EPISODE 636

[INTRODUCTION]

[0:00:00.3] JM: Video streaming platforms such as Netflix offer a convenient way to watch video content online. We're now able to watch our favorite TV shows, movies and content creators on a range of devices. However, buffering while watching the video can be a painful experience on mobile phones and tablets that use 4G or LTE. As streaming becomes available to a wider range of devices with varying bandwidth restrictions, different encodings of the video need to be created for different devices and these different encodings of the video can also compensate for different bandwidth situations.

To get the best quality viewing possible with the bandwidth available to whatever connection a user is on, there needs to be a balance between the resolution of the video, the bitrate and everything else that defines the data that the video consumes.

Mux is a company that builds video hosting and analytics. Today's guest, Ben Dodson is a data scientist at Mux and he built a system for optimizing the bitrate of videos through machine learning. In this episode we discuss video encoding and how Mux solved the problem of serving the highest quality video with the ideal bitrate.

[SPONSOR MESSAGE]

[0:01:27.9] JM: Azure Container Service simplifies the deployment, management and operations of Kubernetes. Eliminate the complicated planning and deployment of fully orchestrated containerized applications with Kubernetes.

You can quickly provision clusters to be up and running in no time, while simplifying your monitoring and cluster management through auto upgrades and a built-in operations console. Avoid being locked-in to any one vendor or resource. You can continue to work with the tools that you already know, so just helm and move applications to any Kubernetes deployment.

Integrate with your choice of container registry, including Azure container registry. Also, quickly and efficiently scale to maximize your resource utilization without having to take your applications offline. Isolate your application from infrastructure failures and transparently scale the underlying infrastructure to meet growing demands, all while increasing the security, reliability and availability of critical business workloads with Azure.

To learn more about Azure Container Service and other Azure services, as well as receive a free e-book by Brendan Burns, go to aka.ms/sedaily. Brendan Burns is the creator of Kubernetes and his e-book is about some of the distributed systems design lessons that he has learned building Kubernetes.

That e-book is available at aka.ms/sedaily.

[INTERVIEW]

[0:03:03.3] JM: Ben Dodson is a data scientist at Mux. Ben, welcome to Software Engineering Daily.

[0:03:07.3] BD: Thanks Jeff. It’s great to be here.

[0:03:09.0] JM: This conversation is going to be about a specific machine learning use case and I think it's going to be a great example for why machine learning is so broadly applicable, because this is an application of machine learning to improve another area of software engineering, which is video encoding. In order to get to the machine learning part, we need to start with that topic; the video encoding subject. What is video encoding?

[0:03:37.2] BD: Video encoding is basically when you take a source video and you decide to encode it at along, let's call a bitrate ladder. Usually the raw file of video file is huge. It can be hundreds of gigabytes and it doesn't make sense to stream that to the client, because they're just – you'd have rebuffering four hours essentially, right?

A process that many streaming sites do is something called transcoding, in which you encode the video down to lower bitrates, let's say like 1 megabyte, 2 megabytes, 3 megabyte streams in which your interconnection could actually handle that size a video. In order to do that, it goes through some compression algorithm. That's what you hear when people talk about H.264, or the AV1 is sort this new tech, compression technologies coming out in a few years. They're compressing down the video to a more manageable size and putting that in what's called a manifest for you to then choose the appropriate size file that your bandwidth internet connection can handle.

How you decide to do encoding, because there's a lot of different settings that go into encoding and how you decide to do encoding can have a big impact on the type of video quality that you see.

[0:04:52.9] JM: Encoding is a type of compression. We're not talking about a compression like a zip compression, where you can't read the file without decompressing it. This is a type of compression like MP3 compression, versus a WAV file. This a lossy format versus a lossless format, where you can actually play back the file; you can read the file, but it is a slightly degraded format. The reason you want to do that with video files, because video files are tremendously large and depending on what device the user is using, depending on what the connection the user has, depending maybe on the screen size, you want to use a different encoding mechanism that is customized for the user in the given user's context. Is that accurate?

[0:05:46.7] BD: Yes. Yeah.

[0:05:48.1] JM: Okay. With an audio file, my source audio gets compressed into either MP3 or WAV, or AIFF and these video files, it's getting compressed into two different files. It's not different file formats though, right? It's just different bit rates of the same format?

[0:06:07.4] BD: Right. It's all still the same format, it's just that the bitrate basically determines how much information gets processed for each frame. You have essentially a set number of bits that you can allocate, let's say to each frame. For high bit rates, you can put a lot of the details and the motion will be in there. Whereas, for the lower bitrates, you have to do more with less. Depending on how good the compression algorithm you're using, you might be able to get away with that down to a fairly low bit rates, but it also depends very much on sort the content of the video that you're looking at, which is where part encoding comes into play.

[0:06:47.5] JM: Right. We will get into that. If we're ignoring that for a second, if we're just talking about broad encoding mechanism, so if I'm streaming video from my home connection and maybe I've got a T1 connection and I've got super high bandwidth, high performance network connection and I want the highest quality video, versus maybe I'm sitting on my phone in the middle of nowhere and I've got a really spotty connection, but I still want to stream House of Cards, how many different compression options are there along that spectrum?

[0:07:30.0] BD: It really depends on who you're streaming it from basically, because a site that doesn't have a lot of experience in video, they might decide to just do a single encode for example, and say, “Okay, everyone's going to get this middle tier quality bitrate,” so people on very high connections will be able to watch the video with no rebuffering, but the quality might not be there. They don't want to make it too high, because then people on lower connections on their phones or something like that might experience too much rebuffering, right?

Then someone, maybe let's say like Netflix, or someone with a lot more video engineering resources, they might decide to do a much larger ladder, in which they generate essentially different versions of the video file for low connections to high connection. Someone on that T1 connection, there's a pretty close to lossless version, I would say like a 20-megabit of the video file in which it’s pristine quality and you will have no problem streaming it on your connection, versus someone on that device, let's say there's at the low-end of the bitrate ladder provided they have a 250 kilobit version that actually it looks pretty decent, because of the compression algorithm that they're using, and the quality might not be quite as good, but you'll be able to stream it without any rebuffering as well. It’s just maximizing the experience as far as things like the startup time, rebuffering quality, depending on the situation that your audience is in.

[0:09:06.7] JM: You've mentioned this term, ‘bitrate encoding ladder’ a few times. I think this means that for any given video title, you want a set of encodings for that video that can be delivered to target users, given a user's constraints. A ladder is an exploration of the trade-offs between bitrate and quality, I guess? Because if you have a higher bitrate, that means it's going to be a bigger file, but it's also going to be higher quality. If you have a lower bitrate, that's going to be lower quality, but it's not necessarily a linear trade-off scheme. Can you define the term bitrate encoding ladder a little bit more?

[0:09:56.2] BD: Yeah, sure. Basically, your description of it is pretty spot-on. It's basically there's these files called renditions essentially, in which we're taking the original source file and creating another additional versions of them. From pretty close to lossless down to lossy compression versions, I mean, which the quality goes down. That's why you see things like JPEG artifacts, right in low-quality video, because it has to do with not having enough bits for particular scenes and basically there's – you can't allocate enough bits to accurately encode what's going on on the screen. That's one quality really starts to go down when you don't have enough bits to fill out the details.

You can put in as many renditions really as you want in a bitrate ladder. The renditions are defined as the bitrate that you choose, so typically on the high end of a bitrate, ladder it might be something like 10 megabytes. The thing is no one really knows what bitrate ladder should ideally look like. The reason for that is because it changes really depending on the content.

Some people use 10 different steps in the bitrate ladder. Some people use three or five four, or whatever. Apple has their recommendations for what a bitrate ladder should look like, and so a lot of people just use that, but it's really more general purpose than optimized for anything. The encoding letter that you provide to the end user, it's basically they don't necessarily just choose one-two. There's something called an ABR algorithm that's built-in as part of the player.

As your internet connection fluctuates, let's say you start off as a high connection, then it will look for the appropriate rendition in that bitrate ladder to pull and it will pull from the higher end, and if your connection drops, then it will say, "Okay, we can't enter a connection and can't handle this rendition anymore. Otherwise, we'll experience too much rebuffering. Let's go ahead and pull from the low end." That's a fairly simplified view of how it works in a sense that we send a manifest with different renditions, and on the client side there's an algorithm that chooses the best rendition given your internet connection speed.

[0:12:15.2] JM: When I was in school, I learned about how entropy affects file compression. That is to say that if you have two files that are the same size and you want to run a compression algorithm on each of those files, you won't necessarily get the same size of the compressed end result if there are different entropy rates in the bits within those files, because compression algorithms work based off of the patterns in the bits within a file. This is why you said that you don't necessarily have a straightforward trade-off between bitrate and file quality that's uniform, because it's going to actually depend your bitrate encoding ladder. The ideal bitrate encoding ladder is actually going to depend on the title of the movie, because different movies have different rates of entropy in the file itself.

If you have a movie that is just two people talking in chairs sitting across from each other, like if we did the movie version of this podcast, that's not a very high entropy title. You could probably compress a whole lot of that information into a smaller file that would still be relatively high bitrate. Whereas, if we took an action movie with all these cutscenes that's the same length as a movie of you and I talking to each other, that might be much higher entropy. Therefore, it's going to be more bandwidth intensive to encode it even to the same bitrate as that movie of me and you talking to each other. Is that an accurate description of the whole bitrate encoding ladder versus entropy thing?

[0:14:01.0] BD: That's definitely accurate. When you're talking about, right, like us just sitting here, let's say talking about a video of that, the things like the background for example wouldn't change, right? Then the background probably would be very simple; be white walls or something like that. The commercial algorithm is able essentially to look at that and say, "All right, these pixels they're frame from frame, they're staying the same essentially," and we don't really need to allocate a lot bits to figuring out to creating these frames. We can put more detail into the parts that do change and just the overall number of bits that are required to reconstruct the video file is fairly low, versus if there's a lot of motion, or if there's just a lot of changing detail, a lot of different scene cuts, then you can't really extrapolate a lot of information. You have to just use the raw bits to reconstruct those fine details.

If you look at, watch video that's streaming online, oftentimes it's a little hard to tell sometimes, but if you watch for it, you can really see the JPEG artifacts pop out in fast motion scenes that maybe this is an otherwise very high-quality video, because during those fast motion, since essentially it cannot cheat and extrapolate any information like it has been in still, or motion scenes, and that's when you get that those blocky frames.

A lot of people don't notice that, if they're not looking for it, especially if it's just a single maybe scene in a longer video, but it definitely becomes a lot more noticeable if you get more and more complex scenes, or more and more motion scenes, especially in things like sports videos, or I like to say action movies.

[SPONSOR MESSAGE]

[0:15:58.3] JM: Over the last two years, I spent much of my time building a video product. We had issues with latency, unreliable video playback, codecs. I was amazed that it was so difficult to work with videos. As it turns out, video is complex. Figuring out how to optimize the delivery of video is not easy, especially since there is both mobile and desktop and mobile users might not have as much bandwidth as desktop users. If you're an engineer working on a product that involves video, you just don't want to think about any of this. I can tell you that from firsthand experience, and that's why Mux exists.

Check out mux.com and find out how Mux makes it easy to upload and playback video. Mux makes video hosting and streaming simple. Today, you can get $50 in free credit by mentioning SE Daily in the sign-up process. Even if you aren't working on video right now, if you think you might work with video in the future, Mux is a really useful tool to be aware of. Check out mux.com. If you're an engineer who's looking for work, you can also apply for a job at mux.com.

On Software Engineering Daily, we've done two shows with Mux, and I know that Mux is solving some big difficult problems involving lots of data and large video files. To find out more, you can go to mux.com. You can get $50 in free credit by mentioning SE Daily, and you can apply for a job if you're interested in working on some of these large challenges. Thanks to Mux for being a sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

[0:17:53.0] JM: This gets us to this method called ‘per title encoding.’ Netflix developed this method for encoding for compressing their bitrate ladders, for defining their bitrate encoding ladders, Netflix developed this method called per title encoding. Explain what per title encoding is.

[0:18:13.0] BD: Yeah. per title encoding is basically people do per title encoding in different ways, so let me define the way that Netflix does it, because they were really the first ones to do it and to announce it to the world. The way that they did it was first, they defined their own type of quality metric for video. There's a bunch of different quality metrics for video in order to judge what's the amount of loss that occurs from the source video. That alone is a whole area of research that's fairly complex. The golden standard of figuring out what quality video is basically just asking someone, right?

Having a human eye judge the quality of video, because it's a fairly subjective thing. You can't just judge based off of the amount of blur, or the amount of artifacts, or things like that because for example with things like blur, sometimes the blur is supposed to be there, or sometimes with film grain, there's this noise that's actually part of the movie and you can't necessarily judge that as lower quality. We only really know that, because as human beings we can make that, we can discern that difference, but it's very hard for us to put that as far as in a quality metric.

Netflix created their own quality metric called VMAF. They basically used machine learning algorithm to approximate VMAF fairly closely to what's called a mean opinion score, which is the golden standard of just having lots of human beings look at video and rate it essentially. VMAF is considered now probably the best, or one of the best quality metrics when judging video. Then they took VMAF and they said, “Okay, we're going to take all these different types of video and we're going to do thousands of different encodes on it based off of the resolution and what the bitrate is.”

That’s the axes of a bitrate ladder is deciding one, what bitrates you want to put into the renditions, but also what resolution you want to encode at that particular bitrate. You can imagine if you're stepping up from 250 kilobytes up to 10 plus megabytes, or 20 megabytes if you're encoding at Ultra HD video and you're testing five to 10 different resolutions for each one of those, you start doing thousands of encodes for every single video, which Netflix is able to do, because they have the resource to do that, but that's very, very expensive, right?

For each of those encode, then they run VMAF on it, which then says, “Okay, at this particular bitrate, this particular resolution is the highest quality. That's the bitrate of a resolution pair that we're going to choose or this bitrate.” Then they move on and do that for every single bitrate. After they've done that, they know, okay each particular bitrates, these are the resolutions that we want, so that's the encoding ladder we're going to do for this video. Which means, at any particular bitrate, they know with confidence that they're giving the highest quality possible to the end user, subject to how accurate VMAF is, because sometimes people view VMAF as – because it’s very, very good, that's the end-all be-all quality metric, it's still an approximation of quality. It's very, very likely that they're giving the highest quality resolution, or bitrate to then a user, which means that they can do a lot of really interesting things.

Either they can give you better quality than anyone else is providing for a video, or they can say, “Okay, our quality is already pretty good, or quality really doesn't change much, but now we can also optimize bitrates.” Let’s say you get diminishing returns along the bitrate ladder as far as quality goes a certain point, so instead of encoding things at particularly high up on the bitrate ladder, we can get away with 99% of the quality with half of the bits, which means you need half the internet speed connection to watch Jessica Jones in HD, essentially. We still a lot of really interesting possibilities like that.

[0:22:30.4] JM: Does per title encoding ultimately boil down to how complex and how entropic the video file is, or are there different types of complexity that can exist in a video file?

[0:22:46.4] BD: Yeah, for sure. They started off with per title encoding. They've actually moved on to something else called per shot encoding recently. Essentially, they look at how – the complexity within a single video, right. It's just on average in a way and say, “Okay, how do we make sure that the resolution for this bitrate is optimized on average across the entire video?” Which means that if you have a video that switches pretty drastically between, let's say high entropy versus low entropy scenes, might not work as well, because then you're not optimizing for the individual scenes.

In order to optimize for individual scenes, imagine like for a single video, they were doing thousands of encodes. If you have a couple hundred scenes in a movie, right? It gets to order of magnitude that even that Netflix can't really scale that affordably, so they have to do some other things to make that work.

[0:23:48.5] JM: Okay. In order to get to the machine learning part of this conversation, I think we need to explore why Mux's product is different than Netflix. Why there's a different use case and the per title encoding, the slow per title encoding, which Netflix is per title encoding is slow, in order to run it for all their movies. That it's just a different use case than Mux. Can you explain why is it okay for Netflix this per title encoding to be slow to run and how does that differ from the context that Mux is doing encoding in?

[0:24:31.5] BD: Yeah. Doing that many encodes is very expensive to do. Because Netflix has a fairly like relatively small video library there and 5,000 EC2 instances that they can spin up for this, they're able to do that in a scalable way. For Mux, we expect to – because our video product, we hope one day it will scale to billions of videos. We can't do thousands of encodes for every single video that's uploaded. In order to do that, we have to approximate per title encoding through machine learning. That's the only way that it can really scale.

Another issue is that we have – we do what's called just-in-time encoding in which an encoding for a single video happens much, much faster than real-time, because we encode, or we cut up the video and do paralyzed encode across the video. When you upload a two-hour movie, let's say to Mux, it doesn't take a few seconds, maybe a minute to encode an entire movie. We don't want to add on unnecessarily long processing times to do part title encoding, because the as part per tile encoding is essentially two steps; you take – first, you do the encode and then you run VMAF on it.

If we were to do that, let's say for a two-hour movie with increased – it would take probably about 24 plus hours to do and even in a paralyzed environment. It takes a really, really long time to do per title encoding and the way that Netflix is doing it.

Instead, we decided to say, "Okay, what if we created a type of video classification algorithm instead to approximate the complexity of what's happening at a video, right?" Because in a lot of ways, if I was to tell you, "Okay, like we were doing it before, this video is an action movie with these type of scenes, versus this movie is a rom-com, or something like that." You don't really need to know, even to really see the scenes to know that okay, that action movies going to have higher complexity, right?

In order to then say, okay, we know that this is an action movie, we know that these are the type of scenes that are in it, can we then approximate a bitrate ladder from that instead? Which is a lot faster and more efficient to do for a video than to analyze the frames individually bit by bit, which is what we have had to do if we were to take the more brute force method of per title encoding that Netflix started.

[0:27:15.5] JM: Outline the plan to increase the speed of the encoding in a little bit more detail, and explain why deep learning was a potential solution to this speed-up.

[0:27:28.7] BD: Yeah. Essentially, the sets that we take for this is first, we take a video and we do look at the individual frames for it, but we do a sample of the frames, and we use what's called transfer learning in which we're taking embeddings, feature embeddings from the frames to get an approximation of what is the content of the video. Essentially what our model does is it's a content classifier that then embeds the content into a very large feature space that we then map to optimal bitrate encoding ladders.

Our bitrate encoding ladder is around 50. There's 50 runs in a ladder let's say, so going from 250 kilobyte, so up to 10 megabytes, right? You can think of the end output of that to be at each rung of the ladder with the optimum resolutions, right? That's what the model wants to predict at the end of the day, is trying to predict essentially a sequence of resolutions in 50 steps. That is a much faster process to take, at let's say frames of the image and move it through a first image classification model, and then into a video classification model, and then into a predicted 50 subsequence output, because training that we do for this model is done on actual per title encodes.

We've created this database of videos in which we have done these manual encodes for and in the same way that Netflix does, right? Let's say for the action, we've done lots of different action movies and we've done the Netflix style per title encoding, say like okay, these are the 50-step sequences that these different types of action movies look like. If we’ve done the optimal encoding for Terminator 1, right? Then someone uploads the Terminator 2, right? The encoding ladder is probably going to look pretty similar to it. We don't necessarily need to re-encode, do per title encoding again for Terminator 2. We can approximate that saying, using this video classification system and which works in seconds, rather than in hundreds of hours.

[0:29:57.2] JM: Part of this process is being able to have a classifier that can classify the videos as effectively being highly complex or low complexity, is that right? Is that what the classifier is doing?

[0:30:09.0] BD: Yes. It's not a direct measure of complexity. It's more of a measure of what type of video this is. That is often a proxy for complexity, especially since the way that when you do the per title encoding and becomes grouped in different types of encoding ladders and that is the state in which you're trying to predict, then the feature space in which we've embedded the videos, then they become grouped around tiers of complexity, which is what we end up seeing, but it's not what we are directly trying to classify the videos into.

[0:30:44.4] JM: The features that you're developing measurements for each video, what are some of these features? Can you put them into words, or are they in articulable?

[0:30:55.7] BD: It is a black box model in which we're not pre-defining the features, but rather we create a model active architecture and choose a layer within the architecture in which we think that it makes sense to pull out that feature. Some of the considerations that go into that is you don't necessarily want too big of a feature space, for example. If you have too many, too high of a dimension for your feature space, then nothing can coalesce around each other. You can't really get any sensible clusterings, or you need just enormous amount of data in order to cluster around a feature space of very, very high dimensions.

At the same time, if you want a feature space that's too small, or you get a feature space that's too small, then you might have too much overlap between different types of complexity between the videos.

[0:31:50.2] JM: There is a YouTube dataset that you used, called YouTube-8M. Describe why this dataset was useful and what it does and what you used it for.

[0:32:01.3] BD: The YouTube-8M dataset was really important for us to pre-train our model on. Basically, YouTube had already done a lot of video classification and open source it through the stay the same which different YouTube videos had been in some supervised way had been labeled into different categories. I think it's about 4,700 categories that the dataset that was released last year had. It's not all the labels were perfect, because they weren't hand labeled, but they were essentially graded against a hand labelled dataset to boost the accuracy.

They allowed us to basically have a model that was pre-trained on different types of video classifications that you see on YouTube, which then taking that pre-trained model, we are able to much faster train that in a way that instead of classifying just on the content classified on the bitrate encoding ladder. We use transfer learning essentially multiple times during the creation of our per title encoding model. That was one of the critical steps was for us to, instead of training our own video classification all from scratch which would require thousands of hours of machine time, we are able to get a really nice dataset from YouTube that allowed us a really good starting point onto then train on another our own dataset of encoding ladders.

[0:33:29.6] JM: Explain what transfer learning is.

[0:33:31.3] BD: Yeah. Basically, we have a machine learning model, a lot of the things that one machine learning model, or particularly deep learning is where transferring is used. When you have trained a very, very deep model let's say, the last few layers of that model are often relevant to let's say other models that might not have to sit and go, but might require the same type high-level classification.

For example, it's used a lot in image classification, right? Let’s say that a very popular image classification model is called inception. Inception, the way that deep learning model works is it takes low-level features and then creates high-level features as it moves up in the layers, until finally it gets to the classification layers in which it puts labels on things.

For example, when you give a image to inception, first thing it does and say, “All right, let's find where all the lines are and all the edges are. Okay, here are the edges. From that let's, figure out where the shapes are, right? Here's a circle, here's a square, here's a couple cylinders, whatever.” Then on the next layer up from that it might say, “Okay, let's combine these shapes, and so this circle and these cylinders, they go together." At the very, very last layers it says like, "Okay, when a circle and cylinder goes together like this, that's the table right?"

Maybe we want, instead of something that labels tables, we wanted it to label faces instead, right? Well, training inception is a very costly and time intensive task, and so in order to detect faces, we have to detect things like edges and shapes. We don't necessarily need to retrain all of inception, because it essentially is able to have that edge and shape detection already built-in. We just cut off the end of inception where it says, "Okay, let's label this circle and cylinder as a table." Instead, we say like, "Okay, let's add a few more layers and instead, show a bunch of faces instead, and it can combine these shapes and edges into a face detection model, and you can do that much, much faster."

That's essentially what we do with image classification using the YouTube model, or sorry, the YouTube-8M dataset and say, okay instead of labeling YouTube videos, you've seen tons of videos and or have been able to process all the low-low features of the video, now we just want to direct that into classifying bitrate encoding ladders, rather than high-level YouTube classification.

[SPONSOR MESSAGE]

[0:36:16.6] JM: In today's fast-paced world, you have to be able to build the skills that you need when you need them. With Pluralsight's learning platform, you can level up your skills in cutting-edge technology, like machine learning, cloud infrastructure, mobile development, DevOps and blockchain. Find out where your skills stand with Pluralsight IQ and then jump into expert-led courses organized into curated learning paths.

Pluralsight is a personalized learning experience that helps you keep pace. Get ahead by visiting pluralsight.com/sedaily for a free 10-day trial. If you're leading a team, discover how your organization can move faster with plans for enterprises. Pluralsight has helped thousands of organizations innovate, including Adobe, AT&T, VMware and Tableau.

Go to pluralsight.com/sedaily to get a free 10-day trial and dive into the platform. When you sign up, you also get 50% off of your first month. If you want to commit, you can get $50 off an annual subscription. Get access to all three; the 10-day free trial, 50% off your first month and $50 off a yearly subscription at pluralsight.com/sedaily.

Thank you to Pluralsight for being a new sponsor of Software Engineering Daily. To check it out while supporting Software Engineering Daily, go to pluralsight.com/sedaily.

[INTERVIEW CONTINUED]

[0:37:55.9] JM: Okay, let me see if I understand this right. You've got this YouTube dataset of 8 million videos and those videos are labeled with certain entities that are in the video, like cats and baseball bats and sunrise and a cup of coffee, all these different things. You can take those embedding, the labeled entities that are in videos and you can correlate them with ideal compression ladders, or bitrate encoding ladders for a given video. If a video has a cup of coffee and a cat and a baseball bat in it or something like that, maybe you can correlate a certain type of bitrate encoding ladder with that. Am I understanding it right? Is that what you're doing?

[0:38:44.6] BD: Yeah, absolutely. You see, it's not really measuring the complexity of the video directly. It's more saying, this is the type of content that the video has. Often, that's highly correlated with the complexity. Knowing that it's a video of two news anchors talking is another way of judging complexity in the video without actually measuring, directly measuring the bits, which is much more time intensive and difficult.

[0:39:14.5] JM: Two news anchors talking to each other, low complexity, the background is never changing. On the other hand, explosions and cars driving fast, that is going to be a high-complexity scene, and you're going to need a different bitrate ladder for the low-complexity scene, versus the high-complexity scene.

[0:39:33.5] BD: Right, exactly. I mean, imagine if I give you a bunch of videos to label the complexity on, essentially what you would be doing, you wouldn't be measuring change, like how many bits are in a frame and like how do they change and everything like that. You'd be saying things like, "Okay, here's a number of same changes and here's there's explosions happening and there's this person running, right?"

You'd be labeling the video in these high-dimensional features of the video, and so that's the direction that we wanted to do with our deep learning approach, rather than directly measuring, using VMAF and encodes.

[0:40:11.7] JM: If we have our video of you and I talking to each other in chairs just in a room, and that should probably be measured to have complexity that's similar to the two news anchors talking to one another, if the two news anchors was in the training dataset for the model. What would happen with our brand new video? How would that get ascribed with the correct complexity level and the correct bitrate encoding ladder?

[0:40:42.1] JM: Right. You imagine, essentially let's say it's seeing a video of two news anchors and then it sees the video sort of us talking, right? The visual features of that are fairly close. It's fairly related. It's a lot of those features then would be next to each other within the feature embedding space. Because of that, it would output a similar encoding ladder to the news anchor. This has a trade-off and that maybe, it's an approximation, right? That maybe, it might be a bit off. It might be slightly not optimized in certain points of the bitrate ladder, but it should still be fairly close.

Sometimes that approximation is really good when we see something, I type the video that we've never seen before, or something that we've seen similarities of between two different types of video. For example, let's say we have a – in our dataset, where we've done the actual encodes, we have a football game, an NFL game, and we also have video games as a category. Then someone uploads a Madden game, in which it has video game features, as well as actual football game features as part of that.

Well, the machine learning model should be able to say like, "All right, there's features that are related to this cluster and related to this cluster. Let's approximate and let's put it in the middle, right? It should an encoding ladder that is a blend of both those categories, which hopefully would be then pretty accurate in terms of being docked to an encoding ladder for a new category."

[0:42:22.1] JM: How did you measure the effectiveness of this deep learning solution, of this training, this machine learning training process for doing per title encoding?

[0:42:36.6] BD: We had to do a lot of actual brute force per title encodes, in which we knew what that optimal encoding ladder was. Therefore, we could measure it and say at these particular bitrates, these are the right resolutions. We do a fairly standard training test split in our dataset, in which we evaluate our deep learning approach on the types of, including ladder sets output and we measure against what the actual level is.

For our evaluation, we saw that along all 50 rungs in the ladder, it was getting between 70%, 80% of the rungs correctly, but that's actually because we don't choose all 50 rungs, sometimes that doesn't – that necessarily means, 70%, 80% as good. It really depends on how you choose the renditions from the bitrate ladder, as well as where it's off by. Because if it's off by a few bitrates at the high end, but we don't even look at the high end, let's say for a particular video, it might be a 100% accurate, or it might be 50% accurate if it's often a bad spot along the ladders. That's another issue that we have to optimize for and have to use additional machine learning techniques, in order to figure out what the best rate to choose that rendition is.

[0:43:58.1] JM: Is this running live today for you?

[0:44:01.3] BD: Yeah, it's in production for us. If you go to mux.com and you – and within our API, you set per title encoding equals, or to true in your API post, then it will run our per title encoding on your video and you can see the optimized encoding ladder as well that it uses, versus a standard set, which is we use, sort of Apple's recommended, pretty close to Apple's recommended ladder for the standard encode.

[0:44:33.3] JM: What was the length of time from ideation to deployment for the full soup-to-nuts, I guess architecture or whiteboarding scenario, to actually deploying this per title encoding algorithm?

[0:44:49.0] BD: It probably took about from ideation, I would say about six months before we got the first version out into production. The work on it definitely hasn't finished, because as we're still expanding at the dataset, expanding edge cases for the video, so there's still a lot of work that we want to do on it, but we're pretty happy in the state as currently in this initial version that we launched.

[0:45:17.0] JM: Mux is a fairly young company and a six-month project for improving the compression ratio for videos, or improving the compression ladder algorithm, I guess you would say. It's a large investment. Why was this project so worth working on? Why is it so important or so core to Mux that it was worth putting six months into devising this deep learning algorithm and doing all these engineering around just making a better encoding solution?

[0:45:48.7] BD: Well, we really view it as the first step for a lot of future features that we want to include to Mux. It’s definitely not a siloed project in which we implement per title encoding and that's it. It really leads to a lot of future projects. A big focus of where we want Mux video to be in the future is dead simple API for engineers to use, that doesn't just give you standard video, versus if you were to implement it yourself and put all the plumbing together or your video project. Really just a project in which you get video that would normally require a whole team of engineers from Netflix to implement. Instead, it takes you a few minutes to set up and you get the highest quality video possible, right? You get all these learning features that are built into it so that you know that the quality is optimized, you have different features that allow you to save space on the mount storage, have optimized delivery of your video.

There's a lot of future projects that we want to build on top of per title encoding. For example, per shot encoding is something that I mentioned earlier in which not only is the – you essentially take this model and instead of looking into the video from end to end, you look at you've run the optimization for every single scene. That really makes a huge difference, because you know that every single scene that you will get within video is the highest quality possible for what to your users’ internet connection will be. It's definitely not just a six-month project for us. It's an ongoing project, in which we want to make Mux video the best quality, the best video product available to engineers out there.

[0:47:37.2] JM: What I liked about studying this project was that it exemplifies a lot about where we are in software engineering today. First of all, you've got great machine learning frameworks that you can do all this in quite productively. I mean, six months it's a long time, but it's still much more rapid than you would have without a deep learning framework to be using. Not only that, you're leveraging Netflix's research in the per title encoding concept and their willingness to talk about that, and they have a blog post from 2015, or 2013, or something.

You also leveraged the YouTube dataset, the YouTube 8 million labeled dataset that YouTube was kind enough to put out there. You use transfer learning on top of that, and I like that it's just – this project is this example in time of all these different trends in these different companies being willing to put stuff out there that that other people can leverage. It's positive some ecosystem from a lot of different angles on this one.

I know we're near the end of our time, but did you have any broader insights about the state of software engineering, the state of machine learning when you were working on this project for the last six months?

[0:48:50.9] BD: Yeah. I mean, definitely implementing this project certainly felt standing on the shoulders of giants as far as how much we leverage from open-source projects. We've talked to Netflix, some of their engineers about how they use VMAF and how they do per title encodings, try to get more insight into our system. Everyone, the whole machine learning environment I think is it's great how much of a focus there are, there's around open-source projects.

I know that for me personally, there's a lot of different options I felt when we were evaluating how to do this. The amount of datasets out there, the amount of willingness between different companies to share the insights they've gotten around video classification, is just tremendously helpful. In fact, there's a new Kaggle competition that just opened on a few weeks ago in which YouTube had published an update to their dataset. The dataset now has better labels, it's a bit more accurate and on Kaggle itself, which is owned by Google. A lot of people share the different types of models that they're working on.

It's great to be able to go talk to the community about what's happening in video classification and to see companies like Google publishing these datasets for free, that would – if we were to do it ourselves, it would just be incredibly expensive, and I don't think it would really be possible in this state.

One thing that I did feel though in terms of looking at the recent advancement of different video classification and computer vision projects is that, there definitely seems to be more of a plateau as far as the actual deep learning techniques happening if you look a few years ago based off of different computer vision, deep learning techniques that as far as model architectures go, there was a lot of different shifts, a lot of new papers being published as far as what people were using and implementing. I feel that definitely has coalesced a bit around certain machine learning techniques as being the main ones that people go to and really just smaller iterations, improvements on those techniques now, rather than big leaps forward, like we were seeing two, three years ago. That sort of been an interesting development I think to see over the past year.

[0:51:19.6] JM: Ben Dodson, thank you for coming on Software Engineering Daily. It's been great talking to you.

[0:51:22.3] BD: Thank you. Thank you for having me. It’s been fun.

[END OF INTERVIEW]

[0:51:28.1] JM: GoCD is a continuous delivery tool created by ThoughtWorks. It’s open source and free to use and GoCD has all the features you need for continuous delivery. Model your deployment pipelines without installing any plugins. Use the value stream map to visualize your end-to-end workflow. If you use Kubernetes, GoCD is a natural fit to add continuous delivery to your project.

With GoCD running on Kubernetes, you define your build workflow and let GoCD provision and scale your infrastructure on the fly. GoCD agents use Kubernetes to scale as needed. Check out gocd.org/sedaily and learn about how you can get started. GoCD was built with the learnings of the ThoughtWorks engineering team, who have talked about building the product in previous episodes of Software Engineering Daily, and it’s great to see the continued progress on GoCD with the new Kubernetes integrations.

You can check it out for yourself at gocd.org/sedaily. Thank you so much to ThoughtWorks for being a long-time sponsor of Software Engineering Daily. We’re proud to have ThoughtWorks and GoCD as sponsors of the show.

[END]