**AES\_ML\_Hawley\_Transcript (via Descript)**

[00:00:00] **Scott Hawley:** Hi everybody. I'm Scott Hawley. I'm at Belmont university where I teach mostly audio engineering students. And actually it was my first AES in 2013, where I first heard about machine learning when Elias Kokinas of Accusonus spoke right after me and blew my mind with some of the things they're doing today.

I'm going to talk about learning tuneable audio effects via neural networks with self attention. Our topic for today is learning to model audio effects, a couple of products out there that you may be familiar with. The Kemper Profiling Amplifier and Fractal Audio has their ax FX unit. And both of these systems can learn hard to replicate certain sounds. Fractal has a whole bunch of what we might call basis functions that it can adjust parameters for, and both of these systems are in heavy use. A recent webcast from Doug Doppler had top guitar techs from around the world, and you saw a lot of Fractals and Kempers.

But what if we want an effect that's not on the list. If we want to try to [00:01:00] learn effects, how general can those be? Could this be a new way of modeling effects and allow users to model their own effects or even invent new effects.

Some work that we've been doing on this in the area of black box modeling, we call the model SignalTrain for a trainable signal chain. And this was last year at AES, myself, Ben Colburn, and had a paper and then coming up next month, you'll get to hear from Billy Mitchell on some more work we've done. Basic ideas. You have some audio effects unit or physical process, and it has some maybe adjustable parameters on it. And you want to be able to put input audio into it and look at the output audio, and then try to mimic that by minimizing the difference between the output of some neural network with that.

And so we want to try to go for general audio effects, nonlinear time dependent effects. We want to be able to learn what these knobs are and what they do. And in this case, we're not so interested in making a generative [00:02:00] model, even though those are very interesting, we're more interested in capturing what this effect or process can do.

And just for the interest of time, there are other groups working on this Marco Martinez and people at queen Mary in particular. Lots of other people, just in the interest of time, I'm going to stick to what we've been doing. Basic model architecture is we have input in the wave form, domain and output in the wave form domain.

And in between we convert to spectrograms in terms of magnitude and phase, it starts off as a spectrogram, but actually we let that evolve as the neural network learns. And then the controls or the knob values. We stick those into the middle. These are auto encoders that are actually done as fully connected layers.

And then the original input and the output from these layers are merged in the case of phase, it's a skip result, digital connection. What we might call a wet, dry mix in audio and for the magnitude, it's a skip filter, which is [00:03:00] kind of like gain. So this has similarities with unit and wave net if you're familiar with those or wave unit, but there are some differences.

One of the things I should mention is that we have a, when I call look back buffer where we don't try to output the very beginning of the input audio. So we're taking kind of a random chunk from our source audio and then trying to match our prediction with the target. But we throw away the beginning in case there are time dependent effects like the decay of a compressor.

Some effects are harder than others. And if something's happening in the moment like distortion, that's usually pretty easy. In fact, four years ago, I got really excited about this work just from the different kinds of distortion that it could learn. Echoes, not too bad, but compressors were hard. And so we ended up spending a lot of time working on just compressors.

And more recently we started branching out into some other effects. There are lots of interesting effects to model. One of the issues we've been having is just trying to get the noise [00:04:00] down. So Billy's paper this year is going to be all about noise. And then also for practical audio production, we really do want to have.

CD quality or better sample rate, which is going to mean long sequences, which can be a challenge. Now, if you want to look at mine modeling of other effects, Marco at all have already done a lot of intro listing of facts without knobs on them. Again, I'm gonna stick to our work, a couple of other effects I haven't published about, but I've showed it talks are these sort of inverse effects, right?

Like the D lots of things. So you can send some good input and add noise and then reverse the order of them and get a tuneable de noiser or in particular, this thing can learn time alignment. So you can start with audio that's on the grid, mess it up, make it go off the grid, then reverse the input and output.

And the model will learn to put things back on the grid so we can do drum editing. So if you're late, it'll this line is on the grid. It'll bring it. On [00:05:00] now, if it's early, if you've ever done editing on your own, you know, you have to put in a cut and then move it back and then fill in the silence with say time stretching or something.

In this case, the model actually learns to synthesize more of what went before. So really excited about that. A few things just to keep in mind for what we're going to talk about next. This notion of having our foray transforms fully connected layers that operate globally, or at least on windows that can do entire chunks at once long sequences and time alignment.

So I promised I would try and help people in audio get up to speed with attention or self attention, which showed up in natural language processing. For language translation, image, captioning, and speech to text a lot of people in the audio world still find it confusing. So I'm going to go into that a little bit methods before attention would use it sort of encoder and decoder architecture, where you say you're translating English into French and all your words would go into this context [00:06:00] vector, and then they would get decoded.

And the problem was if you had really long sequences, maybe this context vector wasn't big enough to hang on to all the information. So the idea was. Let's connect up the inputs or the input States with these decoder States. And so you end up having a weighted mask that goes into the context vector, and you can hook this up to later decoder.

States as well. A lot of times you'll see these graphs showing, say input and output and the attention weights or showing the alignment between say words and French and words in English. They pretty much line up except for say adjectives, get reversed or in. Speech to text. There's a lot of good alignment and this is relevant because attention is essentially a soft differentiation version of alignment.

That's a statement by Lucas Kaiser, who is a coauthor on this other very important paper. Attention is all you need. [00:07:00] And their model is called transformer. So earlier models had added attention to some recurrent or convolutional architecture with Attention is all you need. They said, no, let's just use that this fully connected or feed forward attention, and then add some other things.

And when you do that in particular, taken out the recurrent neural networks, you can operate on whole chunks at a time in parallel, which makes it a lot faster. So a few features of this. Positional encoding. We'll talk about that next. We've got feet forward layers, skip connection. And, the multi-head we don't need to worry about that.

Not shown in this, our query key and value matrices, which we'll talk about a little bit later. So the positional encoding, once you have the fully connected layer in order to give it a sense of positional structure or sense of place, they added some extra what we might call channels to sort of encode where along the sequence things are.

And the particular way they chose to do it was using a set of sign and cosign functions, which [00:08:00] later when they end up showing up in dot products, ends up being akin to a form transformed. It's not exactly the same thing, but there are a lot of similar properties. One of the problems with these attention weights is let's say your sequence is N samples long.

So you're going end to end. You end up with N squared weights, which can get expensive because it goes as order and squared. So this year we've seen some models that attempt to mitigate that by either kind of breaking it down into windows or windows that kind of have some gaps in between or some combination thereof.

And when they do that, They can do long sequences, which for text, the largest size saw was 64,000 tokens in terms of CD quality audio. That ends up being about 1.5 seconds, which is good. But for example, in LA to a release times about five seconds now, one interesting thing I want to point out from earlier this summer, someone used one of these NLP models to do de reverberation, and I invite you to [00:09:00] check that out.

All right. So some similarities between the transformers and signal, train, chin, this positional encoding is kind of like a Fourier transform. We've got these fully connected layers operating globally or within windows entire chunks at once. Long sequences, different ways of doing that and time alignment.

Now, one thing that these trends forums will do, they'll have these Q K V vectors, which again, I think people in audio find confusing. I found it confusing, but essentially we're trying to answer the question. If attention is a weighted sum, how are we going to do the waiting? And so if the queue is the query is like our.

Possible output and the key and the value are related to our inputs or the hidden States on the encoder. Then what we do to get our attention scores is we take a dot product between the query and the key and use the transpose just to make the dimensions come out. Right. What you end up with is a square matrix with values between zero to one, that sum to one along rows.

In other words, these [00:10:00] are weights. So essentially you do summing on the values, which are like the inputs. Using those weights and that'll get you your context vector, and you can move on from there. So if you want to add that to signal, train, I found that a little confusing, but in may Francoise Luray posted this great attention toy one D model where he's doing something.

Kind of like audio sequences and showing different ways to manipulate that. And so I took what he did and converted it from convolutional layers that he was using to fully connected layers. And when we do well, what we find is actually, so red is the new line. Black is the old signal train. And with attention, the loss ends up being quite a bit lower.

As a function of how many iterations you do now, it still ends up at about the same noise value that Billy and I have been butting our heads up against. But yeah, it gets there faster. In fact, it gets there about three times faster. Overall it runs a little bit [00:11:00] slower per epoch, but in terms of E-box, it's about five times faster.

So it's a net win of about three times now, as far as that. Error or noise. You'll have to stick around for Billy's talk next month at AEs one 49, or ask me some questions during the Q and a also for people who are interested in learning how to do machine learning and audio Ryan Miller. And I maintain a page on GitHub called ML audio start that has links to tutorials and.

Online courses from lots of different people. And we've got contributions from people all over the industry. So I invite you to check that out and with that, thank you for your time. Enjoy the rest of the symposium.