EPISODE 338

[INTRODUCTION]

[0:00:00.5] JM: Convolutional neural networks are in machine learning tool that uses layers of convolution and pooling the process and classify inputs. CNNs are useful for identifying objects in images and video. In this episode we discuss the application of convolutional neural networks to image and video recognition and classification.

Matt Zeiler is the CEO of Clarifai, an API for image and video recognition. Matt takes us through the basics of a convolutional neural network. You don't need any background in machine learning to understand the content of this episode. Matt also discusses the subjective aspects of the image and video recognition and some of the tactics that Clarifai has explored. This is far from a solved problem. There is lots of exploration and experimentation to be done.

Matt also discusses the infrastructure of Clarifai; how they use Kubernetes, how models are deployed, and how models are updated.

[SPONSOR MESSAGE]

[0:01:06.8] JM: Artificial intelligence is dramatically evolving the way that our world works, and to make AI easier and faster, we need new kinds of hardware and software, which is why Intel acquired Nervana Systems and its platform for deep learning.

Intel Nervana is hiring engineers to help develop a full stack for AI from chip design to software frameworks. Go to softwareengineeringdaily.com/intel to apply for an opening on the team. To learn more about the company, check out the interviews that I've conducted with its engineers. Those are also available at softwareengineeringdaily.com/intel. Come build the future with Intel Nervana. Go to softwareengineeringdaily.com/intel to apply now.

[INTERVIEW]

[0:02:00.0] JM: Matt Zeiler is the CEO of Clarifai. Matt, welcome to Software Engineering Daily.

[0:02:04.1] MZ: Thanks for having me.

[0:02:05.3] JM: Your company Clarifai specializes in computer vision and I want to work our way towards some of the modern techniques and the modern applications, but let's start with some older ideas within computer vision. Before all of the excitement around neural nets and deep learning, what were people doing for a computer vision?

[0:02:27.8] MZ: Way back it was a lot of hand-engineered algorithms, and when I say way back it was really up until the last four years when people were using hand crafted algorithms for computer vision, and that mean things like edge detection, color histogram, the formable part models where they have some kind of springs in them to line different parts of objects in order to recognize them. Those types of things where you kind of have a preconceived notion of what would be important for recognizing things within images, and so you build the engineer algorithm for that. That's much different than neural nets, was simply learned from the data.

[0:03:10.2] JM: The older computer vision applications were not something that you would call machine learning?

[0:03:16.3] MZ: Yeah, that's correct. For the most part it was more hand-engineering stuff. That's not to say machine learning was just invented in the last four years. The algorithms for a machine learning are also many decades old but they weren't the preferred algorithm for computer vision up until recently.

[0:03:34.6] JM: What does that term machine learning mean to you?

[0:03:37.3] MZ: I think of it as being a synonym to just learning from data, and that's why it's machines learning from the data. Its pattern recognition that the algorithms instead of hard coding examples, they can actually doing these patterns directly from looking at many training examples set up and labeled by humans and they can extract those common patterns and learn a representation for those common patterns within the model. There are lots of types of different machine learning algorithms and neural nets that are thrown out more recently as the go-to-algorithm are just one algorithm within machine learning.

[0:04:13.9] JM: Linear regression is core to help machine learning works. Explain what linear regression is.

[0:04:20.2] MZ: Yeah, you can think of it as actually one layer neural network. It's just fitting a single weight to your input. Your input would be a vector of numbers, let's say, and those numbers might be the pixels of an image or they might be a collection of stock prices, or something, some kind of signal. You want to optimize weight vector that when you dot product with your input vector, it gives you a high score when it's a positive example and a low score when it's a negative example. Linear regression helps solve that problem.

[0:04:53.5] JM: Explain how linear regression fits into machine learning? How is it applied in machine learning?

[0:04:59.4] MZ: Yeah, it's one type of algorithm within machine learning and typically it's used for simpler problems where you don't need lots of layers of understanding and representation in order to extract a more complex signal. One of the other benefits is it can be a very large parameter space. You can have lots of different inputs and a scale as well and there're lots of both engineering toolkits and algorithms to support really high dimensional linear regression problems.

[0:05:31.9] JM: But it sounds like linear regression is maybe not as applicable or not as effective for image recognition. Is that accurate — Or an image processing?

[0:05:31.9] MZ: Yeah. If you look at, for example, a large picture and there is a dog and a tree and a mountain or something like that, there's a huge level of abstraction that the mind does when it sees those pixels and says, "That's a dog, and that's a tree, and that's the sun, or the mountains." It's going through lots of processing to do that. When you think of linear regression, it's really only going through one layer of processing to do what it does. When you think of a neural net, it goes through many many layers and that's what makes neural nets more powerful for image recognition versus just linear regression.

When you look at what a neural net learns, it actually learns things that are pretty logical as it builds up these higher-level representations for understanding images, things like edges and colors are learned in the very first player and that's something you probably could learn with some linear model where it's just applied to patches of images, but these neural networks apply it to images, learn this in the first layer without even having some constraint on the fact that it should learn edges or it should learn colors. They just naturally do that.

By the second layer, starts building edges and colors into things like circles and corners and parallel lines. By the third layer it will start building things like eyeballs and fingers and hands and stuff like that and then higher layers might build a face or an arm and the top layer might output person is likely to be in this picture. That's the power of having multiple layers versus linear regression which can be considered as one layer of a neural net.

[0:07:20.8] JM: In machine learning we are interested in building models. For Clarifai, these models are trained by processing images one by one. How does the process of getting a single image integrated with the model? Give me an overview of that process and how it relates to the construction of the model that you're actually developing over time with multiple images.

[0:07:20.8] MZ: Yeah. The way these algorithms are trained is, for example, our general model has seen over 10 million images that are labeled by humans, and in the training process it goes through those millions of images in small batches. It's not one at a time but more like 128 at a time, for example. It's pretty much as much as you can fit on one graphics card which we use for training, because that's very parallelizable, very efficient. The more you can throw at the GPU, the better.

As it gets these 128 images, it runs them through the current state of the model and it outputs its best guess at that time, and of the start of training, all the models parameters are just random. Instead of having nice edges and colors in the first layer it just has random initialized parameters that don't look like anything in particular.

Its best guest is basically random probability for each of the output categories that you care about recognizing. It compares those guesses with the ground truth labels that a human has provided and that gives it its error, that's the mistakes it's currently making, and then there's a bunch of basically calculus that you learn and underground in university, that same calculus defines how you take that error signal and update all of those parameters throughout your model. You make a small adjustment to each of the parameters, not the huge one because you've only seen one or 128 images in this small batch. You don't want to just memorize those images and only optimize from them. You make a small update to your parameters and then you move onto the next batch, and using the new parameters you go through and make a guess, correct the errors and make a small adjustment to your parameters and you loop through your whole dataset and millions of images in that way.

Over time starts learning common patterns in things like this is a dog, and it's a dog because it recognizes this edges in the first layer, these corners in the second layer and so forth, and higher layers. That's kind of the training algorithm that builds up from these incremental improvements on small sets of images to learning from the huge collection of images.

[0:09:59.6] JM: You have touched on elements of a neural network, and the last several years have seen growth in the adoption of neural networks. Can you give a formal explanation for what the neural network is?

[0:10:13.0] MZ: Well, it's an algorithm. It's just a computer algorithm and it's meant to simulate how the brain works. That's where it gets its name; neural, but nobody really knows how the brain works so it's a best guess, best approximation. It's composed just like the brain is in multiple layers of processing. When you see an image come through your eyes, it actually has very similar layers that recognize edges and colors, then they get composed into mid-level primitives like corners and circles and stuff like that. Then much deeper into the brain after multiple layers, then it starts recognizing high-level things, like dog and tree.

These neural network, as I explained, in a very similar way, but it's, again, is an algorithm that has a very simple mathematical model. These update rules are called gradients because there are just calculus based on how you want to change your parameters in order to optimize your accuracy. It works out to be pretty simple calculus from school.

[0:11:14.7] JM: The adjective that often gets pre-pended to neural network is convolutional neural network. Explain what convolutional neural networks are.

[0:11:23.4] MZ: Yeah, that's right. For images and video and other types of data that has kind of local patterns like edges and colors and neighboring patterns, convolutional neural networks work very well and that's because convolutional networks have what are called filters that are small regions of weights. Those are applied over every location in the image just like template matching. Think of them as templates that you're just going to slide over every possible location of pixels in the image and when they overlap very nicely with the underlying pixels that gives you a strong response that gets passed to the next layer.

You can think of convolutions of as simply template matching, and when picture that, these templates sliding over images, it makes a lot of sense why they work well for images because they become invariant to things like the dog is in the left side of the picture or the dog is on the right side of the picture. If you slide the same template over all parts of the picture, you can eventually line up with a dog really well. It's going to find that dog regardless of where it is in the picture.

That's why they work much better than just supplying the traditional type of neural network which is often called a fully connected neural network because it's fully connected to all of the pixels. That doesn't have the same property where when you move an object within the image it can no longer use the same weights. It can't share those same template weights. It makes it much more difficult for that type of neural network to learn images and video which are inherently kind of translation invariant as I described.

[SPONSOR MESSAGE]

[0:13:13.8] JM: For more than 30 years, DNS has been one of the fundamental protocols of the internet. Yet, despite its accepted importance, it has never quite gotten the due that it deserves. Today's dynamic applications, hybrid clouds and volatile internet, demand that you rethink the strategic value and importance of your DNS choices.

Oracle Dyn provides DNS that is as dynamic and intelligent as your applications. Dyn DNS gets your users to the right cloud service, the right CDN, or the right datacenter using intelligent response to steer traffic based on business policies as well as real time internet conditions, like the security and the performance of the network path.

Dyn maps all internet pathways every 24 seconds via more than 500 million traceroutes. This is the equivalent of seven light years of distance, or 1.7 billion times around the circumference of the earth. With over 10 years of experience supporting the likes of Netflix, Twitter, Zappos, Etsy, and Salesforce, Dyn can scale to meet the demand of the largest web applications.

Get started with a free 30-day trial for your application by going to dyn.com/sedaily. After the free trial, Dyn's developer plans start at just $7 a month for world-class DNS. Rethink DNS, go to dyn.com/sedaily to learn more and get your free trial of Dyn DNS.

[INTERVIEW CONTINUED]

[0:15:12.2] JM: My understanding is you take an image, you turned that the image into its mathematical representation so it looks like a bunch of numbers and then you have these different matrices that you perform some sort of calculation between the smaller matrices across these different image. These smaller matrices serve as "edge filters", and these edge filters help you outline different regions in the picture and then you can build up a higher level understanding that it's not just the pixel-wise understanding of that image in subsequent layers of the network.

[0:15:53.7] MZ: Yeah, that's exactly right. To give you some rough numbers if the image is like 256 x 256 pixels, these small template filters, they might be 5 x 5 or 7 x 7 or even 3 x 3. Just small portions of the image. That's because, again, there is multiple layers in these neural networks and this can be filters just like those ones that the plight of the pixels, the applying higher layers to the responses from the lower layers and that's how it builds up these higher level representations for dogs and trees and so forth.

[0:16:29.7] JM: The term convolution is derived from the word convolve. Explain what we are convolving and what that actually means in this context.

[0:16:39.2] MZ: Yeah, it's our mathematical term that is used in signal processing for sliding over a window over top of another signal. In this case, the window is a filter and they are being slid on top of the overall image. When it matches, is doing a dot product essentially in order to match, and that's just what applying the weights with the underlying values of the pixels. When they line up very well you get the strong responsibility value.

That's all convolution is. There is a related term correlation, they just basically flip the filters with each other, but it's just too weak correlation. I think that's a more common term used in English than convolution, but they are very closely related and they're all just mathematical signal processing terms.

[0:17:25.2] JM: We take an image like a picture of a dog and we take all these edge filters and we convolve these edge filters over the picture of a dog and it gives us a set of edges. Let's say we convolve a set of vertical edge filters and then a set of horizontal edge filters and a set of diagonal edge filters. We started to build up these different understandings of that image in a pixel-wise fashion. We get horizontal edges. We get diagonal edges. What do we do with these collection of edges?

[0:18:06.2] MZ: To make a bit of a distinction, you mentioned horizontal, vertical and diagonal, and there is going to be many more than that in practice. As you apply each one of these filters to the image it creates what we call a feature a map which is all of the responses of sliding that filter over the image at each location. That feature map, there is one feature map for every type of filter you have; the horizontal will produce one feature map, the vertical will produce another and so forth, and they are roughly the same size as the original image. They are much bigger than the filters themselves. That's a convolution player and that's where the convolution neural net gets its name.

There are other types of players that become important as you think of building up this whole neural network. Another one that we haven't discussed is what we call a pooling layer. It's pulling because it groups together small regions of this activations that were produced from the previous layer and the something like either average them or take the maximum element, and max pooling typically works better.

You look at all the values and say a 3 x 3 region of these feature maps and you simply take the strongest one. The only get one value for every nine values. What that does is shrink the size of a feature map down by a factor of three and sometimes these regions overlap so it might be less than three. Then that's the type of layer that helps with this in variance to where exactly in the image the object appears.

If you think of it visually you could have the dog on the far left of the pooling region or the dog on the far right as long as that activation is the strongest activation. It's still going to be relayed up to the next layer in the stack of neural net layers. Then that just gives you smaller size feature maps the same number as was in the previous output from the convolution layer and then you typically repeat this process. Those other types of layers that are less important like just keeping all the positive elements is actually an important one but it's very simple, and you just throw away negatives. Those are the kind of the building blocks is you stack multiple of these layers up.

You might have another convolution layer and a pooling layer and other positive layer, maybe multiple convolution layers back to back, and that's where kind of the expertise, like the people at Clarifai here have come to know overtime, how to actually build and architect these neural networks in the right way with the right number and proportion of layers, the right size of these layers. How many of these filters you actually want in each layer and so forth. It's not just convolutions, there are multiple types of players and many layers to get to the final prediction at the end.

[0:20:53.3] JM: .When you're talking about — Let's just take the first — If we are talking about the first convolution where you — Let's say the feature maps that we're getting are the vertical edges and the horizontal edges and the diagonal edges and perhaps some other future maps that are developing. As you've said, perhaps after that, were doing a pooling which will condense a feature map like a vertical edge feature map.

At the pooling stage are we doing any sort of interaction between the different feature maps that we have developed in the first stage or are we just retaining these different feature maps through the different layers of the network?

[0:21:30.0] MZ: That's a great question. Typically what people do is they don't pool across different feature maps. We actually tried that. Rob Fergus, who is my Ph.D. adviser and myself at NYU tried this during my Ph.D. and we found mixed results. The benefit of pooling across or pooling together multiple features is that you can speed up processing because now you have a less information to pass to the next layer but it just didn't learn as good of representation. It wasn't as accurate at the end of the day. It's possible but it doesn't seem to work quite as well.

[0:22:07.4] JM: Do you throw out any of these feature maps throughout this — You're alternating between this convolution and pooling and convolution and pooling, or do you have the same set of features throughout each different phase of —

[0:22:22.2] MZ: No. Yeah, that's a key point as well. If you think of the first layer, the image itself, you have red green and blue. You have three color channels. Essentially, if you think of the pixels themselves as feature maps, you have three in the first — The input to the first layer is three, and then you get to decide how many outputs of that first convolution layer you want. Typically it might be 64, 96, 128, something like that. Then you do pooling that keeps that same number. You do rectified which is just keeping the positives and that will keep the same number of feature maps.

Then the next convolution layer, you get to choose again how many outputs of that player. It might take N64 one whatever the output of the first layer is, but then you get to choose. Maybe you want it to be 128 or 256. That's another dimension, another hyper parameter that we call it that goes into tuning the capacity of these architectures. You are free to choose that in that controls kind of the amount of the information that flows up to the network.

[0:23:27.1] JM: As you go through these stages of convolution and pooling and convolution and pooling, eventually you get to a higher level understanding of the contents of that image. Help me understand how you get to that.

[0:23:40.0] MZ: Yeah. The final few layers of the neural network typically aren't doing convolution anymore. You can still think of it as convolution, but at that point because of these pooling layers that keep producing the size of the feature map the size often gets all the way down to one by one. There is no more spatial extended. It's basically one pixel. It's the only information flowing through.

That other dimension that you can control, the number of feature maps and the layer, typically it's really big at that size. You squish together all the number of rows, the number of columns, but you explode the number of features. That lets it learn lots of different categories for example and the differences between fine grain categories.

That final layer is the number of features corresponds exactly to the number of categories that you want to recognize. Finally, the outputs are normalized in some way typically so that instead of just giving you kind of the floating point range that you would get from the mathematical operation, you want to normalize it to be probabilities, and you can do that in multiple different ways. You make it sum to one so that out of all the different possible concepts in your vocabulary you want them to add together to have unit probability.

Another possible way, and that's what we do at Clarifai, is normalize each one of them to be independent so that way you can have — in the single image they can have things like a dog and a tree and a mountain all appear in the same image. Whereas if you normalize the sum to one, they kind of conflict with each other.

[0:25:19.5] JM: I want to just try to get a better understanding of this because we've been talking about the idea of building up and understanding of where the edges are in the picture and now were talking about identifying whether a mountain or a tree exists in this picture. How the bridge the gap between those two different things? Because those seem like very different problems and perhaps it involves getting multiple images going through this model and having some sort of phase where we are identifying what a collection of edges aggregates to. Can you just shed more light on that?

[0:25:58.7] MZ: Sure. This is the beauty of neural nets, is we don't have actually phrase it in the algorithm or define how that comes together. We simply feed it. After we reconfigure the number of layers, how wide each layer, the different types of players. That's kind of the experimental process, but then you just feed it pixels and categories and it learns the rest.

That's a really important part because it is very complex and this is where the hand-engineered algorithms fell apart. You could reason about different ways of grouping together edges and colors and simple stuff like that, but at some point it gets too complex so you don't understand how to actually build up a high-level representation that is flexible enough to recognize a dog and the dog could be running or sitting by slipping, it could be in different colors, different types of fur, different breeds of dogs. There are lots of different things that come in together that a human would classify a dog being present.

That's the type of stuff that's really hard to program in that's where neural network shine and that's why they are such a powerful tool for a lot of different problems because if you have the right data you can apply it and it will learn what's important from the data. It can make that leap from the well understood staff lower down the network all the way up to predicting the categories that you care about.

[0:27:22.4] JM: Right. One key part of this is that the input to the neural network is typically labeled images. You have a large set of labeled images and some human has said that, "Oh, this image contains dog. This image contains tree," and the job of the neural network is to find sets of edges or sets of features that correlate with those labels. It can find a set of edges or a set of features and it will say, "Oh! Okay. This image has these set of features," and it would look at the labels and it will say, "Okay. Dog." The set of features correlates with dog.

[0:28:09.7] MZ: Yeah. Exactly. That's why it's really important to have a really high quality dataset in order to get a really high-quality model that its trained from it. That's something we did very early on at Clarifai is start writing web crawlers, collecting lots of data, making sure its high-quality labels on the data. That something that's really important to understand is not necessarily the quantity of data. It's not like you can just crawl the web and do a very broad crawl and get high quality data. You really need good sources of data that go into producing a good-quality model, because ultimately that's what it's learning from, is taking those pixels, taking those hand-labeled categories and trying to learn the rest for you.

This feature of neural nets, this learning capacity is really exciting because it opens up lots of possibilities. For example, we have customers in the medical domain and we don't employ any medical doctors within Clarifai, and we don't need to because our customers can be the experts in their field. They can label data however they categorize it and they could be experts of going to school for 12 years or whatever it takes them to become a doctor these days. Then the model is learned from that data and it can learn to predict these are very complicated things that many are to become experts at very quickly and very easily for the model because it's just a pattern recognition machine.

That's what makes it really exciting because it would be very hard for us to define the differences between different diseases if we are to sit down and think about it and program at, whereas the neural net can just learn from experience.

[0:29:48.0] JM: If I got a bunch of optometrists or ophthalmologists, I guess, in a room and I give them a stack of photos of eyes and I told them to identify whether this eye has diabetic retinopathy or not and I get some consensus around that, or even I could just take photos where they are confirmed cases of diabetic retinopathy, I could feed it into a convolutional neural network and the network would learn what are the trends that are — What are the visual trends that tend to, in terms of the picture, that tend to correlate with diabetic retinopathy. How do you validate that a model has developed an accurate enough understanding of what diabetic retinopathy for example is?

[0:30:37.4] MZ: Great question. Typically what we do is split all of the data we have available into two different sets. I mention the training set already. That's the stuff we have labeled really well and that's typically high volume. More data you have that's really were labeled the better the model in general that you'll get.

Then we hold out a set of data that's typically called a validation set or at test set that the model doesn't get to see when it's training and so you can evaluate the performance of the model on that held out data to see how well it's learned. That is really important because what you can do if you have a really big model when lots of parameters that can learn and a very small dataset, you can literally memorize the whole dataset. The models have that much capacity that it will learn very subtle things that probably a human would key on in order to memorize it, but the model has enough capacity to do that.

If it just memorizes the training set, it’s is going to do very poorly on the new data in the test set that it hasn’t seen before. That’s why it’s important to hold it out the training process and evaluate your accuracy on that held out data.

[0:31:51.5] JM: We’ve described the general technology that underpins image recognition at this point and I would love to get the finer understanding of some of the subjective decisions that you’ve made in building models at Clarifai, because obviously there is a lot more nuanced than we have covered in our survey of this technology in this conversation so far.

Let’s talk about how this translates to some of the applications that you’ve built at Clarifai. For example, you have a model that can detect different pieces of apparel, so I can send it in image and it will say, “There is a probability of .8 that this image contains a turtleneck.” Explain how you trained that as an example.

[0:32:35.0] MZ: Yeah. It all starts with identifying a need in the world and then getting data to build a model to solve that. We talk a lot of customers. We are fortunate to have a lot of inbounds, which is great. We do a lot of outbound sales team to reach out to different customers in different industries. Fashion and apparel is one of our customer bases and they help educate us on the different types of categories that they care about recognizing, and that could be their product catalogue under actual retail page or just the way they organize their warehouse. There’re a lot of different applications and the catalogue might have different categorization depending on that.

Typically for some of our big enterprise customers, they have the data already available that they have manually labeled according to these different categories. If they don’t have data available we can also collect data on their behalf that’s really valuable for them, saves them a lot of time, saves them a lot of money in getting AI into the platform.

Then once the data is collected, we train the model and then it usually — The final stage before getting put into our platform is going through quality control checks. We look at things like the accuracy on held out data. We look just qualitatively and this is where we use the demo you see on our site internally to check the quality of our models before rerelease into the world. Do they handle the common use cases? What about some edge cases? Does the vocabulary make sense for the application? And so forth. There is a lot of kind of steps to the process and then the final step is documenting it and getting it into our demos. People can see and start integrating it into their products.

[SPONSOR MESSAGE]

[0:34:25.8] JM: Don't let your database be a black box. Drill down into the metrics of your database with one second granularity. VividCortex provides database monitoring for MySQL, Postgres, Redis, MongoDB, and Amazon Aurora. Database uptime, efficiency, and performance can all be measured using VividCortex.

VividCortex uses patented algorithms to analyze and surface relevant insights so users can be proactive and fix performance problems before customers are impacted. If you have a database that you would like to monitor more closely, check out vividcortex.com/sedaily. GitHub, DigitalOcean, and Yelp all use VividCortex to understand database performance. Learn more at vividcortex.com/sedaily and request a demo.

Thank you to VividCortex for being a repeat sponsor of Software Engineering Daily. It means so much to us.

[INTERVIEW CONTINUED]

[0:35:43.8] JM: What if I wanted to train something that was more niche? If I wanted to train a model to recognize specific beanie babies? How self-serve can you make that? Obviously the dream is to build the generalizable platform where — People are getting more of an understanding for how these models work in training and whatnot. Pretty long way from where you can build something that's self-serve because talk about nuance, you have to — With a lot of nuance, explained to the customers, "Hey, here's how you can build your own thing." It sounds like we're still sort of at the era where you don't want to make it completely self-serve because you need to do a lot of stuff for the customer. If a customer wanted to build a specific beanie baby identifier, to what degree can you make that a self-serve thing?

[0:36:36.6] MZ: Yeah, we actually launched a product in the fall that does exactly this, and we'd like to call it custom training. We don't actually have a demo of it unfortunately on our site. It's something we are working on currently. It lets our customers customize a platform so they can recognize — They can teach it to recognize whatever they care about. We phrase it in the same whereas people upload images, they label them with different concepts and then they train a model based on those images and concepts that are labeled in order to recognize the concepts that they care about.

Then they can apply that trained model on any new images that they feed into the APIs. Beyond just the APIs, all that, everything I just said is available as APIs, but we also made its user interface so it's even simpler to use. That lets you just create a Clarifai account, you can sign up for free and you don't even need a credit card and you'll see a link to what we call our preview UI. It lets you view all of your data. It lets the event drag-and-drop data off your desktop. Then it starts — indexes it first search so you can organize it and then it lets you train whatever you want to.

Some of the objectives of launching the product were to remove the roadblocks that we see in training artificial intelligence today and we view it as five different bigg roadblocks that are really hindering people training AI and reaching that point that you just mentioned.

One of them is you need lots of code if you want to build something yourself. Typically, you download some open source toolkit and you customize it and that's a lot of configuration. You really need to have a team of data scientists that know what they are doing and can actually set up the data pipeline the right way and configure these neural networks. That's the second thing; a team of data scientists.

The third thing is a special hardware. We use graphic cards to train a neural network and they need optimized code. They need different types of monitoring and maintenance, and so we take care of that in our API. Typically to train a new category of a thing, you want a thousand examples per concept that you want to recognize, and we want to get rid of that because it hinders you from doing things like teaching it the name of your dog, because you probably don't have thousands of pictures that are readily available.

Then the final roadblock is it takes weeks of time to either get all of these different factors together or to train every iteration of the model. We wanted to make that significantly faster. Now it's down to literally seconds to teach the platform and just a few images and you don't need special hardware, you don't need any knob tuning or data balancing or anything like that. We take care of it for you.

In our APIs, it's literally only six lines of code. It's very simple to use, and then with the user interface you don't even have to write a single line of code. We factor all these different roadblocks to AI and made it very simple for you to customize it and get exactly what you want.

[0:39:36.8] JM: We've been talking about image recognition and image analysis. You also have APIs for video; you have done a lot of work in the regional analysis and prediction space. Since we've already explored the convolutional neural network approached for image the construction, can you explain how well that relates to video analysis on what you have to do differently when it comes to video?

[0:40:08.8] MZ: Sure. Convolutional neural nets also work very well for a video and you can think of it as just another dimension you have the number of rows and number of columns in your images and that's each frame of video is essentially an image, but then there's the time dimension. It's a third dimension. Some people do three-dimensional convolution. Instead of just a template over the rows and columns, there is a template dimension over time as well.

That sometimes works good, sometimes it's just a very inefficient because it's very expensive computationally. Other people do things like run on every frame of video, run a regular image convolution neural network and then high up in the network start connecting overtime. That's something we do at Clarifai. That makes it kind of a blend between efficient and accurate.

[0:41:02.7] JM: I would love to talk more about the data engineering side of things because you mentioned all of the complexity involved in that. You kind of glossed over it. Training conditional neural networks is computationally expensive. How do you keep the costs down? What is your infrastructure look like?

[0:41:22.3] MZ: Yeah. We actually went through a big revamp of our infrastructure over the last year. We have a mix of two different clouds; one is AWS, where we run all of our production traffic, api.clarifai.com goes there. Then for a training these neural networks we actually host our own machines over in New Jersey in the co-lo facility.

That gives us the best of both worlds. For our customers, it's highly reliable and maintain and monitored, thanks to AWS. For the training side, we can have the latest graphic cards installed as they come out, and so we just did that last week it turns out. Drove over the New Jersey and upgraded the co-lo, graded the network so that we can train across multiple machines, that kind of stuff, to give us the latest training infrastructure for our research team to push the limits of these neural networks.

Once we have a model that's trained on our co-lo, we can push it into production and it's deployed over there in AWS. Our stack is composed of a lot of pipeline code for the neural network side of things. Our APIs are written in Go. We just started doing that over the last year which is exciting new language and it's paying dividends thanks to its way, it handles lots of request in parallel and we deploy all of our code into Kubernetes. That makes managing lots of different services and doing deployments very easy and low maintenance. We've got all of that in place and that has made our continuous deployments much faster.

[0:43:02.4] JM: I hear around the Kubernetes conversation. How often are you retraining these models, because if you get the bunch of new data you could retain a model?

[0:43:13.7] MZ: Yeah. It goes through different phases when we get it into the model gallery. Initially, it's launched into kind of this preview stage where people could start playing with it. As it flows to beta and general availability there is sometimes tweaks between the model. That either is just we are finding the vocabulary of the concepts that we recognize or collecting more data. That's usually a tighter iteration.

Then after we launch it in GA, we continuously collect more and more data. It's one of our KPIs as a company that we track our key performance indicators and so it's a big initiative for us to keep improving these models. Then the other KPI related is model accuracy. We actually drive that overtime, and so it depends a lot on the model and how quickly we get more data. It could be anywhere from weeks of iteration or months of iteration between model versions.

[0:44:09.5] JM: I suppose if you are building a model for diabetic retinopathy are cancer, maybe it's more important to update the model more frequently than if you're building at tree detection model, so you can imagine different schemes for updating different models, different frequencies.

You mentioned the Kubernetes stuff. I still don't have a good picture of how in this kind of company — What's the mapping between the model and the computer nodes that it's deployed against it. Is it one container per model or like, I guess, maybe a — What is it? A replication group per model or something like that, or I'm a envisioning this wrong?

[0:44:53.5] MZ: No. That's exactly what we have actually current. We are actually revisiting that in the future but currently we have one deployment per model and it has multiple replicas. It's nice because with Kubernetes you can talk between different services just by their DNS name. We have a way we specify our models internally. The Go APIs that serve our customers' requests translate that request to the underlying model identified and then it just fires off that to that DNS name and it just finds where the neural network is deployed and then we get a response back and then send it to the user. It makes kind of that service management and deployment much much simpler than what we had previously.

[0:45:40.4] JM: What's the workflow between training a model and deploying it? I guess you just training it completely off-line and then deploying the model once it is fully trained or are you forking? You could also fork every submission that's going to prod. You could fork that submission to training a model that's being trained to update and be deployed. Yeah, I don't know. What does that look like?

[0:46:07.3] MZ: Yeah. The way we do it is more on the off-line facet. We train on the co-lo which is a separate deployment or when it's ready, when its high-quality high accuracy, we can then take the trained model which is just the parameters. It's not the data. Then it just needs to be checked in to our code repo so that the new APIs know about it. They know the unique model identifier and all that kind of information that they need to know in order to just serve requests for that model.

Overtime as we introduce new types of models, there might be code changes as well. For example, we launched celebrity recognition recently, which not only recognizes the names of celebrities, but it actually finds where in the picture the face is of that celebrity. It's a different format in the API, different type of model different request in the backend, all that kind of stuff. It has correlated code change, and so they have to be deployed together.

As things scale up, all the parameters of the model are available to the different pods in Kubernetes. If we have more traffic than one type of model we can scale up more pods to service the traffic.

[0:47:22.0] JM: It seems to me that the bread and butter of a company like Clarifai — Obviously you have the typical challenges like hiring an infrastructure and stuff, but what we went over earlier with the convolutional neural network is just like deciding how many layers and what you're tuning. That's really the special sauce. It's not like you need to build your own machine learning framework from scratch, you could take some machine learning framework or deep learning framework off-the-shelf. Maybe you can tell me what you use, maybe TensorFlor or something. That's less important than the fact that your company has built up a domain expertise in how to construct these models. Is that accurate?

[0:48:11.1] MZ: Yeah, that's right. That's a big differentiator between trying to build this type of thing yourself and how it's like if you're a developer at the company, it's much better to use a service. You use this all the time. If you think of like sending a text message to do verification of somebody's phone number or an email, you use Twilio. If you think of doing payments, you use Stripe. If you think of recognizing things and images or video, you should use Clarifai. You shouldn't try to reinvent the wheel. You wouldn't want to write a database in-house, for example. We use RDS, which is hosted in Amazon. It has lots of benefits.

That's not the expertise in re-architecting the whole toolkit that serves neural networks. It's how to deploy very high-quality models, collect data and bring that as a very well-maintained and easy to use service for our customers.

Internally, actually when I founded the company in November 2013, the first thing I was working on was our own toolkit because this was multiple years ago before things like TensofFlow and Theano and Caffee were available and well supported. Now we are actually in this process where we are transitioning between an internal toolkit to one of these external toolkits so that we can get the benefits of the community as well.

[0:49:33.7] JM: Can you say which one you're using or do you not know yet?

[0:49:35.8] MZ: Yeah, we're going to use TensorFlow. I actually did a couple of internships at Google Brain so I know a lot of the engineers who wrote TensoFlow including Jeff Dean who is my intern host. That was a cool experience when I was there. I think they are going to do an awesome job at building something that is very efficient and scalable. That's what they are known for in that team.

That and I think the community is also kind of rallying around it. They have a big following out there and so that helps it grow and integrate with other types of things as well.

[0:50:08.9] JM: We've really seen the power of the community with Kubernetes and it seems like the same thing is going to happen with TensorFlow.

[0:50:15.1] MZ: Yup. Yeah. Exactly. That's why we actually, at Clarifai, we were the first people to add GPU support to Kubernetes and we contributed that back to the community in — I can't remember now, 1.3, or 1.4, release of Kubernetes. That was our code by a guy named Rudy here at Clarifai. We loved contributing back as well because we know it will come back around and help us in the future as well.

[0:50:42.9] JM: I know we are nearing the end of our time. A high-level question. We are talking about the identification of images and videos or things in video or images. How applicable is the technology were discussing? Actually, I know it's applicable but how far along are we in terms of the recreation stuff, because that's where it starts to get spooky and where we really have to rethink some of our social norms around digital stuff when you can replay somebody's voice or you can recreate somebody's visage in a video. Things start to get really crazy pretty fast.

[0:51:25.1] MZ: There's a lot of interesting progress under way in the research community. Actually, majority of my Ph.D. was actually focused on this, it's what we call unsupervised learning and generative models. Unsupervised means we don't need any labels in order to learn. And so what I was working on was just learning from images directly how to extract patterns and I did it in a way that would generate images and compare it to the original image and if it was close that was its learning metric. It had to be close to generating the original image.

It show promising results but people have since had a lot more progress in that regard. I just saw — I don't know who the author was for a paper, I just saw it on Twitter, where they converted a video of a horse running around to our video of zebra running around.

[0:52:13.2] JM: I saw that.

[0:52:14.5] MZ: That was pretty cool, because there's a lot of work on what people are calling generative adversarial networks and there are showing a lot of promise in being able to generate high-quality images and video and other signals like voice. I think there is a lot of promise and we're looking at it as well as to see kind of where the opportunity is there.

[0:52:34.4] JM: What about speech?

[0:52:36.2] MZ: Yeah. I think before image and video it was actually the first thing that really started to work with neural networks. Around 2012 were so, University of Toronto had some really good results and a few other places as well applying neural networks to the audio signal. Again, these convolutional neural networks turn out to work really well, another type of neural net, all the recurrent neural network, or along short-term memory recurrent neural network. There are some pretty crazy names in neural net terms.

They take into account time series and iterate using the same parameters over every time step. Those typically work well for speech recognition, and you're starting to see it in products everywhere; when you talk into your Siri device, your Google Now device, your Alexa device.

That’s all using speech recognition powered by neural networks. It’s out there, it’s working and it works pretty well.

[0:53:29.8] JM: Okay, Matt. I want to thank you for coming on Software Engineering Daily. It’s been a real educational conversation for me.

[0:53:36.0] MZ: Awesome, thanks for having me.

[END OF INTERVIEW]

[0:53:39.8] JM: Thanks to Symphono for sponsoring Software Engineering Daily. Symphono is a custom engineering shop where senior engineers tackle big tech challenges while learning from each other. Check it out at symphono.com/sedaily. That’s symphono.com/sedaily.

Thanks again to Symphono for being a sponsor of Software Engineering Daily for almost a year now. Your continued support allows us to deliver the this content to the listeners on a regular basis.

[END]