things that you would kind of have to do one-off Python scripts for and then we'll export them and then you're ready to train your model.

[00:03:09] JM: And how does this save time?

[00:03:11] B: Yeah. So most of our customers are looking at this as kind of a tradeoff between doing it themselves by writing all those Python scripts or using something like Roboflow. So we talked to one of our early customers and asked them like, "Hey, how much time are you actually saving by doing this?" And they told us that it had been on their backlog to try out a Pytorch torch model. They were a TensorFlow shop for quite a while but had never like gotten to the top of their stack, because it just seem like such a pain to switch all their image processing pipelines, convert all their annotation formats and all that stuff. And so they estimated that that would've taken a week of development time for their team of three. But with Roboflow, then they

were able to do it with a single engineer over the course of an afternoon and they had a Pytorch model trained and actually found that it worked better than their TensorFlow stack that they'd been using for quite some time.

[00:03:57] JM: Can you explain what annotation formats are? Why are there multiple annotation formats?

[00:04:03] B: Yeah. So and annotation format describes where the examples are in your images. So when you train a machine learning model, you're kind of training it by example you're showing at like, "Hey, here's the thing that I'm looking for. Try and figure out how to replicate these inputs that that give you." And so an annotation format is just a way of encoding that. It could be XML. It could be JSON. It could be a binary format like TF record, or a text file, or any number of other different formats. And it seems like the reason that there is so many different formats is that everybody just kind of invents their own thing internally and then releases their stuff as open source. And so they don't play nicely together. So you have all these researchers that are publishing papers and they're not thinking about like, "Oh, how are people going to use this to actually deploy stuff into the wild? They're thinking about how do I get state-of-the-art results and release something? So it's all these kind of like hacky like academic code rather than production code. And then people take those research papers and they're trying to convert them into production, and they end up like having to use basically whatever default format the researcher had in their paper to reproduce that. And so you end up with this just kind of spaghetti-ness of all these file formats that don't play nicely together.

[00:05:12] JM: Do you have a prototypical example of a company that's using Roboflow

[00:05:17] B: Yeah. So it spans the gamut. Everything from hobbyists that are building like raccoon detectors to train little robots to chase them out of their backyard, to some of the world's largest companies. So we have a Fortune 500 oil and gas company that using Roboflow to train models within their security camera footage to look for oil leaks in their pipeline. So previously, this was something, these are like unmanned 10,000 miles of pipeline and they have like these little leaks that turn into big problems and they don't like get addressed until a human notices them. So they're training models to automatically look at their security camera footage vision real-time and alert them so that they can fix the tiny leaks before they become big problems.

[00:05:57] JM: Let's go through a little bit more on the raccoon example. So let's say I'm training a model to recognize raccoons and chase them out of my backyard with maybe some served drone or like a Roomba like robot. What would be my workflow with and without Roboflow?

[00:06:14] B: Yeah. The raccoon detector is a funny example, because we actually have six distinct users all working on detecting raccoons. And so one of them is trying to automatically turn on their garden hose to spray the raccoons. One of them has this little spider robot that he is trying to train to like chase them. So it's been kind of a surprising niche that we've found.

So kind of that the workflow would be most these guys are using like nest cameras to export a bunch of images from your nest camera. Find some of the ones that have raccoons and some of the ones that don't. So like one guy is trying to make sure that his robot doesn't also chase his dog. So he wants to label his dog. He wants to label the raccoon. And then he'll end up training a model to do that in real time. So without Roboflow, he'd be writing all these Python scripts to like after he labels his stuff to convert them, to augment the images so that it works well in the daytime and nighttime, whether it's a cloudy or not, if the camera happens to be at a slightly different angle, if the wind has blown stuff around. And so instead of collecting millions of different images, he'll do something called data augmentation, which it just like helps your model generalize more by giving it more examples that aren't all exactly the same. So those are some examples of things that you'd be writing Python scripts to do and are very specific to like the input format for one specific model.

With Roboflow, that's all done within the platform. And so you can try a whole bunch of different experience experiments much more quickly and then export them and you can try maybe efficient det or Yolo V5 or Yolo V4 on Pytorch, TensorFlow, Darknet, and you can try all those things over the course of a week rather than having to spend a week each basically.

[00:07:53] JM: What have you had to build in Roboflow? Tell me a little about the infrastructure.

[00:07:59] B: So probably the biggest parts of the app are like our annotation parsing system. So we support all these different formats importing and exporting from them. So I think when

you do all the combinations, it's like 400 or 500 different combinations of X to Y formats. And previously if you like Google like converting Pascal VOC to COCO JSON, there's like a specific Python script that you can find on GitHub to do that one specific thing. And so we spent a considerable amount of time basically abstracting that so that it's really easy for us to provide this kind of abstraction layer over the top of all these different formats.

And then we have a pretty robust image transformation pipeline that spins up servers to do things like rotating images and adjusting their brightness and contrast, resizing them and getting them ready for training. And then we also have another pipeline that takes all of those transformed images and compiles them together into one output format. So that might be a TF record binary format for TensorFlow, or it might be like a zip file containing all your XML and JPEG images for another format.

[00:09:04] JM: Tell me about some of the particularly hard technical problems you had to solve in building Roboflow.

[00:09:09] B: Yeah. So some of things that were surprising to us, I guess not surprising, but we didn't think that we'd have to address them so quickly, was the sheer scale of things. So we started out by using cloud functions to do a lot of these tasks. What we quickly learned was that cloud functions have some technical limitations in terms of like the amount of time that they can run and the amount of memory that they can use. So you can imagine if you're creating a dataset with a thousand images, that might work fine. But then all of a sudden somebody uploads 250,000 images and all the sudden like that just breaks. And so we had a kind of replicate and abstract away from that. And so we internally have basically our own version of Google Cloud functions or AWS Lambda that runs on Docker containers now instead so that we can create images of arbitrary size. We can add GPs to those that do some things that you're not able to do with the native serverless cloud functions.

[00:10:04] JM: Sorry. So what were the problems with the cloud functions?

[00:10:07] B: Yeah. So I think on AWS and Google, it's 2 GB and 3 GB is the maximum amount of memory and disk space you can use on those. So you can imagine, like if you're trying to create a zip file that's a hundred gigabytes, it's really a tough technical challenge to do that on

an instance that can have maximum of 2 GB of memory and disk space storage. And they're also limited to only running for a certain amount of time. So I think on Google that's nine minutes. So if you're trying to download 250,000 images and process them all on one cloud function and output one file, if it takes longer than nine minutes, you're just kind of out of luck. And so by building our own version of those sorts of tools, we can eliminate some of those kind of hard barriers that you're not allowed to overcome on serverless cloud functions.

[00:10:56] JM: So if you're trying to be this sort of middleware stitching together all these different frameworks and toolsets, it seems like there's probably some issues in being that glue between them. Are there any particularly difficult problems that you've had to solve in gluing together these different frameworks?

[00:11:17] B: I don't think there's necessarily like big technical challenges there. I think there is like philosophical things that we've had to overcome. So you can imagine some of the big cloud providers are also trying to be this like end-to-end machine learning platform. And if you look at AWS's like incentives on their platform, they really want to lock you into using the AWS labeling tool, and the AWS notebooks, and the AWS training, and the AWS deployment stuff. And what we found is that doesn't really work well for a lot of teams. There are all these good point solutions out there that aren't built by AWS, and like that kind of end-to-end platform where like you either take the whole thing or none of it at all just doesn't work for folks.

And so I think one of our like core observations is that if we can help people use the best point solutions for each step of the pipeline, that can be really powerful. And is one of the reasons that people choose us is that we're like interoperable with all these other tools. We don't have to like control your training flow. We don't have to control your labeling. We can just play nicely with everything and help you use those altogether and be kind of the tool that just reduces friction between all these other parts of the pipeline.

[00:12:28] JM: Do you find that people pick a particular cloud provider and just go all-in on it like on AWS? Or they go all-in one the Google TensorFlow stack? Or do you find the people really want heterogeneity?

[00:12:42] B: Yeah. It really depends. We have a couple of different like customer types. And

certainly there are like really advanced companies that have built out a team of PhD's that are working on computer vision. And those kind of more mature teams definitely have their workflow that they prefer. But there're also these other teams that we find that are kind of just getting started. We think one of our like core, like mission statements, is that we want to make computer vision something that all software developers can use. You shouldn't have to hire a team of PhD's and be a company the size of Google to use it. And so by letting these teams experiment more quickly with their existing engineering resources, you let them come to find what works best for their problem.

And so when you're first like exploring a problem space, you don't know like whether TensorFlow or PyTorch is going to be the right solution for your problem. You probably want try them both. And so we help you like navigate that kind of problem space really efficiently and quickly and find what's going to work best for you. And yeah, we have seen teams like switch from TensorFlow to PyTorch, but usually it's in the course of experimentation. Machine learning is one of those things where like it's never done. You're always iterating and trying to find something that works a little bit better. And so if a new model comes out and it is has research results that work better but the code is in PyTorch and you're in TensorFlow, you don't want to be stuck in your legacy platform if there's something that's going to work better. And so helping them experiments and use whatever tool is best at a given moment in time I think has been really valuable.

[00:14:07] JM: Tell me more about what you need to do to integrate with the labeling providers.

[00:14:12] B: Yeah. So it's actually been interesting. We assumed that most people who are doing computer vision and production, we're going to be outsourcing their labeling to tools like scale or AWS SageMaker ground truth. But what we found is even a lot of big companies are still kind of in this experimentation phase where they're just doing things in-house. And so they're having these highly paid, highly skilled engineers that are labeling bounding boxes on their images just because like It's kind of big like problem and pain point for them to go out and source of provider. And a lot of these providers have like big minimums that they have to spend. And there's like a procurement process.

And so we feel like if we can like reduce the amount of friction to that and free up the time of

developers to be working on the things that actually need their unique skillset, that can be really powerful. So right now we integrate with basically all of the self-serve labeling tools. So whether they've used CVAT or VoTT or another like tool to label the images themselves, they can import those all directly into Roboflow. But we also have helped many teams outsource their labeling for the first. And so there's a bunch of different providers. They all have their own pros and cons and we feel like we can help match those users with the labeling provider that's going to be right for their use case that can provide a lot of value to them as well.

[00:15:30] JM: And what about the model training tools? What are the integration with the model training tools like?

[00:15:37] B: Yeah. So we have a model library that has co-lab notebooks that are set up to do most of the like modern, state-of-the-art object detection models. So you kind of pick those up and play around with those. We also have integrations with all three major cloud providers, AutoML tools. So you can try Google's, you can try Microsoft's, and you can try Amazon's and kind of get a baseline for what is the like naïve level of performance that you'd get just kind of off-the-shelf. And that something that's hard to do right now. You'd have to integrate with each of their individual APIs to try them out. But with Roboflow, you can just try all three and see how it works.

One thing that we've found is that a lot of these software developers that aren't machine learning experts that are using computer vision via Roboflow, they go to our model library and they have these like Jupyter Notebooks, and all they're doing is hitting enter-enter-enter-enter. They're not really like doing anything custom. And they end up with this weights file that they don't know what to do with them. So when they want to deploy it and use it in their application, they're then trying to like figure out how do I spin up servers and host this and like build DevOps infrastructure around that?

And so one of the places we're moving into is providing our own sort of hosted training and deployment environment for some those users who just want to use computer vision, get something that works well enough, built it into their application and not really worry about all the details of tuning their model and whatnot. And so as we have talked to users, we've found out that while we're solving a bunch these problems and eliminating like the boilerplate Python code

that they have to write, by getting them over that hump, they then hit these other problems like, “Oh crap! Now I have this trained model. What do I actually do with that?” And so we’re really trying to make that easy for them to integrate computer vision to their apps and focus exclusively on the things that are unique to their app and not the things that are kind of this boilerplate computer vision infrastructure that's reinventing the wheel of things that's already out there and not providing unique value to their domain-specific problems.

[00:17:39] JM: On your website, you have some areas where people can share datasets and seems kind of random compared to your other tools. What's the objective with these datasets, the shared datasets you have on your website?

[00:17:54] B: Yeah. So we feel like beyond providing the tools, if we want to enable any developer to use computer vision, we need to kind of like chop down those barriers that make it hard. And one of the things that we found is that a lot of developers, they have like a problem that they have like this inkling of an idea that they could use computer vision for. But they don't have like a dataset that they've already gone out and collected. And so they just like this hump where they don't have something to try it out on. And so we figured one way that we could get them over that hump was to provide a whole bunch of datasets that they could use to play around and try things out.

And so we curated and released a bunch of open source dataset. Some of which we collected. Some are from our users that were willing to share those with other researchers and some that were already open-source that we either improved or converted. And so that's kind of like one of those humps that like if you don't have a data, it's really hard to get started learning computer vision. And kind of along those lines, we feel like education and like teaching people how to use computer vision is another big stumbling block. If you're such software developer, computer vision can be one of these things where it feels inaccessible and like something where you'd have to go back to school to use it. And in fact, I was a software developer before with no computer vision or machine learning expertise, and at one time it seems like an insurmountable hurdle to me too. And so we feel like educating people and like putting out tutorials and making sure that we’re like doing everything that we can to democratize this technology and make it accessible and a part of every developer’s tool chest is something that we should play a part in. And so we have tutorials, we have YouTube videos. We have those public datasets. We have

those open source models, and we're really trying to do everything we can to make that hurdle to getting computer vision into your app as low as possible.

[00:19:42] JM: And what are the main hurdles to getting computer vision into my application today? Let's say I'm building like a to-do-list app. I'm a brand-new developer. I'm building a to-do-list app and I want to have computer vision in my application because I want to – I don't know, take a picture of a blanket and have it recognize that it's a blanket so it can tell me to fold the blanket and put a to-do on my list for folding a blanket. Why is that hard today?

[00:20:15] B: Yeah. So one of the biggest hurdles actually is for software developers to even realize that this is something that they can do. I mean, for the first 50 years of computing, teaching a computer to understand image data was like an intractable problem. It was just something that even with a team of PhDs you can do. And it's really only been in the past decade that this has become accessible to not only teams of PhD's, but just a single solo software developer off the street.

And so one of the biggest challenges is just convincing people that it is possible and that it is something that they can do. Once you get them over that hump, a lot of the other stuff is just normal software engineering stuff of getting a Python script up and running and following a tutorial and getting through things. And then I think the last challenge is on the deployment side. It gets kind of complicated when you're looking to actually like deploy it into the wild, because whether you're deploying it on a server or on a mobile phone or an embedded device somewhere else, you almost have to start from the end and think about like, "Where am I going to put this?" And then that informs a lot of the decisions beforehand. And so it's kind of like this like forwards pass of I need to figure out that I can do this. And then a backwards pass of, "Okay, so I think this is tractable. Now, I want to do this for real. If I want to put this on a Raspberry Pi, what considerations and decisions do I need to make before that to make sure that the model that I come out with is deployable there?" And so, yeah, I think it's kind of like an iteration process of getting over the hump of like training your first model and then creating your first project that you can use in the wild.

[00:21:55] JM: Let's take it from the top again. Let's say I have a bunch of images of a chessboard and each image has like a configuration of a board situation and I want to generate

like solutions to use to those chess problems. What would be my process for using Roboflow do that?

[00:22:20] B: Yeah. So actually this is a great example. My cofounder and I actually built a computer vision powered chess solver for a hackathon Techcrunch Disrupt last fall. So maybe it would make sense for me to walk you through how we did that. So we came into this hackathon with basically nothing. We had a chessboard. We had an iPhone. We had our laptops and that was it. And so the first step was setting up the chessboard and setting up a bunch of positions, taking pictures of those with our iPhone. And then you offload those pictures from your phone. And now you have the unlabeled images. So you need to label those images.

We brought those into a tool on the Mac called RectLabel, which creates indentations in a VOC XML format. So you go through, you draw a box around each individual piece. You tell it this is a white queen. This is a black queen. This is a pawn. And then you have this kind of serialized format of what is the state of the chessboard. And then from there, at the hackathon, because we didn't have Roboflow yet, we run a bunch of Python scripts to like modify that, resize the things, create some augmentation so that it would work depending on different lighting. With Roboflow, you would just drop those images and annotations into our software and you get a gui where you could play around with all those different settings.

At the hackathon, we then train the model. We used Apple's tool called CreateML, which is a no-code training platform. With Roboflow, you could still do that. You just click, “Hey, I want these annotations and create ML format.” Hit go. You get a zip file. You drop those into the app and hit train. You could also with Roboflow say, “Hey, I want to train these on AWS with their recognition custom labels.” Or I want to use Roboflow Train, which is our competitor to that. You click a button, you get a model.

So for us, training the model at the hackathon was something that we did overnight the first night. So while that was going on, we are working on scaffolding out the app that was going to consume this model. And so it was taking images from the iPhone camera. It was going to feed them through this black box model and get back JSON results essentially of like is it looking at a chessboard? Where are the pieces? And then you have basically a traditional problem to solve, right? You have the location of these pieces that you have serialized to like say, “Okay, so this

X-Y position represents this position on the board." And once you have that, then you can feed it to like a chess solver app. I think the one that we were using – I can't remember it. I think it was Stock Fish at the hackathon. And so basically like you're treating that computer vision as a black box that like converts your image into like usable computer data. And so it's like these two concurrent processes of developing the app and then developing the model that then end up working in tandem.

[00:24:59] JM: The different phases of using Roboflow; analyze, preprocess, augment, convert, export and share. Could you go through each of these in a little more detail?

[00:25:12] B: Sure. Yeah conversion, I think we we've touched on. There're all those different formats. And so we're like the universal conversion tool where you can import in one format and export in another. And one of the ways that we like to think about that is if you are an author and you were spending a bunch of your time converting like .doc to .pdf as part of your process, that would be ridiculous. And that's kind of how we think about engineers and machine learning people spending a bunch of time converting formats and writing python scripts to do that. It's just kind of a ridiculous thing that you would have to spend any time writing file conversion formats in 2020. And so that's kind of the piece of the process that we handle with the conversion side.

On the analyze side, so we have these tools that once you upload your annotations into Roboflow, we can perform checks and like tell you, "Oh, hey. This was a malformed indentation that's going to cause problems with your training script." We automatically fix a bunch of those and we bring to your attention other potential problems. So as examples of that, some things that you run into you when you're training a machine learning model are like class and balance. So let's say you have your chessboard images and it turns out there is 16 pawns on the board for every one queen. You're going to end up with your model overweighting and seeing way more pawns than it sees queens. And so you'd probably want to rebalance those things so that your model is not able to cheat by just like guessing pawn, because that's what optimizes its score, because in like the later game, like there's going to be less pawns and more queens relatively.

So we help you like identify by like class imbalance. We help you identify things like, Hey, this

queen on this chessboard, like in 90% of your images, it was on the exact same portion of the image. You might want to augment that so that your model doesn't learn, “Oh, the queen is always on the same white square,” and learn to basically like cheat that way. We can do things like re-cropping the image or translating that bounding box around or going out and taking more photos with more examples of the queen on different squares.

Augmentation is what we mentioned earlier of making sure that your model generalizes. So doing things like adjusting the brightness and contrast, rotating it, cropping it. There are some advanced augmentations that you can use. One of which is called mosaic, where it will take multiple different images from your training set and it will combine them all together to create like an image that has four pieces of other images. And the purpose of augmentation is really to help your model generalize. If you feed it the same image over and over again, it just learns to memorize that particular iteration of your problem. And so by augmenting your images and feeding it a slightly different variation every time it sees an image, you get better results on images that it's never seen before.

And then on the training side, I mentioned we have all those export formats that go to TensorFlow, PyTorch, the cloud AutoML tools or our training, one-click training platform. And we’re adding support for more and more as time goes on. Our hope is to basically be that connector that connects every labeling tool with every training tool. And so when customers come to us and they’re like, “Hey, I have this like random annotation format from a Chinese paper that was published in 2012. Do you support that?” The answer is always yes. And we just spend an hour adding support for that before they get on-boarded. And our hope is to support every single format and every single training platform.

And then on the share side, this is one of the big pain points that we felt when we were building our own apps, is that it feels like the olden days like before Dropbox where you would be like emailing around like these version 2.final.reallyfinal, and you have like multiple people working on these datasets. And let's say it in the olden days before Roboflow, I took 20 chess images. I emailed a link on Dropbox to my cofounder. He combined that with his 20 images that he took. Then he found a problem with one of my images and he updated it. Well, now you have like three different versions of the dataset and it’s like not entirely clear which one you should use or how you should be working together on that. And so Roboflow is like the single source of truth

for your datasets by combining them in this platform that's like a multiuser sort of thing. You can religious keep track of who's done what. What are the different versions? Who trained which models on which versions? And make sure that you're like staying in sync rather than having a bunch of different versions floating around out there that like some of them are cropped and some of them are resized. You really just want like your original files and then transform for your models in a non-destructive manner so you can experiment without like getting completely lost in all the data.

[00:29:49] JM: And again, the process of preparing datasets for training. Let's go a little bit deeper into that. So the different things that Roboflow is going to do is assess annotation quality, fix unbalanced classes, de-duplicate images, visualize model inputs and version control datasets. And then you can share them with your teammates. Tell me more about the preparation for dataset training.

[00:30:14] B: Yeah. So as I mentioned, like not only are all the labeling tools using different formats, but the training tools are all using different formats as well. And most of the time they don't match up with any of the labeling tools. So for TensorFlow, you have to create what's called a TF record, which is like a binary format that has all of your images and all of your annotations compiled into one file that it's going to load at training time to go through and create a data loader and like iterate through all of your different images. That's something that traditionally you'd have to write your own Python script to take all of your images from disk, pair them with your annotations, encode them in the specific format and then output this TF record file that's going to go through TensorFlow to do training.

And so like you can imagine, that's something that there are countless stack overflow questions about like how do I convert this format into a TF record for training with the TensorFlow object detection API. And so with Roboflow, it's just a click of a button. When you click export, you get a drop-down list, and one of the options is create a TF record, and then it will compile those altogether and it will either let you download that zip file to your computer or give you a link to that, host it in the cloud so that when you spin up your cloud server or boot up your co-lab notebook you just drop in that one line of code and it will downloaded it from the cloud, unzip it, and it will be ready for training with your model.

[00:31:38] JM: What problems do you think machine learning is uniquely positioned to solve in the next year, or 5 years, or 20 years?

[00:31:46] B: Yeah. So that's actually pretty interesting. I think one of the like opinionated stances that we take is that computer vision is actually its own unique beast. It's certainly a part of machine learning. But we think that the tooling and solutions that are needed for computer vision are actually much different than the ones that are used for, say, natural language processing. And so I think like when you think about like what is machine learning going to do, it's such a broad answer. And I think like focusing in on what is computer vision going to do is probably the part that I'm most suited to answer. And I think – like if I think in 20 to 30 years down the road, I think that the state that we're in right now with computer vision is similar to like maybe how the web was in the 90s where certainly there were like e-commerce websites in the 90s, right? But in order to build them, you had to you like invent your own database and create your own web server. And if you wanted to like accept payments, you had to like be an expert in cryptography to be able to do that. And I think as we go forward, all those things are going to be like abstracted and made into tools that basically any software developer off the street can pick up and use.

And so we think that that's kind of our mission. And when you do that, you enable all of these new use cases. And so if you think about like what computer vision has done to the car industry with self-driving cars, it's just like this massive transformation that not only changes how cars operate, but also like how cities are going to be organized. And our kind of core hypothesis is that computer vision isn't just about self-driving cars. It's kind of like the pc or the internet where it's going to touch every industry and transform every industry.

And so if you look at kind of some of the use cases that are coming down the pipeline, I mentioned detecting oil leaks, but that's just the start. We have a student that is working on detecting wildfires from computer vision. So you can imagine having these like security cameras on top of weather stations that are looking for smoke. And he wants to deploy a drone at the first sign of smoke to like douse the fire before it gets out of control. Or we have like these other students that are doing human rights monitoring. So there's this tribe in Africa called the Maasai people that the government is burning their villages, and he wanted to track like their migration. And so he's using satellite imagery with computer vision to find where the campsites were,

where they are now, and kind of track how this tribe is being displaced.

And we have like companies that are building their entire company on top of Roboflow with computer vision. One of those was a Y Combinator company that is building a pill counting app. And they're replacing this old $15,000 machine with something that runs on commodity hardware. And so it's going to basically like make this accessible to all these small pharmacies and just like make their job so much easier.

And so I think like when you when you think into future, it's like every app or company is going to be able to use computer vision without it having to be a core competency and without having to hire a bunch of PHDs. And that's really exciting. It's like a future that I want to live in. And like if I could take a time machine and like travel to the future and see how amazing things are going be once developers have access to all this technology, it's totally something that I'd be intrigued by an interested in doing.

[00:34:58] JM: As a company goes from a test model to a production level model, what are the considerations they must take regarding datasets and dataset pipelines?

[00:35:09] B: Yeah. I mean, I think one of the biggest kind of paradigm shifts for developers getting into machine learning and computer vision for the first time is that it's not like a binary sort of thing. It's not like your machine learning model works or it doesn't. There's like these gradients of how well it works. And so, like traditionally, in software like you can just write a test and be like, "Yes, my code works. It does exactly what it's supposed to do. But with machine learning, it's not entirely clear like when you're done. And in fact you may never be done.

And so it's like there's this iteration cycle where you want to get something that works well enough for your first version, deploy that and then find all of the kind of edge cases where it's failing, and then pull that information back from your production model that like the things that it's not confident about or that a user reports are incorrect and then pull it back into the beginning of the flow and put that back into your dataset to make it more complete. You train another model and you go through this iteration process where you deploy it and then you see, "Okay. Well, what's it still messing up?" Bring that back. And like, over time, your model gets better and better. But you really have to close the loop.

And so I think like the workflow and the cycle is such that you need to like figure out what is my MVP? And like make sure that you're picking a problem where like you can actually deploy like a first version of the model and it's not going to cause like a car to run people over. Such that you can then figure out what are the edge cases and go back and keep iterating and making that model better and better every time. I think that for a lot of software developers, that's kind of a new paradigm where like you're shipping something out there that you know is only going to work 80% of the time and then thinking about how you design your software around that knowing that 20% of the time it's not going to work well. And I think that that's not only a software problem, but also like a design problem and a business problem and a strategy problem that needs to be solved. And it's kind of a new frontier for folks. And it's been interesting to kind of discuss that with people and have this skepticism about like what you mean my software is not going to be rock solid and work 100% of the time? It's just like, "What's a probabilistic thing?" There are definitely use cases where having it be right 80% of the time is better than not trying at all. And so, yeah, it's kind of like an exercise in defining the right problem where you're going to be successful rather than need something that you're going to have to spend a decade on bulletproofing before you release the first version.

[00:37:35] JM: Are there some other common issues in dataset management that you've seen lead to poor model performance?

[00:37:41] B: Yeah. So there are a few. So one of the most common things that people run into is trying to detect tiny objects in their images. So you can imagine like let's say you're trying to train something on satellite data and you're trying to detect like people on a beach. Well, the resolution is such that those people only are a few pixels in your image. And the way that machine learning models commonly work is they have like an input size. And so even if your images is, let's say, 20 megapixels, it's going to get shrunk down to like 800 x 800 or 416 x 416. And those like small number of pixels in the big image end up like becoming one pixel or less in that shrunk down image. And so this is kind of like one of those things where it's like an implementation detail that if you're not a machine learning expert, you're just a software developer that's kind of like following a tutorial, you might not have this mental model for like what's actually going on behind-the-scenes.

And so if you have like small objects inside of your images, it turns out that a lot of times your model just can't detect those things, because when it gets processed by the model, it gets shrunk down so far that there's just not any information for it to detect. So that's one thing, is like detecting. If your objects are small relative to the size of your input images, you have to do some things to account for that. So one of those might be like tiling your images where instead of running your model one time on this 20 megapixel image, you shrink it down and cut it into 10 x 10. And now all the sudden you have 100 different images that you run through your model, but each one, like the relative size of the object compared to the size of the image is bigger. And so it gives the model more pixels to work with.

[00:39:26] JM: What are the other parts of the machine learning or computer vision process that could be automated?

[00:39:32] B: I think that the training and deployment is an interesting area where it's not clear that you actually need a custom model for each problem. There is plenty of models that are very good off-the-shelf that you can – It's called fine tuning them. So you can like take existing weights that are trained on a large dataset like COCO, which is one that was released that has millions of images that represent a whole bunch of different things. And the model can represent these generic things. You start from that base and then you train it to like learn your individual objects. So if it's chess pieces, the COCO dataset doesn't know anything about chess pieces, but you can start from, "Oh, it knows how to like identify dogs and cats." And like the features of dogs and cats and like curves and like changes in patterns and those sorts of things are applicable also to like isolating chess pieces on top of a chessboard. And it turns out that just by fine-tuning this model, you can get pretty good results without actually changing the architecture of the model. It's just changing the weights.

And so you have this kind of – Like you can think of the model as like this meta-program that can learn a whole bunch of different domains. And so I think over the last decade, a lot of work has gone into like optimizing your model architecture. But I think we're getting to the point where these models are good enough for a lot of problems where you don't need to do a whole bunch of like core research on the model architecture. You just need to retrain them on something else. And once you do that, like you can get to these solutions where it doesn't actually need human intervention. You can just like run through the same process. Get new weights, and it works

pretty well.

And so I think doing that and then deploying it is one area where – Certainly, for some problems, you're going to need to do some core R and D. But for a lot of problems, you can just kind of automate that process and get something deployed that works pretty well.

[00:41:30] JM: Any other predictions about the future of computer vision?

[00:41:34] B: I think augmented reality is one that I think people have written off at this point, because like the early kind of example applications that people have come out with yet so far have been pretty underwhelming. And I think when you combine augmented reality with computer vision, it enables you to do really interesting things. And so when you hear Tim Cook saying that he thinks that AR is going to be like the follow-up to the world's most successful product in the iPhone, I think people roll their eyes. They're like, "AR is just this gimmicky thing that lets you like put Pokémon on the street."

But, really, when you combine it with computer vision, it allows you to put like a software overlay over the top with the real world, which I think is really interesting and thinking about like taking real-world objects and enhancing them with software for the first time without like embedding a computer in the thing to make it smart. You just make it smart by adding a software layer that understands what it's looking at and can add features to it is something that is going to take people by surprise. And so I think I would not write-off AR just because the first version of it was pretty underwhelming. I think that there's a huge like greenfield of opportunity there.

[00:42:44] JM: Okay. Well, thanks for coming on the show. It's been great talking to you.

[00:42:47] B: Yeah, likewise. Thanks for having me.

[END]