EPISODE 1430

[INTRODUCTION]

[00:00:00] JM: Automating video search requires a data pipeline that extracts metadata from videos and allows users to annotate the video with information that correlates to that metadata. The video needs to be segmented into intervals that define the search space, and the search space needs to be queryable by a user. Sieve Data is an automated video search platform that enables developers to add video search and analytics to internal tools and applications. Mokshith Voodarla is the CEO of Sieve Data and joins the show to talk about video search and what his team has built.

[INTERVIEW]

[00:00:28] KP: Mokshith, welcome to the show.

[00:00:31] MV: Yeah, thanks for having me, Jeff. How are you doing?

[00:00:34] JM: Doing great. You work on Sieve Data, and that is a video search platform. And video search is kind of complicated, because it's not as simple as, for example, an audio search or, if you think about audio search, I've got a recorded piece of audio, I can easily just turn that into text and issue text queries against it. But a video, you would have to – I'm thinking of like a scene in a movie, all kinds of things can happen across that scene and being able to search across all the different scenes in a movie would require lots of annotations, and lots of other things like that. So, give me an overview for what video search even means.

[00:01:23] MV: Yeah, so I mean, I think before we even talk about video search, aa good question to ask is what's even hard about a video specifically? I mean, you touched upon a few of those things, right? But I think the first thing is density, right? How big video is. The number of frames, the length, size, it all accumulates super quickly. The single person can't really watch footage across many cameras in real time, and it doesn't make sense to store a little bit raw, forever. So, those are some of the hard things there. And you also touched upon this idea that

depending on the video, basically different things actually matter about what's happening based on understanding a scene, or whatever it is.

So, when we talk about video search, we're talking about basically extracting the information that you care about in video, and making that searchable, basically, across variables that you know. Typically, YouTube, or some other entertainment platform, you want to store the video raw, because you actually just want to consume it. But a lot of the time with things like monitoring, you don't really care about the raw video, you care about what's actually happening in it.

[00:02:33] JM: Can you say more about that?

[00:02:35] MV: Yeah. So, I mean, maybe we'll take the example of like security footage. Security footage, few basic things you might care about. Maybe the people, maybe if it's like a parking lot, or something, the vehicles, and sort of maybe some malicious activity that happens. That's what you actually care about in video. You don't care about the raw footage that ends up being stored. But the issue today is that people end up storing raw footage, because it's really hard to actually extract these insights, effectively, sort of across scale. And it's much easier to just store it and have people watch. That's an example in security, but you might imagine in construction, it's different. Maybe in construction, people care about unsafe things workers are doing or when certain equipment is moving, and things like that. So, depending on the use case, if you have a camera monitoring something, you care about different things.

[00:03:30] JM: So, it's more like the application of video search that we're talking about here is customizable video search.

[00:03:41] MV: Right. Something that depends on what video you even have.

[00:03:47] JM: Got it. So, if I'm developing, I guess, like a custom security video scanning system, I assume I would have to have some kind of labeled data set around what constitutes a violation of a secure environment, right? I mean, it would have to have, or maybe a level data set around what is not problematic. And then any deviation from that I would want to have a response to, I guess, just give a deeper example of an application or a use case of Sieve Data.

[00:04:25] MV: Sure. So, I mean, I'll sort of break it down. Basically, the idea is that we have these sorts of camera-based analytics applications that are going to be everywhere. You're already sort of seeing this and sort of monitoring. There are tons of companies in security, or retail or supply chain factories, all trying to do different things with basically monitoring something and giving people what they care about basically, either. It's some unsafe activity or some anomalous event, or just monitoring shopping activity or something.

So, we basically think there are tons of companies building specific software for these spaces, be it dashboards for security or there's actually a $3 billion company called Equipment Share that builds cloud software for construction. All these companies are basically solving these industry specific problems and would eventually benefit from say something like being able to process tons of video and add those insights to their platform. But right now, there's no easy way to do that. And so, the idea here isn't to build a tool for say, ML developers, to sort of build new models or anything, rather, it's to provide sort of a holistic solution from the processing of the video all the way to actually extracting the information that's relevant to you, sort of end to end as quick as possible.

So, what that involves is obviously, infrastructure that processes video very efficiently. But it also involves building good models and something you touched upon there. One way we can do that is basically offer a model zoo, and sort of fine tune that customer to customer. But also, other features we've been thinking about, like reverse image search or similarity search by say, allowing the customer to draw a box around, say, they care about this blue ball in the scene as an example. And then us being able to surface that same sort of thing that's visually similar for every piece of video that they keep sending through our platform. So, it's sort of supposed to be a plug and play solution for companies that may not want to invest time into building their own AI teams or their own video teams.

[00:06:39] JM: So, let's say I've got a video stream of a police station or the outside of a police station, I want to create some kind of custom system for detecting, I don't know, when a new prisoner is coming in or when something is going wrong outside of the police station. Give me an overview for how I might use Sieve to do that, and why that's easier than just building ad hoc video analytic infrastructure?

[00:07:13] MV: Yeah, sure. So, there are two parts to this. Number one, the actual problems with video. And then number two, the actual problems with extracting information from that video. What I mean is, say you have this camera setup across a set of police station, if this camera is recording at 30 frames per second, within a week, you have about 20, 25 million frames, individual frames of images to sort of process. And if it's coming in at that rate, it's sort of really hard to manage and scale or even understand, should I be running a model on every from of video? Probably not, that's probably not the most efficient way to do it. It's really expensive for you as well. And you're going to have to build infrastructure that basically either does something less complex, either, say, looking at every 10 frames of video, or something, or just figure out how to deal with sort of the scale of video coming in with whatever cloud infrastructure you have. If you scale this up across even 10, 50 cameras, it becomes really huge, right?

So, that's one problem you're going to face. The second problem is, as you said, like, okay, say you want to detect something bad happening outside the prison. You get this example of like a prisoner walking out. So, in terms of how you use Sieve, you basically plug into our API, and you can configure a project. We already have some off the shelf models for different things like people detection, et cetera, and different things of sort of being able to detect prominent colors across different boxes and things. So, you could say, if you know, you want to define a prisoner as a person wearing orange, you can do that with Sieve, completely automatically using the API. And you don't have to worry about, say scaling your video processing. Because like one thing we say is we can process 24 hours of video in under 10 minutes. That's because of all this parallel processing, we do, interpolation, all these other sort of infrastructure gains, we're able to sort of do that you don't have to implement yourself.

So, that's like a simple example of how you might use Sieve. Another example could be if you have your own models, you could actually plug that into our infrastructure as well, if you don't want to use our own models. It's just we offer models because companies sometimes want to get off the ground as quick as possible, and so we think we're basically the quickest way for anyone to do that.

[00:09:36] JM: How much of this is like, are you able to build the platform, mostly off of high-level cloud services like that already exists? I think you've talked about people detection and stuff that's built into your models zoo? Are those just off the shelf models you can take? I guess I'd like to know like how much of this stuff you've had to write from scratch and how much of it was kind of building a good UI and a good user experience over some existing API systems.

[00:10:08] MV: So, there are sort of two parts to the system with the model zoo. Obviously, a part of it is sure, making some off the shelf models sort of available using our API automatically. But there was a lot of work that went into also building custom models that work for, say, different scenarios. For example, we have models that are really great at object detection, and sort of more close-up scenarios versus top down views. And those are things that are much more easily sort of fine tuned to customer data, depending on what their application is. And so, there was a combination of off the shelf work and sort of, like more custom modeling work that went into trying to build a model zoo that's modular and sort of, can be ramped up for a specific customer really quickly. There's also a lot of features we've been working on, like I mentioned, sort of reverse image search, where across video, you can basically try to draw a bounding box over something you care about. And then we'll automatically surface things that look similar to it in the future, as you sort of push more video to our platform. And that means you don't have to build a specific model every time you care about something new. That allows you to basically leverage that feature. And the more boxes you sort of draw around things that are similar, the more accurate we get at surfacing things that are similar.

So, that's sort of one of those things. But in terms of like, the actual infrastructure for processing video, most of it is built on top of serverless. So, you can imagine sort of AWS Lambda, and sort of things like that, along with obviously GPU instances and things that we used to run models. But we think that abstraction gets pretty complex, if you want to support really any scale. And you also want the cost breakdown to actually work out. Video is notoriously expensive to process. And for us, if we want to basically meet our margins, we have to lower our costs as much as possible. And so, it's in our best interest to basically do as much optimization as we can there to lower our costs, while also making it sort of a scalable solution for say, a customer that comes in and says once you use this, this video video search in perpetuity, right? So yeah, does that make sense?

[00:12:23] JM: Yeah, absolutely. Can you give a little bit more description of the ingest pipeline, and then we can talk about what actually happens after a video gets ingested?

[00:12:35] MV: Sure. So, the way our sort of pipeline works is video is ingested, you basically send a secured or signed URL to our platform. And we ingest that video, we split it up into a bunch of chunks, so sort of fixed interval chunks, and that allows us to basically parallely process different parts of the video. And then we actually have sort of a layer of filters. And sort of what I mean here is, depending on the application, you might care about different things, motion could be an interesting sort of filter on top of video that might help you get rid of, say 90% of places where nothing is happening. And so, we have these layers of filters that we run, figure out which parts of video are sort of potentially more important or more relevant to actually process. And then as soon as we sort of have these specific intervals of video that pass through multiple layers of filters, depending on what we configure for the project, we can run our most expensive models on just the ones that have passed, and then sort of tried to interpolate the results across say, places where there was no movement, for example. Or we can sort of make an educated guess, on what actually happened there based on say, surrounding brains. And so, that's sort of how we think about our pipeline right now, and sort of like how video actually gets ingested and processed at the high level.

[00:14:03] JM: So, once the video stream is being ingested, give me a sense of like, how human detection, for example, runs across that video stream?

[00:14:22] MV: How human detection runs across that video stream?

[00:14:25] JM: Yeah, for example, if I'm trying to, you know, detect all the frames with humans in them or segments of a frame, how is the detection of those frames occurring? How is the video stream processed? And yeah, I guess just give me a sense of how that code is executed.

[00:14:44] MV: Yeah, sure. So, I mean, if it's something as simple as sort of person detection, and that's all you care about, it will basically do a few things. Say, with the people, you can make an educated guess that if that's the scenario you care about, you probably most likely – you care about things where motion is occurring to some extent, where motion exists. And so, maybe we'll configure a project that has a motion filter. We sort of ingest this video, it gets

chunked up into these different sorts of intervals, and then each of these intervals sort of individually runs through a very cheap sort of motion filter, something that can basically flag, whether there's any sort of relevant motion or not. Say, there are specific intervals where there are. Most typically, maybe it's like, if it's some typical security footage, maybe only 5%, 10% of the video might actually we have any sort of motion going on. And so, 10% of the video makes it through.

And then we're basically running a people detection model on these different intervals. So, each interval, we sort of run this model on on each frame, depending on configurations, we can skip frames and then interpolate in between, but we basically run a people detection model on each interval. Say there's gaps of video, obviously, 90% of the video didn't make it through. So, we only ran this model on 5%, 10% of the video. Now, we're basically able to pinpoint say, the middle section of each of these gaps, run a model on that, and then sort of interpolate those results to fill in the entirety of that gap. So, that basically is like a very simple way you might do more efficient, say, people detection on video and sort of save costs on your end.

The complexity comes when you're interested in other types of attributes that might change more gradually, for example. Say you care about how blurry this video is over time to make sure your camera isn't getting fogged up or something, or what the lighting is like. Those sorts of processes are a little more intricate, right? Because the change happens a lot more gradually, and it's not a simple motion filter that you can run. So, there's different types of filters, basically, that we have to configure depending on the types of attributes people are interested in, to basically efficiently process that over over video.

[00:17:18] JM: Again, so the processing of the human detection algorithm against a video stream, this code you've taken off the shelf, or you wrote custom human detection for it?

[00:17:36] MV: For human detection, I mean, we have tons of custom models, internally. Obviously, there is an off the shelf offering, if you want, but that's pretty simple to sort of integrate. But depending on different scenarios, we have tons of different types of human detection models basically ready to go for people to use.

[00:17:53] JM: Okay, and if I issue a query against the video stream, can you give me an understanding of how that query actually is processed? Because I think of if you want to map a text query to a video stream query, there has to be some kind of enrichment or transformation from text to image.

[00:18:20] MV: So, if you are querying for say something as simple as I want all the intervals of video where there's more than two people. If it's a query like that, when we process video, we are extracting that information, saving that piece of metadata as a variable in sort of our database, and then making a query you can imagine sort of simple, no SQL databases and things, right? Except here, we store it in sort of context of intervals, rather than, say, each individual frame. So, the query is quicker, obviously.

But it's purely based off of individual metadata. When it comes to sort of the similarity search feature, though, which is the more powerful thing, sort of more robust, where you don't really have to define a specific model. You can just search across whatever similarity. It's a matter of basically figuring out different parts of the video where we can basically embed the images into sort of some lower dimensional space. So, these are deep learning models for context, typically, they might have encoder like, there's a sort of encoder, decoder architecture, that's pretty popular. Basically, translating an image into some sort of lower dimensional representation, which you can use to say compare similarity between two different images. Typically, people might say use an image net model or something or a ResNet model, run an image through and then use that to sort of compare how similar two images are to each other.

So, over video, we basically have to figure out which frames does it make sense to store these embeddings for. And then every time we surface a new video, we basically compare the embedding that's averaged up from, say, this previous content that we've stored to this new video frame that we think we should compare against live, and then basically store that result. So, when you're actually doing the queries, there's no processing happening. It's all stuff that's been stored. All the processing is happening as sort of videos ingested and run through our pipeline.

[00:20:36] JM: So, if I understand correctly, for a given video stream, you build essentially like a time series of the images that are detected across that video stream? Is that correct?

[00:20:55] MV: Could you sort of elaborate on that?

[00:20:57] JM: So, the way I'm imagining it is like you have a video stream, which is just images over time. And at a given interval, you're detecting essentially images or the presence of models in those given segments, and then those become searchable as they're detected by the video stream detection system?

[00:21:22] MV: Yeah, so the the we're basically saving this stuff is by saving per video, the intervals where we think there is no difference and sort of the "metadata" that exists there. So, say we detect an interval where there's like, one person, and it's blurry. Then for that interval, we basically store the start frame, the end frame, and then the actual attributes for that interval. And then the time we basically append to this database is when we detect even a single change in any of those variables. And so, when when you query, you basically have this much simpler way. There's very like processing you have to do, but the database doesn't sort of scale up super quickly, if you have a bunch of duplicate information stored frame to frame rather, it's being stored, basically, interval to interval.

[00:22:16] JM: Okay, so each interval, you're trying to detect the presence of one of these models that you might be looking for, correct?

[00:22:26] MV: By model, you mean sort of –

[00:22:29] JM: Yeah, getting more of a sense of like, for a given interval, what is being analyzed, and like what you're looking for in a given interval?

[00:22:38] MV: Yeah, I sort of explained the layer filters earlier, the sort of set of filters that you run a model, run a video through. And so basically, when we save these intervals in our database, along with what each interval contains, we're basically looking for changes in at least a single variable. So, say, we're trying to detect people, vehicles, lighting, weather, blur, those are five things we care about. And even if one of those variables changes in any way, we basically create sort of a new append to our database, and then store what that change was. So, over time, we basically have – we're storing less actual things in our database, because

we're not storing duplicated information. But people are still able to sort of query in the same way.

So, that's sort of how we actually store the intervals. In terms of what you're looking for an interval, obviously, that depends on the use case, right? We have small model zoo, and you can basically pick things there, or you can choose to plug in your own models that we basically selectively run on the frames, we think we should run them on based on our infrastructure. So, that's the definition of what you're looking for in each of these frames.

[00:24:05] JM: Okay, and so if I want to define my own model, what am I doing to create that?

[00:24:16] MV: Yeah, so that can mean two things. When you say define your own model, that could either mean you care about some new variable, and you don't actually have a model for it yet. In which, what you could do is basically use the similarity search feature I was mentioning, to draw either a box around a certain frame or sort of tracking object across an interval, and then ask us to surface every time that comes up in the future. And we basically use sort of this embedding thing that I was mentioning earlier, to make comparisons between different frames and intervals, to surface things that are similar in the future. So, that's one way you could say, “define” a new model automatically, right?

But another way is if obviously you have a team that's built a model, say it's an unsafe workplace activity detection model. So, you're detecting whatever, say a construction worker is doing anything unsafe. It's very domain specific. It's possible you have a team actually working on this internally. What you can do then is define an endpoint, deploy your model at your endpoint, and then submit that endpoint to us. And so, that endpoint is what's then called by our platform, and we basically expect a certain type of input and output format. But you can use our platform in the exact same way, the difference is that your model is not caught as often. So, it costs you less, because we're not actually running it on every single frame.

[00:25:46] JM: So, in the case of like the home-built model, you would have as the video stream is being processed, like my own hosted model would be pinged for each interval?

[00:26:04] MV: So, when you say interval, the interval might not be the right term there, right? It's like, it's pinged every time our infrastructure decides it should pick the model. And that definition is based on obviously, frames passing the layer of filters, and then the definition of how many frames we decide to skip, or based on when we actually decide to run the model and say, in the middle of intervals, where nothing is happening, for example, or on every frame in an interval where things are actually happening. Yeah, that's the idea there.

[00:26:38] JM: And the granularity of those like how many frames go into an interval, that's configurable by the user?

[00:26:47] MV: Yes, I mean, we're basically able to process at up to 30 FPS granularity, meaning, by default, if you use our platform, we basically generate metadata for video running at 30 FPS. If you submit video that's higher FPS to us, we automatically skip frames, obviously. But typically, most applications we've seen, no one wants anything more than 30 FPS. But it is configurable, if you want less than that.

[00:27:19] JM: So, can you walk me through an actual customer use case? And I guess, also give me a sense of the infrastructure, that customer use case would be hitting on your site? I'd kind of like to get an end to end picture into what like cloud services and stuff you're using from kind of a top down perspective.

[00:27:45] MV: Yeah, sure. So, let's take the customer use case of this customer building a dashboard for factories or industrial environments to sort of monitor their facilities. So, they're building a really, really awesome dashboard that helps this sort of plant owner or whoever just monitor everything going on. And historically, they've built a product that basically just hooks up to say some IP cameras, and maybe you can share links. It sort of allows you to just basically sift through the content with a nice UI. And in terms of how this company might use our product, they can basically configure a project, configure a sub project, and if it's maybe an industrial environment, say they care about people and vehicle detection. And in terms of how they use our platform, they make an API call, they basically, every time they have video pushed up to the cloud.

So obviously, in a cloud sort of a factory setting or something, there might be bandwidth issues. And so, you're not pushing video to the cloud 24/7. Maybe, the way it works is that you have these cameras running. And locally, you might have some sort of NVR, or some sort of recorder at the factory. So, every time there might be some form of motion detected or it could be if the factory owner is like, “Oh, I want to bring up all the footage from last week to the cloud so I can share with my employees that are remote somewhere or something”, that data gets pushed up to the cloud. And as soon as that's the case, it might be in some bucket. So, imagine some S3 bucket that has just tons of videos.

Now, what what a person does use our platform is create a secure signed URL for that S3 object, and then submit that to us. And so, that gives us permission to view the object temporarily, and only us, and so that gets pushed to our platform and we run that whole ingestion pipeline I mentioned. It goes to the whole layer of filters, and all the metadata gets saved in sort of our database. And now, when integrating that into your sort of dashboard product as a user, what you'll do is, say have a feature for your customers to search for every instance of a person. So, say, the UI has something like, oh, a drop down menu, vehicle or person, and then gives you an option of timestamp, you want to find all the vehicles have been people between 5 PM and 6 PM, on this day of the week, at this specific camera, in the factory.

Basically, that exact query can be submitted to us. And then what the platform gets back is intervals of video that match that query. And so, they can basically pull up the content on their end, and then surface that to the user because of the exact frame numbers that they received from our API for that query without having to worry about video ingestion, or processing, or running these models or anything. So, it's sort of this end to end, sort of a full stack dev can basically use our API.

[00:30:59] JM: You just touched a little bit on the on the cloud infrastructure there. So, the video hits an S3 bucket, and then, I know, you've got a fully serverless system, so you just like, have a bunch of lambda functions that are processing these video streams?

[00:31:20] MV: I mean, the lambda functions are things that could be running the filters and sort of the interpolation and things like that, that I mentioned. Obviously, the actual models aren't running in lambda functions. The actual models are probably running in GPU instances,

separately, that we have separately deployed that the lambda functions are able to call. But yeah, like the actual sort of core processing infrastructure itself, that's not deep learning or anything is running in tons of lambda functions that are spitting up.

[00:31:51] JM: What's the the level of parallelism that you have on one video stream? How many lambda functions are we talking about?

[00:32:00] MV: So, that's actually completely configurable. So, we want each lambda function to say a process a maximum of a thousand frames. We could just set that arbitrarily. And off of even A, minute a video that might be, I don't know, 10 minutes long, you could be sort of up in the thousands in terms of lambda functions that are getting spun up, sort of instantly to sort of process that video. And so, the sort of beauty of all this is that, based on that defined interval length, regardless of how long your video is. The actual process takes the same amount of time. And so, it doesn't matter how long the video you submit to us is.

[00:32:47] JM: Can you tell me about some of the most acute engineering difficulties in building Sieve? I'd love to know about just what's been most hard to build?

[00:32:58] MV: Yeah, so I mean, I think a few things sort of come top of mind. I think the first thing is the beauty and sort of the limitations that come up with lambda functions, especially when you are trying to spin up as many as we are, as quickly as we are, there are a whole bunch of weird errors you run into, especially with even concurrent connections to a no SQL database and things like that. And so, there's a lot of small tricks and things that that we've sort of encountered and weird bugs, basically that we've had to resolve, because of sort of the visibility to do serverless.

But that's sort of one thing. I think another thing is, this whole idea that for us, we save on costs, if we lower the amount of times we call a deep learning model. So, figuring out basically, how to lower that as much as possible, and thinking about creative ways you can sort of cost there depending on the application, and even making it application specific. That in itself was also sort of an interesting problem. I mentioned this whole idea of visual search, similarity search on tons of videos. If you're trying to run and generate and then compare embeddings off of streams of

video coming in, it's actually a very tricky if that list of embedding you're comparing against gets really long.

So, save the list. This embedding could be a list of a thousand numbers that are basically a representation of what was in that image. Saving that and basically, building efficient ways to sort of compare that against a longer list of these “alerts” people created in the past is sort of an interesting infrastructural problem that we've had to solve and there's actually a lot of open source work on this. There's a project called MILVUS on GitHub, which people can take a look at, which is sort of this embedding search, sort of optimized embedding search sort of thing for images. But using parts of these types of open source projects and things, were sort of a big part about solving some of these problems.

[00:35:24] JM: So, can you tell me more about like, what kind of data structures do you use to hold those like large amounts of embeddings and perform operations on them?

[00:35:35] MV: So, the only operations we need to perform on them are just comparison operations, which are basically just subtraction, and sort of like computing some mean squared error, right? So, when we think about, every time a user creates an alert, or like draws a box, say around these things they care about, there's a few things we do. Obviously, we store this raw list of numbers, then every time say, a user adds a new box to this “alert”, we have to run for an embedding on that sort of piece of content. And then obviously, we'll just do some sort of averaging to keep that as one embedding, that we're storing for that “alert”.

So, data structures we use for that, like typical sort of lists, HashMaps, and things depending on the specific type of operation we're trying to do. But it's nothing specifically fancy or something compared to – I mean, obviously, we're using tons of stuff from these open source projects, which, under the hood, definitely have more complexity, but sort of abstracted away from us.

[00:36:52] JM: When you compare what you're doing with Sieve, to other visual detection platforms, like, I think, like Clarify, what's the product differentiation? Or how are you trying to just separate yourself?

[00:37:11] MV: Yeah, so I think the thing about platforms like Clarify, are that, yeah, they have a model zoo. You can run on a model zoo. You could process tons of images that you have. There are a few differentiators for us. Number one, is our focus on video, and sort of the unique applications that has in sort of these industry specific solutions. And so, when we're thinking about who our target customers are, it isn't just some arbitrary company, working with tons of like, image or image or video data, right? It's rather these companies that are basically building industry specific software, for, I mentioned a bunch of things, farming or agriculture, retail, security, and basically giving them a holistic solution of not only just running the model and being like, “Okay, image in, JSON out”, we're also offering them a way to search that over time. What matters with sort of – you'll see this with Clarify, there's tons of other models as a service platform. With even GCP, AWS, they have their own models as a service sort of offerings. We're not models as a service. Rather, we're a platform that takes into context, the fact that people do care about things like visual search, visual similarity search, based on past content that has been in the platform, and people will actually do care about making queries on data that has been sort of historically stored, and doing that efficiently across metadata. But also, as I said, sort of visual similarity, right? So, it's more of a holistic solution for these kinds of companies. It's not just, “Okay, you want to model, we'll give you a JSON output of what was an image.” That isn't really solving the problem for these people. Does that sort of make sense?

[00:39:03] JM: Yeah, it does. And I think like the – I mean, Clarify, there are a lot of products that they're focused on. So, probably the narrowness of focusing on video search is pretty beneficial. When you look at the product as it stands today, are there particular use cases that are still difficult to accommodate?

[00:39:31] MV: Yeah. I think the hard part, as I said sort of about video, one of the hard things is, depending on the video, different things matter. And so, we are right now trying to basically accommodate for obviously, really any application wherever you're monitoring any type of people. So, that sort of monitoring application, different sort of applications within factories and things and obviously, the thing that sort of limiting there is our model zoo, right? But that's where we sort of think about, okay, what is the core value of what we're providing? And how is that different for different people?

So, some people, our core value is this holistic solution, where it's the quickest way for you to basically build video search for your tool or platform, because we have taken care of the video and the rest and we have a model zoo. For other people, we're the most efficient way for them to process video on the cloud. They don't really care about using us for our model zoo, because they're building very specific models for some extremely domain specific application. Say, in some medical space, or some area where you don't really even have tons of access to the data. And so, you really do need a team specialized working on it, but they do have tons of data coming in. So, that's when they can use our platform with their models, right? It's sort of figuring out where our value prop is sort of the strongest, and figuring out which applications we can actually tune for and sort of what features we can build to basically make this the quickest way to search video, right? One of those features is not just the model zoo, but sort of the similarity search, but what else can we build on top of that, to make this apply to as many industries as possible, and give them the tools necessary without say, having to build whole AI teams internally?

[00:41:31] JM: It seems like the latency of processing video can be pretty problematic if you're trying to use it for real time applications. Are there any kind of aggressive processing systems you've built? Or can you comment on like programming languages or what you've done to kind of address that or process more aggressively?

[00:42:01] MV: So, I think the real time nature, of course, yeah, this real time nature of specific applications is really important. I think right now, we're sort of focused a lot more on the idea of bulk processing. And sort of, not necessarily the the before real time, if you want a result, say a second later. But the way our platform works right now, you could process video as fast as it takes to basically run one of those models with a little like sort of added maybe 10% latency or something, because of the parallel processing there. So, we could basically increase the parallelization by trying to spin up tons more lambda functions by basically lowering the amount of frames each one processes to lower latency as much as possible. Obviously, there's sort of this tradeoff between efficiency there versus the speed. And so, it might cost more obviously, to be spinning up even more lambda functions and doing less smart things with this interpolation and things. But we haven't really explored that to its full extent, when it comes to real time applications yet.

[00:43:09] JM: What programming languages do you use for the bulk of your – I guess, the bulk of the code is probably in the – just give me just give a breakdown of the programming languages and where you use them.

[00:43:22] MV: We use a combination of Python and C++. I mean, C++ is obviously if you're like, trying to get some performance gains on your models and things like, you could make that faster, especially on sort of the deep learning front. But yeah, those are the main languages we use.

[00:43:41] JM: Okay, so Python is mainly for the API's and the C++ is for the actual model development?

[00:43:49] MV: Right.

[00:43:50] JM: Gotcha. What machine learning frameworks do you use?

[00:43:54] MV: Mostly PyTorch.

[00:43:56] JM: Okay. Give me a sense of where the product is going and what you expect to build the near future?

[00:44:04] MV: Yeah, I think there's sort of this huge, huge trend, I'm seeing with just how much rich data we're basically collecting and storing. Our databases were typically – they were built to be great at doing operations on text to an extent and numbers. And you can do different operations of aggregation and count, things like that with text numbers. When it comes to rich data, images, videos, robotics, LiDAR, radar, all these different sensors. What ends up happening is we have a database, we store the raw content somewhere in a bucket, and in the database, we just link it to that raw data store. And maybe we have some extra metadata, like timestamp, something like that.

But indexing this stuff is really, really, really important. So, we're starting to tackle with – we're obviously trying to tackle this with video first, because we see tons of applications in video. But we think the idea of building great indexes for rich data is going to be increasingly important in

the future. You're already seeing this with robotics being sort of a huge thing and growing really fast right now. But also, just tons more cameras, people want to use for intelligence rather than just monitoring or surveillance. And so, we basically think that our grand, grand vision is really trying to be the best way to index rich data, and that is the 10, 15, 20-year down the line vision for what we're trying to build. We're starting to sort of build towards that, obviously, with building great ways to index video. But that's sort of the idea of where we think things are going and how we want to basically play a role in it.

[00:46:02] JM: And lastly, given that you've spent time in a variety of machine learning places, and you probably have a sense of the cutting edge, what are the the most outstanding problems in computer vision that are still causing a lot of problems?

[00:46:23] MV: The most apparent things you might notice, obviously, you'll, notice this most in self-driving is tackling the tail end, right? Tackling all the cases that don't occur that often, that your dataset might not have. People basically are treating these models as black boxes that, okay, they'll generalize, but your datasets matter a lot. And tackling for the tail end in these applications where that tail end matters a lot, for example, in self-driving where you could kill a person, those are really challenging. I think that's sort of one issue.

But I also think a separate issue is this idea that the ecosystem is still very fragmented in terms of how even for simple applications, people can spin up a model and sort of deploy it in production, and have the necessary tools to build a production ready product with computer vision. So, I think there's two ends of the spectrum where it's like, number one for the power users, and for those use cases where every single prediction matters, and then sort of the separate use case in the on ramp for the 90% of people that don't really care about the tail end, but want to deploy something that's production ready, and they understand.

I think like, those obviously, there are tons of problems and research still, but I think there's basically a lag between research and industry. I don't know, it could be like a five-year, three to five-year lag between discovering something in research and then actually applying that, at a sort of wider scone industry. We still haven't figured that out with supervised learning. It's still a challenge. And so, I think these are like two of the sort of big problems people are thinking about that I think are important.

[00:48:13] JM: Cool. Well, Mokshith, thank you so much for coming to the show. It's been a real pleasure.

[00:48:17] MV: Yeah, thanks so much, Jeff.

[END]