Episode 17: Igniting the Next Generation of Deep Learning
Host: Nicole Huesman, Intel
Guests: Penporn Koanantakool, Google; Ramesh, Intel

Nicole Huesman: Welcome to *Code Together*, an interview series exploring the possibilities of cross-architecture development with those who live it. I'm your host, Nicole Huesman.

From natural language processing and image and object recognition to autonomous vehicles, fraud detection, medical diagnosis and treatment, and much more, deep learning applications surround us. Deep learning technology isn't new. It emerged in the 1950s. But increased processing power, massive amounts of data, and the development of more advanced algorithms has led to its rise.

Google and Intel are working together to accelerate deep learning applications. Here with us to talk about this collaboration is Penporn Koanantakool. As Senior Software Engineer at Google, Penporn is focused on applying high-performance computing techniques to help accelerate machine learning applications, primarily through tuning TensorFlow. She leads TensorFlow's performance optimization collaboration with Intel. Hi, Penporn!

Penporn Koanantakool: Hi, thank you for having me!

Nicole Huesman: And Ramesh. As Principal Engineer at Intel, Ramesh works on machine learning and deep learning performance optimizations. He also has deep expertise in computer vision-based applications, as well as compiler and optimization tools. Thanks for joining us, Ramesh!

Ramesh: Thanks, Nicole. Nice talking to you, Penporn.

Penporn Koanantakool: Nice talking to you, too!

Nicole Huesman: We're excited to hear how deep learning is advancing and what's on the horizon. I understand you're doing some incredible work to optimize TensorFlow.

Penporn Koanantakool: Yeah, so TensorFlow is an end-to-end open source machine learning platform. We have a rich ecosystem of tools and libraries that let researchers push the state-of-the-art in machine learning and also for developers to build and deploy machine learning applications. In our collaboration with Intel, we actually worked on two perpendicular efforts. So, the first one is optimizing the vanilla TensorFlow, and the second one is to make another build for Intel-optimized TensorFlow, or what we often call TensorFlow-MKL. Both of the builds use oneDNN to some extent to help optimize the performance.

We can start with vanilla TensorFlow first. So, the most expensive ops in neural networks are matrix multiplications and convolutions, and TensorFlow originally used Eigen to do that. Eigen has a fast matrix multiplication routine. To compute any matrix multiplication of convolutions, it actually splits those computations up to just many smaller matrix multiplications, and just calls that single optimized multiplication routine. But then that routine has a few downsides. It doesn't have as many optimizations for AVX-512 instructions, which many recent Intel processors have, and also it can only use the vector instructions that it is compiled with. For example, if the user compiled it with AVX-2, it wouldn't be able to use AVX-512 either. And that is kind of an issue for TensorFlow because, for TensorFlow Python package, we compile it with just AVX flag, just so users will be able to run it in the majority of the machines available today. So, it means that, prior to using oneDNN, TensorFlow can only take advantage of up to AVX instructions and not AVX-2 or AVX-512, which is available in many more Intel processors nowadays. So with that, in TensorFlow 1.13, we made Eigen call the fast matrix multiplication routine from oneDNN instead, and this routine is able to use a lot more AVX-512 instructions and also has dynamic dispatch, which means it can detect the CPU architecture at runtime and then generate code that is able to utilize all of the available instructions.

Episode 17: Igniting the Next Generation of Deep Learning
Host: Nicole Huesman, Intel
Guests: Penporn Koanantakool, Google; Ramesh, Intel

Ramesh: So from what I understand, you took the Intel® oneDNN library and integrated some parts of that into TensorFlow.

Penporn Koanantakool: Yeah. So in all the TensorFlow matrix multiplication and convolutions, it is calling oneDNN's matrix multiplication routine as a basic building block now. And because of that, even though TensorFlow is compiled with just AVX flag, but because oneDNN has the ability to dynamically dispatch and generate code, it actually can use AVX-512 or AVX-2.

Ramesh: And so customers should start seeing improved performance on some of their deep learning models with this change, right?

Penporn Koanantakool: Yeah. So we saw up to 30% speed-up by just switching from Eigen matrix multiplication to oneDNN matrix multiplication.

Ramesh: That's great. That's fantastic.

Penporn Koanantakool: The next one is the TensorFlow-MKL build. So that one actually has many more optimizations that oneDNN has to offer because vanilla TensorFlow only just calls one matrix multiplication routine from oneDNN, but oneDNN actually has many more functionalities than that, and the TensorFlow-MKL actually calls a lot of primitives from oneDNN, and we're able to use many more optimizations. This TensorFlow-MKL is the default TensorFlow build on Google Cloud Platform. For example, it is the default TensorFlow image in Deep Learning Virtual Machine images and Deep Learning Containers and in AI Platform Notebooks, and one of our teams are also using TensorFlow-MKL for their development.

For example, the DeepVariant project from the genomics team in Google Health. So DeepVariant is an open source tool for analyzing DNA sequences. And it is built on top of TensorFlow-MKL, which allows it to benefit from a lot of those AVX-512 optimizations that oneDNN has to offer. For example, one of the most compute-intensive steps in DeepVariant is called "call variants", which calls a convolutional neural network to classify whether positions in an input genome data differs from a reference genome. And that stage alone could take up to 14 hours on, in DeepVariant version 0.7 on a 64-core Intel Skylake machine on Google Cloud. That was without AVX-512 optimizations, but with AVX-512 optimizations from TensorFlow-MKL, the running time went down to just 3.5 hours, which means 3X speed-up, which for researchers is about not just being able to get their results three times faster, but it also means spending three times less costs on getting the results as well.

Ramesh: Yeah, that's an interesting project that we've been working on with Penporn and the team at Google. So for the last three or four years, we've been looking very closely to integrate oneDNN components, and last year with the launch of Skylake and then the launch of Cascade Lake, we've seen some significant improvements in performance using the work that we have done.

Penporn Koanantakool: I agree. Yeah. I think we can talk about the two most recent ones maybe with Cascade Lake and Cooper Lake.

So to give a bit of background, deep learning workloads are a little bit different from scientific computing workloads in a way that they actually need fewer numerical precision. So for example, in scientific applications, we usually need floating-point 64 bits or 32 bits to actually get good results. But for deep learning, a lot of times, 16 bits or 8 bits or even lower number of bits is enough. And when we have fewer number of bits means less storage and less memory requirement, and also faster compute.

Episode 17: Igniting the Next Generation of Deep Learning
Host: Nicole Huesman, Intel
Guests: Penporn Koanantakool, Google; Ramesh, Intel

In machine learning, there have been a lot of work that is trying to take advantage of using those fewer bits to accelerate deep learning. And I'm a fan of all the hardware support for deep learning accelerations that Intel has been adding to the recent hardware. So for example, Cascade Lake has these vector neural network instructions (VNNI) that can help accelerate int8 computations for quantization. In our work together, we saw about 3-4X speed-ups in image classification workloads, right, Ramesh?

Ramesh: Yes. That's right, Penporn. Yeah. So effectively we could get the best performance for the model using the VNNI instructions. The VNNI instructions are specifically designed to accelerate 8-bit computation, and by working with Google, we were able to modify TensorFlow to take full advantage of the instructions. And as we move forward, we are adding more instruction sets. This year, we released a Cooper Lake platform that includes special instructions to accelerate the 16-bit data types also known as bfloat16. And we worked very closely with Penporn and the team to incorporate that into TensorFlow.

Penporn Koanantakool: Yeah. So int8 can help with inference, but in training it's a little bit trickier to use integers, so in training, people still use floating point to do it, but we can still use fewer precision bits to do it in training. So what people are doing is using mixed precision by alternating between maybe 16-bit floating point and 32-bit floating point. And the bfloat16 format is the format that is developed by the Google Brain team that has the same amount of the exponent bits as the IEEE 32-bit floating-point format, which means it has the same dynamic range and is able to cover better weight range than the IEEE floating-point 16-bit format. And in this collaboration, users would just be able to use mixed precision right away by just calling Keras mixed precision API, and the framework would just automatically look into the models and see where it could convert the operations to use bfloat16 instead of float32 and do it all underneath. And the user would just see a speed up without needing to really change anything. And we ran it with a few different models, right?

Ramesh: Yeah. So I think we saw about 1.9X improvement in the performance of different models on Cooper Lake. And then I think one of the advantages of working with oneDNN was that a lot of these instruction sets and the optimizations are encapsulated by the library, so for us to integrate it into TensorFlow using oneDNN was a lot easier. We were able to incorporate this in just a couple of months to add support for bfloat16.

So Penporn, we have done some work along with Google to be able to take a model and convert it into int8 format. So can you talk about how that works with TensorFlow and how TensorFlow can be used for those models?

Penporn Koanantakool: Sure. Yeah. So I think for popular models, Intel already hosted some of the models that have been converted and tested by Intel. So customers can just download and use them. But for the custom models that they have, they actually can also just call your Intel AI quantization tool, right? So they just need to serialize the TensorFlow graph definition and then pass it to the tools and the tools will convert it from the floating-point 32 format to quantized graph. And then the user will need to feed in some training data to calibrate the quantized model and optimize it and then get the optimized graph and run it through TensorFlow-MKL.

Ramesh: And I think working with Google, we made this process so simple and easy to use for customers. And we are continuing to work with Google to make things much easier for our customers to use some of the more advanced instructions that we are adding to our process at Intel.

Episode 17: Igniting the Next Generation of Deep Learning
Host: Nicole Huesman, Intel
Guests: Penporn Koanantakool, Google; Ramesh, Intel

Nicole Huesman: That was really fantastic to hear about the different ways that the teams of Intel and Google are collaborating. So can we shift this a bit and can you guys talk about what's next, what's on the horizon?

Ramesh: So Intel is constantly improving our hardware to make sure AI runs better, and I'm excited to be working on the next-generation product. So you have codenamed Sapphire Rapids. Sapphire Rapids will have a new set of instructions using a matrix operation called the AMX instruction set. And we have just begun collaborating with Google on making sure that TensorFlow is optimized for the platform.

We are also working to enable Intel GPUs. We are working very closely with the Google team to be using the modular TensorFlow technology so we can have TensorFlow working on Intel GPUs combined with some of the optimizations that we are doing with oneDNN and DPC++.

And Penporn, can you talk a little bit more about the modular TensorFlow and how it's going to benefit Intel GPUs?

Penporn Koanantakool: Oh, sure. So modular TensorFlow is an effort to let third parties be able to connect to TensorFlow in a way that is more modular. So it means that it doesn't really need to depend on TensorFlow's code. We define a stable application binary interface, where everyone agrees upon, and then people just build their applications based on that and connect to TensorFlow through that. That gives the freedom for the developers to make changes anytime they want, as long as they maintain this application binary interface. And the way it's going to work is third-party devices just release their own library, and TensorFlow customers can just install those binary and then they would be able to call the device and use it right away. They will see it in TensorFlow.

Ramesh: Yeah, that's pretty exciting. So we are looking forward to being able to do an Intel plugin for the Intel GPU very soon.

Penporn Koanantakool: I'm really looking forward to that, yeah!

Nicole Huesman: So Ramesh and Penporn, where can our listeners go to learn more?

Ramesh: So there is an Intel website where we have links to all the different ways you can deploy a solution using TensorFlow. You can also visit the Google TensorFlow.org website for more information on how to use TensorFlow.

Penporn Koanantakool: You can also follow us on the official TensorFlow GitHub page as well. So if you have any issues or any new ideas, requests for features, you can post it on the GitHub or ask us any questions on Stack Overflow as well.

Nicole Huesman: It's been so great to hear how the teams at Google and Intel are helping advance deep learning. Deep learning is such an exciting and quickly evolving space. We look forward to having you both back on the program. Penporn, thanks so much for sharing your insights with us.

Penporn Koanantakool: Thanks so much again for having me!

Nicole Huesman: Ramesh, so great to have you with us today.

Ramesh: Thanks Nicole!

Nicole Huesman: For all of you listening, thanks so much for joining us. Let's continue the conversation at oneapi.com. Until next time!

Episode 17: Igniting the Next Generation of Deep Learning
Host: Nicole Huesman, Intel
Guests: Penporn Koanantakool, Google; Ramesh, Intel

Learn more:

Accelerating DeepVariant with Intel's AVX-512 Optimizations

TensorFlow-MKL int8 Optimizations for Cascade Lake

TensorFlow-MKL bfloat16 Optimizations for Cooper Lake