Patterson/Determined AI

Craig: Hi, I'm Craig Smith. And this is Eye on AI.

Last year I had Evans Sparks and Ameet Talwalkar on the podcast to talk about Determined AI, their startup, which aims to provide cutting edge deep learning software infrastructure for everyone.

Since then, in line with their mission to democratize machine learning infrastructure, they decided to open source their deep learning training platform in hopes that it will help machine learning engineers to more easily build better deep learning models. Ameet, a professor at Carnegie Mellon University and a leader in the emerging machine learning systems research community, will be joining me for the next five episodes to talk to some of his friends and colleagues about various aspects of the machine learning pipeline, including data preparation, model development and training, hardware management and deployment. We begin the series with Turing Award winner, David Patterson, one of the world's foremost experts in semiconductor architecture.

Dave talked about the end of Moore's law and the evolution of computer chips from general purpose CPUs to GPUs to the recent Cambrian explosion of specialized AI chips, including Google's TPU.

I hope you find the conversation as enlightening as I did.

Dave, can you start by introducing yourself, where you are now, where you studied, your work history and your Turing Award, and how you came to work on chip sets for AI.

Dave: Hi, I'm Dave Patterson. I'm here in sunny Santa Cruz, California as we're recording this. I was a professor at UC Berkeley for 40 years, and then four years ago started working at Google in the Brain division, where we help out with building hardware for machine learning. I got all of my degrees at UCLA, Bachelors, Masters, and PhD. And so how did I end up working on machine learning? I needed to cut down my work week as a professor, so I retired and then I knew people at Google and they thought this would be a good match for me. I knew zero about machine learning, but a lot about how to design computers. And so, they thought I could contribute there.

Craig: The Turing Award. Can you tell us a little bit about the work that led to the Turing Award?

Dave: Sure. So, in my first five years at Berkeley, I got involved in a project where we were trying to figure out what's the best way to design microprocessors. So, when software talks to hardware, it has vocabulary. And we call that vocabulary an "instruction set architecture." And since that's so many words, you'll hear it abbreviated "ISA." And since that's three syllables, some people even pronounce it as "Isa" to make it shorter. But in the 1970s, the trend was to make these vocabularies richer and richer in polysyllabic words. I and my colleague at Stanford, John Hennessy, and also people at IBM led by John Koch, had a contrarian view. And particularly for Hennessy and I, we were interested in microprocessors. And given Moore's law, which was in full effect in that we could have twice as many transistors every two years, we thought, with such a fast-changing technology, we wanted a vocabulary that was easy to build. The technical question there was, these sophisticated instructions, you wouldn't need as many of them. But then, if they're a simple vocabulary, as you can imagine, if they're just monosyllabic words, maybe you could read them faster. And so, the bottom line was that we need a third or maybe one and a half times as many instructions of the sophisticated ones to run a program, but we could run them five times faster. So, the net effect was a factor of three or four better.

So that became known as "reduced instruction set computers," that it was an instruction set of reduced words, or these monosyllabic words, you can think of it. And also, Hennessy and I later wrote a textbook because we were dissatisfied with the way computer architecture was being taught. And the Turing Award, which is the Nobel Prize for computer sciences -- we don't have one, this is the highest award -- was given for our contributions to reduced instruction set computers, and for the textbook, which explained how to build computers out of a more quantitative approach.

Ameet: And I can attest to the popularity of the textbook and obviously to the impact of the research. I remember having dinner with Dave when we were in Boston once. And somebody just interrupted us for dinner saying that they learned about computer architecture from his book. And Dave was a little bit unfazed, I think, because this happens to him.

Dave: Yeah. I'm a celebrity in weird circles.

Ameet: David's done a bunch of amazing things, but one, important thing, maybe it wasn't related to the Turing Award, but it's had a big impact on Berkeley and probably other places as well.

The way he does research and the way he organized research was pretty nonstandard at the time and set a new precedent at Berkeley And I know this firsthand, one of the many labs that he helped start was where I did my postdoc and was super influential for me and it was an incredible experience in part because of the way it was organized.

Dave: Do you want me to talk about that

Craig: Yeah, absolutely.

Dave: Well, one of my good friends was on the faculty at Berkeley at the time, John Ousterhout, and I hit upon this way to explain things and it's important for grad students to learn how to give good talks. So, the first thing I did with his inspiration was give a talk on how to give a bad talk. And that went over really well. A lot of people enjoyed it. And so, then I later gave a talk on how to have a bad career. That basically gave me a chance to reflect and see how we did things. And my view is that research usually doesn't work out. It's research. Right? Even products don't work out all the time. So, research, only occasionally does it work.

So, for me, if that was true, then what makes sense in my career was to do multiple projects, to give myself more times at bat. Give myself a better chance of hitting a home run by giving more at bats. And also, it struck me that in universities, graduate students are there about five years. So, we decided to do five-year projects. We would come up with a vision, work on it and at the end of five years, we would throw a party. And that would be the end of the project, and we'd move on to the next one.

So, that became the game plan. The alternative way to do this was to bet on one thing for your whole lifetime and no matter what happens with technology, stick with it because, like fashion, it'll come around again. Like bell bottoms are popular. Bell bottoms were popular in the seventies and have come back. So, we patterned that in my career at Berkeley where we do these research labs. We get a bunch of faculty together from different disciplines. We look at a technology, see if there's some new trends going on that'll lead for some new opportunity, create that vision, work on it for five years, and then wrap it up.

The other thing that seems clear to me if you're trying to do difficult things, you should get feedback from other people, honest feedback, to see how well you're doing. So, part of that five-year lab model was to have two offsite retreats a year where we would invite people from industry to give us feedback on our projects. So twice a year, we got three-day meetings and they would tell us what they liked and what they didn't like. And part of that was, at the very end of these three-day retreats, everybody from industry would tell us what they thought. And to hear the feedback we actually had to install a rule that you're not allowed to argue with feedback. You can ask clarifying questions, and I would rule if the question was a clarifying question. I had somebody censor my questions. So, you can't ask a question, "Are you guys are idiots?" No, that's not a clarifying question.

But particularly at universities where everybody has tenure, they're not used to hearing honest feedback, they don't have to hear it and they don't have to react to it. But when people you respect from industry, when two or three of them all say the same thing, that gives you pause. So that was the game plan. The projects tended to overlap, but in my 40 years, maybe there was 10 overlapping five-year projects. And it's really great for grooming grad students because they got to see the projects, and they got to give talks at retreats, and understand what real problems are, getting the feedback on the research, many of them have picked up some of those ideas and have used them in their careers. So, I think that's what Ameet was talking about.

Ameet: Right. and the one that I was part of, the AMPLab was, like many other ones, a very successful project, very interdisciplinary. Brought in systems people and ML people, and really helped shape my career. It also helped shape our startup, and many of these students also went on to become professors at top universities and they were really well trained in giving talks, communicating with different people, and so on.

Dave: And many went also into industry, and interestingly, the same kind of skill set of trying to figure out a problem: how do you attack a problem? What's a good problem to work on? And how does he work as a team to make progress on it? Those skills have been pretty useful as well, for the people, no matter where they went.

MUSIC: MUSIC

Craig: Okay. Dave we're here to talk about the development of specialized chips for AI.

I thought maybe you could start by, giving us a history of chips, a very high level, and how, CPUs were found to have limitations. Google saw that problem and the "tidal wave coming," as you called it once, and developed the TPU version 1. Could you give us that story?

Dave: Sure. So, what people call chips is just integrated circuits. Gordon Moore, in 1965, we were just starting to put multiple transistors into the same integrated circuit. That's why it's called an integrated circuit. And he projected that for the, what turned out for 50 years, we'd get this doubling of the number of transistors per chip. So that's a chip. And what a microprocessor is, is just a computer inside a chip. So, once we had enough transistors to build a whole computer, we had the microprocessor. Intel's microprocessor, back in the 70s, was probably the first microprocessor. Well, we've been riding that wave for decades because of that rapid change. And for a long time, computers got twice as fast every 18 months driven by Moore's Law. Moore's Law started slowing down recently. One of the issues was for the amount of power that you could air cool, you just couldn't put that many more transistors. They couldn't go that much faster. And it's just hard to manufacture without slowing down. So that's kind of a general trend.

Ameet: I've heard you talk about that over the last five years or so. When would you say things started slowing down?

Dave: That's about right, five years ago. The New York Times actually officially declared when Moore's Law ended. They had a date, and the date was the announcement of Google's TPU. And their argument was, well, if Google has to build their own chips rather than just buy them from Intel, where Gordon Moore founded Intel, then it's over. If even a software company like Google is building chips, then it's over.

So, why did Google decide to build its own chips? Google was early in this neural network revolution. Jeff Dean, and maybe others, were excited about this and so they were early adopters of neural network. Really believed that deep neural networks was going to change the world. And they had an early app; it might've been translation or it might've been a vision recognition one. But in 2014, they did a calculation that if this app, using deep neural networks running on standard microprocessors, if a hundred million of their customers uses this app for three minutes a day, Google in 2014 would have to double the number of data centers and all of the microprocessors to handle that load.

Even Google couldn't afford it, and it would have taken years to build twice as many data centers and fill them up with microprocessors. So, they said, "Oh my God," had kind of an emergency Manhattan Project that, "We have to build something that's going to run deep neural networks much more efficiently than standard microprocessors."

So that was the goal. They started the project 2014, "build a microprocessor just for deep neural networks, that's at least a factor of 10 more effective than standard microprocessors." They called it the Tensor Processing Unit. And since they've done, I guess, four generations now, that one's now called TPU v.1 for "version one."

Craig: You've called that the "Helen of Troy of AI hardware that launched a thousand chips," I liked that.

Dave: Yeah. So, if we go back in time, it wasn't so much that Google did it. It's that when Google announced it, the CEO of Google in May 2016, said at an annual Google event where they brag about things, "Oh, by the way, we decided to build our own hardware for deep neural networks. And it's more than a factor of 10 better than anything you can buy." And it was kind of like a 60 second presentation with one slide, but that was like an earthquake in Silicon Valley. It shocked Intel. Intel subsequently bought four companies, spending billions of dollars. It shocked venture capitalists. They subsequently invested $2 billion in at least a hundred startups. Said, "If Google's got this already, this must be a new opportunity." And that announcement that just changed the course of investments in computer hardware. Before that time, people said, "Well, hardware investments don't make a lot of sense. They take too long, they're too expensive. We should do software startups." But there were many hardware startups that were embraced as a result of that.

The seismic event within machine learning was when the guys at Toronto won the ImageNet competition.

Ameet: The first kind of foreshadowing of it was folks at Google at ICML 2012. They trained massive neural networks and were trying to predict cats. And then a few months later at NeurIPS, that was AlexNet.

Dave: That was AlexNet just a few months later.

Ameet: Dave, how would you say the impact has been on Nvidia?

Dave: Nvidia wanted to get people to start using GPUs for broader classes of applications than just graphics. So, for several years, they've been building software tools to make that easier to do. GPU stand for "graphic processing units." They are special purpose microprocessors for computer graphics. If you play video games or if you go to the movies and see CGI, that's computer graphics using GPUs. Now, to do computer graphics, you do a lot of floating point to do these images and using a GPU.

So, it's pretty cheap because they'd have to stick them into computers that people can play games, but they have a lot of computation power. to compute with it. you had to pretend you were creating an image. And you create a stupid image to get your computation. So, it was very awkward. So, they tried to make it much easier to use. And they were hoping the scientific computing community would use them, the high-performance computing community. That was the goal.

But at Toronto, one of the people who did the AlexNet effort had taken one of these courses on how to program a GPU for regular computation. So, he put the two and two together, that deep neural networks and matrix multiplies. "I can do this with a GPU!" And he used the GPU. So once AlexNet changed the accuracy rates, won the competition, and within a couple of years, everybody started using deep neural networks. Just like the people at Toronto, they also used GPUs, which were pretty cost effective for this particular type of scientific computing. And so, GPUs became very popular for deep neural networks. Particularly, the two phases are training, where you create the model, and then either inference or serving, where after the model is created you use it. GPUs were particularly good for training.

Craig: GPUs, you mentioned matrix multiplication. A matrix is essentially a 2D tensor, is that right?

Dave: Yeah. It's an array. Rows and columns.

Craig: And so, matrix multiplication is fundamental to neural networks. Can you talk about the importance of matrix multiplication, and then how a GPU can handle that, and then how the TPU was built specifically for that?

Ameet: Yeah. I can talk about that first part, where matrices and tensors come in. Let's consider a simple example, say a convolutional neural network. A convolutional neural network is the standard neural network that people use for images, for instance. So, if we walk through how that works, just very briefly, your input is going to be an image. That is a matrix. There's three channels, cause there's different color channels. But imagine even you just had a single set of pixels that you were representing with your input. That would be the input matrix. A convolution operation, forgetting about the details, it walks through this two by two matrix and multiplies and performs some additions, and it spits out another two by two matrix. At a particular layer, instead of having one convolution, you might have multiple filters. And so, the output of any one layer of a network ends up being kind of a stack of these two by two matrices, which itself is a tensor. And this process repeats over the different layers, which is why you talk about tensors, but it's effectively just a series of matrices that you're operating.

Craig: And the operation is primarily multiplication?

Ameet: Yeah. Typically, a series of multiplies and adds.

Craig: And then how would that be handled by a GPU as opposed to a TPU?

Dave: Let me start off and say how it's done in a standard microprocessor, which we often call -- these "PU" names go back to IBM and the mainframe. They call it the central processing unit or CPU. Then, the riff was graphics processing unit, GPU, and then tensor processing unit, TPU. So, on a standard microprocessor, or CPU, you'll have one arithmetic unit. Basically. And so, you've got all this hardware on the chip, but there's kind of only one thing that's doing computer arithmetic at a time. If you think of a matrix that you're operating on, say it's 256 by 256, there's 65,000 multiplies you have to do. It would kind of do them one at a time on a CPU. Maybe you could do four at a time, but not a lot.

The GPU had more arithmetic units, and particularly for one of the data types that you need for machine learning, it's called floating point data types. And so, they had a few more units, maybe they had 10 times as many units as a CPU. And so, it was kind of like, maybe it's a factor of 10 faster for this, or factors of 5 or 10. The GPUs are also much more expensive than CPU. So, it's faster, but more expensive.

Ameet: And is it right, Dave, that a key point here is that these operations, for instance, in the context of deep learning, are easily parallelizable? So, if you have 10 of them, instead of one of them, you could do all 10...

Dave: It's really because it's matrix multiplying. Matrix multiplying, the sizes of the matrices that you're dealing with can be arranged to be 256 by 256, or 1024 by 1024 1024. the matrix, these are big enough that naturally the computation is 65,000 or a million of these individual arithmetic operations.

Computer architects call that data parallel. There's lots of operations to work on. So, what they did for the TPU was say, "Well, if matrix multiply is king, we're going to put 65,000 arithmetic units on it in an array of 256 by 256. So, we're going to process tensors 256 by 256 elements at a time." Which is kind of an astounding number of arithmetic units. So, it was the first chip that was really designed for machine learning. People started using GPUs in 2015 or so, but they were still optimized for graphics. It wasn't until several generations later that Nvidia realized there was this big market and started optimizing their designs for machine learning rather than just for graphics.

Ameet: That was exactly my understanding, that initially people were using GPUs, they were repurposing them even though they were meant for video games. But more recently they had also been specialized for deep learning.

Dave: Nvidia tends to bring out a brand-new GPU microprocessor about every two years, and they give them names after scientists. So, the first one that had support for machine learning was called Volta. And then the one they just announced, Ampere, also has support for machine learning. But the ones before that, which I think was, Pascal was really just what was ever in the pipeline and was available for sale, but, was unaware of the importance of machine learning.

Craig: And then is the future of GPUs versus TPUs?

Dave: Well, because of the Helen of Troy chip, we have many more people building accelerators than just Google and Nvidia. So, in terms of Nvidia versus Google and the GPU versus TPU, Google only makes them for their internal use. And also, they make them available in the cloud. So, you can rent them, but you can't buy them. Nvidia sells to everybody. And also, people buy them and put them in the cloud. Google buys some GPUs and puts them in the cloud.

But so, does Alibaba and Amazon and Microsoft all buy GPUs from Nvidia and stick them there. So, Google's is more restrictive, but it's part of this really interesting thing that's going on in the industry with these cloud companies, their building so many data centers with so many chips on them, they're starting to design some chips by themselves, just for their own internal use. So, Google did this with TPU v.1. Both Alibaba and Amazon imitated Google to build custom machine learning chips for only their internal use, only inside the cloud. You can't buy those externally.

Ameet: You can rent them, you're saying, but you can't buy them.

Dave: Right. Google did it for internal reasons, to support its products. If you use Google products, you're using TPUs indirectly, and you're using a lot of machine learning. A recent example that I personally enjoy is on Google Documents they're doing real time grammar checking, which is amazing. And that's based on machine learning and using TPUs internally. But then because they have a cloud offering, they're also offering it there. So that's kind of the business differences.

Within Google, I think Nvidia chips have a reputation of being very expensive. Not the things you put in your computers for gaming, but if you get the version that goes in data centers, those are pretty expensive. So that's one of the reasons Google decided to do it themselves rather than just buy chips from Nvidia. And then Google has put features in that Google thinks is really good for its production applications. But all of this, Helen of Troy, everybody's got an idea of what would be really good in this new era. I refer to it as the Cambrian age for computer architecture, where we've got dozens of companies with very different bets about the best way to design hardware for machine learning. So, it's a really exciting time if you're a computer designer, you get all these opportunities to do things. And we don't really know, which is the best way. Historically the way we solved this is use the marketplace, which is an imperfect decider, but in computer architecture, you can come up with this idea, you convince the company to spend hundreds of millions or billions of dollars, put it in the marketplace, and you see if you're right or not. And we're going to have this war to settle, which one's the best architecture in the marketplace over this next 5 to 10 years.

Ameet: Do you think that it's going to end up being one winner? Or could it end up being something where the set of hardware is diverse enough that for different applications, different machine learning applications, and so on, that different types of architecture, there's tradeoffs.

Dave: Nobody knows. I mean, I think what would be great from a research perspective is, now they've got all these engineers slaving away, building the best different kinds of architecture they can, all trying to run the same programs. There's the opportunity to examine all these experiments and see if there's some underlying principles here that we can try and understand better how to build better machines in the future, given all these ideas that are being tried.

Craig: Just on vocabulary, you used the word "accelerator" earlier, and when we're talking about TPUs and GPUs, we're talking about domain specific architectures, is that right? And that's what you mean by accelerator? They're all the same, we're talking about chips. Whether you say accelerator or...

Dave: Yeah. I think most computer architects think today, we can't make general purpose computers run much faster. We don't have any ideas that are suddenly going to make this like the good old days of doubling performance every 18 months because of the slowing Moore's Law, and we just can't figure out another way to do it. So, I think the only path forward is things that accelerate individual tasks. And what we're calling that is domain specific architectures, like they're aimed at one domain to do it. Completely serendipitously, just as things were getting bad in computer architecture, machine learning came along. And here is this potentially general purpose, brand new way of computing that, if we accelerate just that, that may be a very broad market.

So, what is happening right now, is we are transitioning from programmers starting from first principles to write programs top down, to bottom up from data, using neural networking algorithms to create applications. If that's the transition that's happening, that's perfect timing because we've run out of steam, how to do general purpose computation. And there's still, hopefully, lots of ideas that people can try to use to accelerate deep neural networks.

MUSIC: MUSIC

Craig: Can you talk then about RISC-V and how that fits into this story?

Dave: So, I told you the Turing Award was based on 1980s reduced instruction set computers. Well, just exactly a decade ago at Berkeley for the research that was going on in one of these research labs that Ameet was talking about, the faculty and some of the students decided we needed to build new microprocessors, particularly for parallel computing, because of the slowing downs of Moore's Law.

And at that time, in 2010, we thought we were going to need to build accelerators. We thought accelerators are going to be the future and we needed a vocabulary to build. So even though this lab was sponsored by Intel and Intel had, at the time, the most popular vocabulary, the most popular instruction set, they owned it and they didn't want us to use it. In fact, if any academic used that instruction set, they get a letter telling them to stop using it because Intel owned it. The other popular instruction set architecture at the time was ARM. Same story. They would license it to you, and if you want to pay millions of dollars, you can use it. But if you don't have a license, you can't use it.

So, we decided in 2010 to build our own instruction set. In the 1980s, I had done four RISC projects. And so, they decided, in honor of that work, we'll call it RISC-V. And since it's 20 or 30 years later, we learned from all the mistakes of the RISC process in the 1980s and built a nice, clean instruction set architecture called RISC-V, intending it to use at Berkeley, to teach our courses and our research. And also, if there are other academics in the same shoes, that we would make it available to them. And so, we made it open and free for everybody to use.

Craig: And just to jump in, RISC stands for...

Dave: Reduced instruction set computer. So, it's the simpler vocabulary one. So, one of the things that Hennessy and I are surprised about, about getting the Turing Award a few years ago, is the best ideas for that vocabulary are still RISC. That of all the things that happened in 30 years, you would have thought something might have happened and that would no longer be a good idea, but it's still the best idea we got to go forward.

Ameet: If the instruction sets haven't changed all that much, what is the reason that an Intel is not willing to share their instruction set?

Dave: So, the way we distribute software normally, especially for personal computers, is I told you software talks to hardware in an instruction set. Well, we distribute software in those instructions. So, it's number of representations of those instructions. They call it binary, but it's basically all the software for PCs is distributed in Intel's instruction set using Intel's instructions. So that's a very valuable instruction set architecture. So other people would love if Intel didn't make a restriction. We'd like to build microprocessors, too, that copy that instruction set. And Intel says, "No thanks. We're going to keep the monopoly, since many people like that instruction set architecture, that interface between software and hardware, that's very valuable. If you want to run that instruction set, you want to run those software binaries, you have to buy microprocessors from us.

Ameet: I don't want to, in any way, minimize what you guys are doing with RISC-V, but to some extent, is it that you're trying to establish a new shared vocabulary that everybody could use? So, people are all speaking the same language and doing it in an open way, such that there's no proprietary concerns or no bottlenecks from any industry partners.

Dave: That's exactly right. And in fact, right now, what I'm doing is I and my granddaughter, who's a film student at Mills, are doing a 10th anniversary video. So, we've interviewed all the pioneers. And one person said it really well: "If we do this right, since computers are going to be everywhere, it doesn't make sense, if computers are going to be everywhere, that the language that you speak to computers is proprietary. The analogy would be some firm in London owns English. And not only do they own it and you have to pay a fee to use it, they control how it changes. They decide what words are in or out, and you can't change it. You have to follow their rules." So that's the analogy here. There's a lot of excitement that we finally are breaking out of the proprietary ones. We looked at our colleagues in software, there's lots of open source software that's very successful, like Linux. And so, we think RISC-V is going to be the Linux of hardware, and there's a lot of people excited about that vision.

Ameet: So, the idea is that it is open source in the sense that it will change over time, but it'll change in a community agreed upon way?

Dave: The vocabulary itself is like a standard, like ethernet. So, it's an open standard, so you don't have to pay any fees to use. The chips that you get follow that standard. So, the source code that describes the chips, that's what's open. But you need an open specification to be able to have open source chips. It's a small point. Most people just say it's an open source instruction set and architects will kind of cringe a little bit, but it's close enough.

Ameet: I see.

Craig: But beyond it being open source, you were talking earlier, you're using the analogy of monosyllabic versus polysyllabic to increase the speed and simplicity. You're intending it to be a better instruction set. Is that right?

Dave: The big deal is the business model, or the freedom to innovate. That's the big thing. It actually is an excellent instruction set, and it's got some novel ideas. All the software that runs on it is required to run on a very small subset of it. Only 40 instructions, where like the Intel instruction set and the ARM instruction set has thousands of instructions. So that makes it very easy to build, which is both great educationally, and if you have some new technology, it's easy for it to take advantage of it. So, it's got some very intriguing benefits. But the biggest deal is that it's an industrial strength open architecture which has led to open source implementations.

Ameet: Who benefits from this in the future? Is it this idea that all of these startups that are creating new chips, they can all adopt RISC-V and...?

Dave: Yes. Particularly if you're a startup, before RISC-V, what it meant was, "My startup involves hardware. I'm going to do, say, a special chip for AI." Well, step one is you would enter into contract negotiations with ARM to try and negotiate: how much do you have to pay for it? What's the royalty rate going to be? Which version are you going to do? And if you wanted to say, "Okay, I want to be compatible with the ARM specification, the ARM instruction set architecture, but I want to sign my own version of that. I don't want to license yours."

They say, "Oh, that's okay. That'll be $25 million. To have the right to design it." And then it's the time to sign the contract and the costs. With RISC-V, you can say, "Okay, I'm using RISC-V," and you automatically get one of these architecture licenses so you can build it yourself. And you can also get them from other people. Either universities or other companies are, like open source software, are putting designs out there, industrial strength designs, that you can just grab and use. So, it's a huge time reduction as well as a cost reduction. Plus, it has this other freedom, it lets you extend the vocabulary for your application. Which, in certainly Intel, they're the ones that do those extensions. But here, anybody can do that now. And so, a classic thing that our people are doing for AI is using the standard instruction set for the normal software. And then, however they want to attack the area of machine learning they want to use, they'll add their own special instructions to it, to do it really well.

Ameet: I see. So, RISC-V started before this whole new deep learning explosion. I imagine it's been a pretty good timing for RISC-V though, right?

Dave: Well, the story was, we started getting complaints about us changing our instruction set for our use at Berkeley from students in India. And it was like, "Why do you care if we change our instruction set for use in our courses? Why are you complaining?"

"Cause we're trying to use it!" And when we talk to these people in India, because the stuff had been on the web and they were following it, we didn't know they were following it. And then we realized there's this need for an open architecture. Lots of people want to use it. And so, in 2014, we tried to go full bore, let people know about it. We wrote a paper called "Instruction Sets Should Be Free: The Case for RISC-V." And then that kind of took off.

There was a person who was building a product, and he read the paper and he changed to RISC-V. And he read it in October and he had a product in December. And people in Switzerland doing open source hardware, were using an inferior instruction set, and they switched over. And then, ever since 2014, it's been growing very rapidly.

Ameet: I see. So, are some of the AI startups that are making hardware leveraging RISC-V, then?

Dave: Yes. Well, that's easy to kind of say if there's a hundred of them. I would say, philosophically, for the standard software, you just want a standard risk instruction set. For your secret sauce, you want to do that yourself. And so, most of the computation, it'll be in the machine learning piece. And that's in your own unique instructions your unique hardware, kind of like Google has done for TPUs. But there's still a lot of standard software you have to run, user interfaces and stuff like that. And then RISC-V is fine for that.

Craig: Can you talk a little bit about where these domain specific architectures are going, and specifically some of the commercial companies, Graphcore and Cerebras, which has this wafer scale integration? And are any of them using RISC-V?

Dave: The companies you mentioned are tackling the training, which is the really computation heavy piece of it. And I don't think any of them are using RISC-V. Training is often done in the cloud, so they're aimed at being cloud compatible.

And I used the word accelerator. So, they typically, they'll take a standard Intel microprocessor that runs a current set of software and they'll add this accelerator to the Intel microprocessor, has been the way they've been doing it so far. The RISC-V approach is much more popular on what's called "the edge." You know, the devices closer to the people. Like if it's in light bulbs, or shoes, or things like that, or eventually maybe mobile phones. But closer to the people, that's where RISC-V is much more popular. The data center is dominated by the Intel instruction set. And so, people have been adding accelerators to that.

Ameet: Do you think that might also change over time or...

Dave: Yes. In that paper, "Instruction Sets: The Case For RISC-V" we pointed out that one of the issues of proprietary instruction sets, is that instruction set's tied to the fate of that company. Well, the big deal that's in the news right now is ARM, which is the dominant instruction set for the edge, was bought. It was a British company bought by this Japanese company called SoftBank, and SoftBank, according to newspaper reports, is putting it up for sale. And Nvidia is apparently one of the prime targets to take it over. So, if you're a rival of Nvidia and you're using ARM, this is concerning. What's going to happen if Nvidia buys ARM, and you get ARM? Like Apple, all the iPhone products are going to be using ARM. Suppose your rival, Nvidia, owns ARM. What's going to happen?

So, a bunch of people who felt like, "Well, RISC-V has many potential technical advantages, but I don't know if I want to take that business risk to try this new architecture." Suddenly, "Oh my God! Maybe what we want to do is have an architecture that's supported by an open foundation, not something that a company owns, because we're not sure what's going to happen if it's owned by a rival. This was a theoretical benefit of RISC-V, but it's in the newspapers right now. And so, there are executives at companies in the cloud who are trying to decide, "What should we do? ARM's for sale.

Whoever buys ARM, how is that going to impact us? And maybe we want to move out of proprietary instruction sets for the reasons that [Krste Asanovic]() and Patterson wrote in 2014."

Craig: Can you talk then a little bit about where you see hardware going? Certainly, with open architecture, but also with new domain specific applications?

Dave: Well, like I said, we're forced to change. Moore's Law is slowing down. We can't just build things the way we have. So, we're forced to innovate. But this was kind of resurfacing, in this domain specific architecture, a lot of ideas that failed for general purpose computing.

Now, part of that is when Ameet talked about this switching over to programming for machine learning. We're programming in much higher-level constructs than in C or C++. You're operating at a whole tensor at a time in languages like TensorFlow and PyTorch. So that raises the level of abstraction that lets us innovate below that. So, many ideas are being tried right now, and we need to wait for this to be sorted out the winner of the most radical approach is Cerebras. Cerebras has resurfaced an idea from the antiquities. It's actually something I worked on while I was a graduate student.

The way we build chips today is like a pizza. It's a big round thing, it's made out of silicon, and it's got lots of layers on it. But it's a wafer about the size of pizza, like 12 inches in diameter. And that gets turned into chips by chopping it up into little tiny pieces, like the size of your thumbnail.

So, from a pizza size, then you'd end up with, say, 500 little tiny chips. And you put them in the packages and that's what we call chips or microprocessors. Forever, people have said, "That's kind of a crazy way to do things. We already have the chips on this pizza, which we call a wafer. Why don't we just connect the chips on the wafer and that'd be more efficient?" So that idea is called wafer scale integration. And it's been around for 50 years as an idea. And people have tried it over the decades to build a company around it and have gone out of business. That did not discourage Cerebras. Cerebras has resuscitated wafer scale integration, and they are selling, right now, a wafer as one chip. Now what they actually do, it really is a round pizza, but they've chopped off the rounded corners and get a rectangle that fits exactly into a 12-inch pizza. And that's what they're selling. And this is radical ways to build computers. And they believe they're getting an advantage by building it that way.

That's a lot of heat in one small area, very hot pizza that they have to cool so that it can operate, but they've tackled those problems. From an engineer's perspective, what an interesting thing to pull off. You're rooting for them to succeed. There's so many problems they have to solve to pull that off, but that's one example of the Helen of Troy, thousand chips there. That's just one of the radical ideas that are going on. If that turns out to be the very best way to do it, and there's a significant advantage over what other people are doing, then the industry could change that way. If it just turns out to be a, "Yeah, you can do it that way. There's a bunch of negatives as well as some positives, so we're going to keep taking these wafers, dicing them up to chips, and building them the way we always have." And that's one of the ideas that hundreds of millions of dollars at least are being invested to see if that's viable or not.

MUSIC: MUSIC

Craig: Dave, do you think there's any point that we missed? We're trying to point this toward the ML pipeline, should we say something about...

Dave: Oh yeah. Benchmarks, I think.

Ameet: Yeah, benchmarks.

Dave: So, in computer architecture, the way we pick winners and losers is we agree to all run the same programs and that's called a benchmark. And that tends to be a suite, not just one program, but a suite of 5 or 10 programs. And because billions of dollars are involved. There's serious marketing efforts. So, you don't mind losing to your colleagues at another institution provided their microprocessor is actually better than yours, and the way we agree on that is benchmarks. So up until recently, there was no benchmarks in machine learning. And so, Google and, Stanford and Berkeley and Harvard and Baidu launched something that's called [MLPerf]().

Ameet: Dave is being humble here. I learned about benchmarking from him, again, when I was at Berkeley overlapping with him. He taught me that benchmarks, for better or worse, define a field.

My understanding is that, in the field of computer architecture, I vaguely remember you saying something about, "Moore's Law was really important, but establishing benchmarks, which you were heavily involved with, was perhaps equally important in terms of getting these companies to compete, but in a way that was productive for everybody."

Dave: The book I talked about early on that Hennessy and I had done, the title captures it. It's "Computer Architecture: A Quantitative Approach." And the quantitative piece was because that computer architecture is taught descriptively, qualitatively. "So, here's all the features this one has. Here's all the features this one has." Like a catalog. And we tried to put formulas and quantitative, and we needed benchmarks to be able to pull that off.

Computer architecture is a very benchmark driven field. And so, we needed that, for machine learning, to be able to, if we're having this Helen of Troy event, Cambrian Age, how are we going to compare all these things? Especially with all these startups. So that's consciously why MLPerf was created.

So early on, there was some startups, they would present the results and I'd ask them, where's your MLPerf results? And they said, "Oh, you know, we'll get to it eventually." And some of those are now out of business. They had these really fascinating technologies. They'd explain it to the investors, they'd get the money. But when push came to shove, when you ran the benchmarks, their MLPerf scores weren't all that great. So that's going to help sort this out. If you're in the ML community, hopefully you know about MLPerf. MLPerf is novel in that they have two divisions, they have a closed division, so that we get fair comparisons between these commercial organizations. But there's also an open division that's intended for researchers and academics to show us better ways to solve the same program. So, you take the same dataset. You have the same goals. But if you've got a better algorithm or anything else, a better platform, better numerics, and you can show it's better, then that'll probably get the attention of a lot of companies involved. So, we're hoping that MLPerf will both push machine learning forward faster, as well as computer architectures for machine learning.

Craig: And then we've been talking about, Determined AI's open source platform for composing and training machine learning models. How does the hardware relate? I mean, one of the things that the platform does is manage compute resources.

Ameet: Yeah. So, I think a big bet of our company is exactly related to this Cambrian explosion that Dave was talking about. In the same way that for RISC-V, being the Switzerland of instruction sets is nice, for an organization that's working with deep learning, it would be nice if you could kind of abstract the decision of what hardware to be using. Because different hardware is going to emerge over time, there's going to be winners and losers that you can't really predict. If Dave can't predict who the winner is going to be, nobody can, right? He's a leading expert in this area.

And so, being able to allow users to get the most out of their hardware, even as that hardware changes, and easily be able to not be locked into one piece of hardware versus the other, one cloud versus the other, this sort of thing is something that we are banking on. And it's something that our software is able to manage. We support CPUs, TPUs, GPUs. Over time, we can support different types of hardware and allow our customers kind of seamlessly to move between different types of hardware. And perhaps, if there's more than one winner, optimize on a per workload basis for whatever hardware is most appropriate.

Dave: The training winner might come from a different company from the inference winner. Do you do both, inference and training?

Ameet: We're focused more on the training aspect of things. We're actually talking to TQ. He helped create TVM, and he has a startup as well. They're focused more on the inference side, at least right now. But again, banking on the same sort of argument of, "How do you help make high level or abstracted workloads actually run well on different pieces of new hardware? I guess that's the piece in between the hardware and things like CUDA, right? So, there's CUDA that also helps make GPUs run fast, so you need that same sort of library of operations for different pieces of hardware.

Dave: To me what's fascinating is how training is just going after supercomputers, or where people are building supercomputers with training. And this GPT-3 results are, as far as I can tell, bigger. Bigger is better. That's the lesson of it. And so, if you're going to start building supercomputers to push the state of the art of machine learning, that's right in the middle of what computer architects like to do. And how do you port that stuff to different platforms? And it's going to be really expensive, right? If you're going to build 400 billion parameter models and computers to run that, being able to run that more efficiently, what's that going to cost? Millions of dollars? Will it cost millions of dollars to train things?

Ameet: There's a question of what hardware to use during training and inference. There's also the question of how to train models at training time, such that they can possibly work in whatever inference setting you have. You talked about, on the edge, if your models need to work on a light bulb or even on a cell phone, a several billion-parameter model might just not cut it. How do you make things smaller? How do you distill these models? And ideally, can you do it with knowledge of what you're doing at training time so that you can kind of connect the training in the inference component?

Dave: You guys talk about quantization as well?

Ameet: So, an area, and this is sort of stuff I know about from research, as well as an application of neural architecture search is in these constrained environments. So not only, "how do I learn an architecture?" But, "How do I learn an architecture if I know it's constraints would be a certain size or to have a certain latency or a certain bandwidth.

But I think that the goal moving forward is going to be to train models in an "inference aware" sort of setting, you know where you want to use these models. Maybe it's not one model. It can be this parade or frontier of models. Here are different models that satisfied different constraints. And there's a tradeoff between, say, accuracy and those constraints. And somehow the way you're constraining them is in terms of size, but also things like discretization and pruning and a bunch of different complementary techniques. and I don't think that's a solved problem, but I think even thinking about how to do inference efficiently, some of that work is certainly going to fall or should fall in the training aspect of things.

Because what we see right now is often people train models. Even if they work, they say, "You know what? It's hard enough to get a model that works well on this new application. I'm going to worry later about how to do inference on it." But then that's a huge headache in and of itself, right? Your kind of decoupling these two problems is not necessarily the smartest way to go about it.

Dave: Well, having a portable platform in this for the Cambrian age, that sounds like you're in a good spot.

Ameet: Yeah. What we have seen thus far, it seems like just talking to customers, GPU is really our king right now. After GPUs it is TPUs, people are starting to get really excited about them. Everything else seems like, everyone knows that these things exist. We hear a little bit about it, but the explosion hasn't occurred yet. Right? It's like it's bubbling up. I'm certain it will happen.

Dave: Are the cloud their favorite? Or...

Ameet: There's a real trade off. I mean, cloud is still a lot more expensive. But if you're on-prem, if you're not kind of utilizing your hardware well, then it's just sitting there and not working. Even though the cloud has obviously gotten better in terms of security issues, for some applications and some organizations, they much prefer on-prem. I would say it's changed over the last few years, though.

We have a blog post about this that is one of the most commonly read blog posts on our company blog. I'd say two, three years ago, the numbers were far more in favor of open source. I think pricing has gone down a little bit on the cloud. There's no one winner yet, but over time, I do think the cloud will get more and more mature and more and more people will use it. But even then, there is this argument you made of, if I want TPUs I have to use GCP. If I want to use something else proprietary from Amazon, I have to use AWS.

Craig: That's it for this week's podcast. I want to thank Dave for his time. If you want to learn more about the things we talked about today, you can find a transcript of this podcast on our website, eye-on.ai.

Ameet and I love to hear from listeners. so, if you have comments or suggestions, please feel free to contact us at info@eye-on.ai. And remember, the singularity may not be near, but AI is about to change your world.

So, pay attention.