This week I welcome on the show two of the most important technologists ever, in any field.

Jeff Dean is Google's Chief Scientist, and through 25 years at the company, has worked on basically the most transformative systems in modern computing: from MapReduce, BigTable, Tensorflow, AlphaChip, to Gemini.

Noam Shazeer invented or co-invented all the main architectures and techniques that are used for modern LLMs: from the Transformer itself, to Mixture of Experts, to Mesh Tensorflow, to Gemini and many other things.

We talk about their 25 years at Google, going from PageRank to MapReduce to the Transformer to MoEs to AlphaChip – and maybe soon to ASI.

My favorite part was Jeff's vision for Pathways, Google's grand plan for a mutually-reinforcing loop of hardware and algorithmic design and for going past autoregression. That culminates in us imagining *all* of Google-the-company, going through one huge MoE model.

And Noam just bites every bullet: 100x world GDP soon; let's get a million automated researchers running in the Google datacenter; living to see the year 3000.

Listen on Apple Podcasts or Spotify; watch on Youtube.

Sponsors

Timestamps

Transcript

Dwarkesh Patel *00:00:00*

Today I have the honor of chatting with Jeff Dean and Noam Shazeer. Jeff is Google's Chief Scientist, and through his 25 years at the company, he has worked on basically the most transformative systems in modern computing: from MapReduce, BigTable, Tensorflow, AlphaChip – genuinely, the list doesn't end – Gemini now.

And Noam is the single person most responsible for the current AI revolution. He has been the inventor or co-inventor of all the main architectures and techniques that are used for modern LLMs: from the Transformer itself, to Mixture of Experts, to Mesh Tensorflow, to many other things. And they are two of the three co-leads of Gemini at Google DeepMind. Awesome. Thanks so much for coming on.

Jeff Dean *00:00:50*

Thank you. Super excited to be here.

Dwarkesh Patel *00:00:52*

Okay, first question.-Both of you have been at Google for 25, or close to 25, years. At some point early on in the company, you probably understood how everything worked. When did that stop being the case? Do you feel like there was a clear moment that happened?

Noam Shazeer *00:01:08*

I joined, this was like, end of 2000, and they had this thing: everybody gets a mentor. I knew nothing. I would just ask my mentor everything, and my mentor knew everything. It turned out my mentor was Jeff.

It was not the case that everyone at Google knew everything. It was just the case that Jeff knew everything because he had basically written everything.

Jeff Dean *00:01:33*

You're very kind. I think as companies grow, you kind of go through these phases. When I joined, we were 25 people, 26 people, something like that. So you eventually you learned everyone's name, and even though we were growing, you kept track of all the people who were joining.

At some point, you lose track of everyone's name in the company, but you still know everyone working on software engineering things. Then you lose track of all the names of people in the software engineering group, but you at least know all the different projects that everyone's working on. Then at some point, the company gets big enough that you get an email that Project Platypus is launching on Friday, and you're like, "What the heck is Project Platypus?"

Noam Shazeer *00:02:15*

Usually it's a very good surprise. You're like, "Wow, Project Platypus!" I had no idea we were doing that.

Jeff Dean *00:02:23*

But I think it is good to keep track of what's going on in the company, even at a very high level, even if you don't know every last detail. And it's good to know lots of people throughout the company so that you can go ask someone for more details or figure out who to talk to. With one level of indirection, you can usually find the right person in the company if you have a good network of people that you've built up over time.

Dwarkesh Patel *00:02:44*

How did Google recruit you, by the way?

Jeff Dean *00:02:46*

I kind of reached out to them, actually.

Dwarkesh Patel *00:02:50*

And Noam, how did you get recruited?

Noam Shazeer *00:02:53*

I actually saw Google at a job fair in 1999, and I assumed that it was already this huge company, that there was no point in joining, because everyone I knew used Google. I guess that was because I was a grad student at Berkeley at the time. I guess I've dropped out of grad programs a few times.

It turns out that actually it wasn't really that large. It turns out that I did not apply in 1999, but just kind of sent them a resume on a whim in 2000, because I figured it was my favorite search engine, and figured I should apply to multiple places for a job. But then it turned out to be really fun, it looked like a bunch of smart people doing good stuff. They had this really nice crayon chart on the wall of the daily number of search queries that somebody had just been maintaining. It looked very exponential. I thought, "These guys are going to be very successful, and it looks like they have a lot of good problems to work on." So I was like, "Okay, maybe I'll go work there for a little while and then have enough money to just go work on AI for as long as I want after that."

Dwarkesh Patel *00:04:08*

Yeah, yeah. In a way you did that, right?

Noam Shazeer *00:04:10*

Yeah, it totally worked out exactly according to plan.

Dwarkesh Patel *00:04:15*

You were thinking about AI in 1999?

Noam Shazeer *00:04:17*

Yeah, this was like 2000. Yeah, I remember in grad school, a friend of mine at the time had told me that his New Year's resolution for 2000 was to live to see the year 3000, and that he was going to achieve this by inventing AI. I was like, "Oh, that sounds like a good idea."

I didn't get the idea at the time that you could go do it at a big company. But I figured, "Hey, a bunch of people seem to be making a ton of money at startups. Maybe I'll just make some money, and then I'll have enough to live on and just work on AI research for a long time." But yeah, it actually turned out that Google was a terrific place to work on AI.

Jeff Dean *00:05:07*

One of the things I like about Google is our ambition has always been sort of something that would require pretty advanced AI. Because I think organizing the world's information and making it universally accessible and useful, actually there is a really broad mandate in there. It's not like the company was going to do this one little thing and stay doing that. And also you could see that what we were doing initially was in that direction, but you could do so much more in that direction.

Dwarkesh Patel *00:05:36*

How has Moore's Law over the last two or three decades changed the kinds of considerations you have to take on board when you design new systems, when you figure out what projects are feasible? What are still the limitations? What are things you can now do that you obviously couldn't do before?

Jeff Dean *00:05:51*

I think of it as actually changing quite a bit in the last couple of decades. Two decades ago to one decade ago, it was awesome because you just wait, and like 18 months later, you get much faster hardware, and you don't have to do anything. And then more recently, I feel like the general-purpose CPU-based machine scaling has not been as good, like the fabrication process improvements are now taking three years instead of every two years. The architectural improvements in multi-core processors and so on are not giving you the same boost that we were getting 20 to 10 years ago. But I think at the same time, we're seeing much more specialized computational devices, like machine learning accelerators, TPUs, and very ML-focused GPUs, more recently, are making it so that we can actually get really high performance and good efficiency out of the more modern kinds of computations we want to run that are different than a twisty pile of C++ code trying to run Microsoft Office or something.

Noam Shazeer *00:07:02*

It feels like the algorithms are following the hardware. Basically, what's happened is that at this point, arithmetic is very, very cheap, and moving data around is comparatively much more expensive. So pretty much all of deep learning has taken off roughly because of that. You can build it out of matrix multiplications that are N cubed operations and N squared bytes of data communication basically.

Jeff Dean *00:07:39*

Well, I would say that the pivot to hardware oriented around that was an important transition, because before that, we had CPUs and GPUs that were not especially well-suited for deep learning. And then we started to build TPUs at Google that were really just reduced-precision linear algebra machines, and then once you have that then you want to exploit it.

Noam Shazeer *00:08:02*

It seems like it's all about identifying opportunity costs. Like, okay, this is something like Larry Page, I think, used to always say: "Our second biggest cost is taxes, and our biggest cost is opportunity costs." If he didn't say that, then I've been misquoting him for years.

But basically it's like, what is the opportunity that you have that you're missing out on? In this case, I guess it was that you've got all of this chip area, and you're putting a very small number of arithmetic units on it. Fill the thing up with arithmetic units! You could have orders of magnitude more arithmetic getting done.

Now, what else has to change? Okay, the algorithms and the data flow and everything else.

Jeff Dean *00:08:51*

And, oh, by the way, the arithmetic can be really low precision, so then you can squeeze even more multiplier units in.

Dwarkesh Patel *00:08:58*

Noam, I want to follow up on what you said, that the algorithms have been following the hardware. If you imagine a counterfactual world where, suppose that the cost of memory had declined more than arithmetic, or just invert the dynamic you saw.

Noam Shazeer *00:09:12*

Okay, data flow is extremely cheap, and arithmetic is not.

Dwarkesh Patel *00:09:18*

What would AI look like today?

Jeff Dean *00:09:20*

You'd have a lot more lookups into very large memories.

Noam Shazeer *00:09:25*

Yeah, it might look more like AI looked like 20 years ago but in the opposite direction. I'm not sure. I guess I joined Google Brain in 2012. I left Google for a few years, happened to go back for lunch to visit my wife, and we happened to sit down next to Jeff and the early Google Brain team. I thought, "Wow, that's a smart group of people."

Jeff Dean *00:09:55*

I think I said, "You should think about deep neural nets. We're making some pretty good progress there."

Noam Shazeer *00:09:59*

"That sounds fun." Okay, so I jumped back in…

Jeff Dean *00:10:02*

I wooed him back, it was great.

Noam Shazeer *00:10:05*

..to join Jeff, that was like 2012. I seem to join Google every 12 years: I rejoined Google in 2000, 2012, and 2024.

Dwarkesh Patel *00:10:14*

What's going to happen in 2036?

Noam Shazeer *00:10:16*

I don't know. I guess we shall see.

Dwarkesh Patel *00:10:21*

What are the trade-offs that you're considering changing for future versions of TPU to integrate how you're thinking about algorithms?

Jeff Dean *00:10:29*

I think one general trend is we're getting better at quantizing or having much more reduced precision models. We started with TPUv1, and we weren't even quite sure we could quantize and model for serving with eight-bit integers. But we sort of had some early evidence that seemed like it might be possible. So we're like, "Great, let's build the whole chip around that."

And then over time, I think you've seen people able to use much lower precision for training as well. But also the inference precision has gone. People are now using INT4 or FP4, which sounded like, if you said to someone like we're going to use FP4, like a supercomputing floating point person 20 years ago, they'd be like, "What? That's crazy. We like 64 bits in our floats."

Or even below that, some people are quantizing models to two bits or one bit, and I think that's a trend that definitely –

Dwarkesh Patel *00:11:25*

One bit? Just like a zero-or-one?

Jeff Dean *00:11:27*

Yeah, just a 0-1. And then you have a sign bit for a group of bits or something.

Noam Shazeer *00:11:33*

It really has to be a co-design thing because, if the algorithm designer doesn't realize that you can get greatly improved performance, throughput, with the lower precision, of course, the algorithm designer is going to say, "Of course, I don't want low precision. That introduces risk." And then it adds irritation.

Then if you ask the chip designer, "Okay, what do you want to build?" And then they'll ask the person who's writing the algorithms today, who's going to say, "No, I don't like quantization. It's irritating." So you actually need to basically see the whole picture and figure out, "Oh, wait a minute, we can increase our throughput-to-cost ratio by a lot by quantizing."

Jeff Dean *00:12:27*

Then you're like, yes, quantization is irritating, but your model is going to be three times faster, so you're going to have to deal.

Dwarkesh Patel *00:12:33*

Through your careers, at various times, you've worked on things that have an uncanny resemblance to what we're actually using now for generative AI. In 1990, Jeff, your senior thesis was about backpropagation. And in 2007- this is the thing that I didn't realise until I was prepping for this episode – in 2007 you guys trained a two trillion token N-gram model for language modeling.

Just walk me through when you were developing that model. Was this kind of thing in your head? What did you think you guys were doing at the time?

Jeff Dean *00:13:13*

Let me start with the undergrad thesis. I got introduced to neural nets in one section of one class on parallel computing that I was taking in my senior year. I needed to do a thesis to graduate, an honors thesis. So I approached the professor and I said, "Oh, it'd be really fun to do something around neural nets."

So, he and I decided I would implement a couple of different ways of parallelizing backpropagation training for neural nets in 1990. I called them something funny in my thesis, like "pattern partitioning" or something. But really, I implemented a model parallelism and data parallelism on a 32-processor Hypercube machine.

In one, you split all the examples into different batches, and every CPU has a copy of the model. In the other one, you pipeline a bunch of examples along to processors that have different parts of the model. I compared and contrasted them, and it was interesting.

I was really excited about the abstraction because it felt like neural nets were the right abstraction. They could solve tiny toy problems that no other approach could solve at the time. I thought, naive me, that 32 processors would be able to train really awesome neural nets.

But it turned out we needed about a million times more compute before they really started to work for real problems, but then starting in the late 2008, 2009, 2010 timeframe, we started to have enough compute, thanks to Moore's law, to actually make neural nets work for real things. That was kind of when I re-entered, looking at neural nets.

But prior to that, in 2007...

Dwarkesh Patel *00:14:55*

Sorry, actually could I ask about this?

Jeff Dean *00:14:57*

Oh yeah, sure.

Dwarkesh Patel *00:14:58*

First of all, unlike other artifacts of academia, it's actually like four pages, and you can just read it.

Jeff Dean *00:15:07*

It was four pages and then 30 pages of C code.

Dwarkesh Patel *00:15:10*

But it's just a well-produced artifact. Tell me about how the 2007 paper came together.

Jeff Dean *00:15:15*

Oh yeah, so that, we had a machine translation research team at Google led by Franz Och, who had joined Google maybe a year before, and a bunch of other people. Every year they competed in a DARPA contest on translating a couple of different languages to English, I think, Chinese to English and Arabic to English.

The Google team had submitted an entry, and the way this works is you get 500 sentences on Monday, and you have to submit the answer on Friday. I saw the results of this, and we'd won the contest by a pretty substantial margin measured in Bleu score, which is a measure of translation quality.

So I reached out to Franz, the head of this winning team. I'm like, "This is great, when are we going to launch it?" And he's like, "Oh, well, we can't launch this. It's not really very practical because it takes 12 hours to translate a sentence." I'm like, "Well, that seems like a long time. How could we fix that?"

It turned out they'd not really designed it for high throughput, obviously. It was doing 100,000 disk seeks in a large language model that they sort of computed statistics over – I wouldn't say "trained" really – for each word that it wanted to translate.

Obviously, doing 100,000 disk seeks is not super speedy. But I said, "Okay, well, let's dive into this." So I spent about two or three months with them, designing an in-memory compressed representation of N-gram data.

We were using- an N-gram is basically statistics for how often every N-word sequence occurs in a large corpus, so you basically have, in this case, we had 2 trillion words. Most N-gram models of the day were using two-grams or maybe three-grams, but we decided we would use five-grams.

So, how often every five-word sequence occurs in basically as much of the web as we could process in that day. Then you have a data structure that says, "Okay, 'I really like this restaurant' occurs 17 times in the web, or something.

And so I built a data structure that would let you store all those in memory on 200 machines and then have sort of a batched API where you could say, "Here are the 100,000 things I need to look up in this round for this word," and we'd give you them all back in parallel. That enabled us to go from taking a night to translate a sentence to basically doing something in 100 milliseconds or something.

Dwarkesh Patel *00:18:03*

There's this list of Jeff Dean facts, like Chuck Norris facts. For example, that “for Jeff Dean, NP equals "no problemo."” One of them, it's funny because now that I hear you say it, actually, it's kind of true. One of them is, "The speed of light was 35 miles an hour until Jeff Dean decided to optimize it over a weekend." Just going from 12 hours to 100 milliseconds, I got to do the orders of magnitude there.

Jeff Dean *00:18:36*

All of these are very flattering. They're pretty funny. They're like an April Fool's joke gone awry by my colleagues.

Dwarkesh Patel *00:18:45*

Obviously, in retrospect, this idea that you can develop a latent representation of the entire internet through just considering the relationships between words is like: yeah, this is large language models. This is Gemini. At the time, was it *just* a translation idea, or did you see that as being the beginning of a different kind of paradigm?

Jeff Dean *00:19:11*

I think once we built that for translation, the serving of large language models started to be used for other things, like completion... you start to type, and it suggests what completions make sense.

So it was definitely the start of a lot of uses of language models in Google. And Noam has worked on a number of other things at Google, like spelling correction systems that use language models.

Noam Shazeer *00:19:36*

That was like 2000, 2001, and I think it was all in-memory on one machine.

Jeff Dean *00:19:44*

Yeah, I think it was one machine. His spelling correction system he built in 2001 was amazing. He sent out this demo link to the whole company.

I just tried every butchered spelling of every few-word query I could get, like “scrumbled uggs Bundict"—

Noam Shazeer *00:19:59*

I remember that one, yeah yeah.

Jeff Dean *00:20:00*

—instead of “scrambled eggs benedict”, and it just nailed it every time.

Noam Shazeer *00:20:04*

Yeah, I guess that was language modeling.

Dwarkesh Patel *00:20:07*

But at the time, when you were developing these systems, did you have this sense of, “look, you make these things more and more sophisticated, don't consider five words, consider 100 words, 1,000 words, then the latent representation is intelligence”. Basically when did that insight hit?

Noam Shazeer *00:20:24*

Not really. I don't think I ever felt like, okay, N-gram models are going to–

Jeff Dean *00:20:32*

–sweep the world–

Noam Shazeer *00:20:33*

–yeah: “be” artificial intelligence. I think at the time, a lot of people were excited about Bayesian networks. That seemed exciting.

Definitely seeing those early neural language models, both the magic in that, “okay, this is doing something extremely cool” and also, it just struck me as the best problem in the world in that for one, it is very, very simple to state: give me a probability distribution over the next word. Also, there's roughly infinite training data out there. There's the text of the web; you have trillions of training examples of unsupervised data.

Jeff Dean *00:21:20*

Yeah, or self-supervised.

Noam Shazeer *00:21:22*

Self-supervised, yeah.

Jeff Dean *00:21:23*

It's nice because you then have the right answer, and then you can train on all but the current word and try to predict the current word. It's this amazing ability to just learn from observations of the world.

Noam Shazeer *00:21:36*

And then it's AI complete. If you can do a great job of that, then you can pretty much do anything.

Dwarkesh Patel *00:23:00*

There's this interesting discussion in the history of science about whether ideas are just in the air and there's a sort of inevitability to big ideas, or whether they're sort of plucked out of some tangential direction. In this case, this way in which we're laying it out very logically, does that imply basically, how inevitable does this...

Noam Shazeer *00:22:05*

It does feel like it's in the air. There were definitely some, there was like the neural Turing machine, a bunch of ideas around attention, like having these key-value stores that could be useful in neural networks to focus on things. I think in some sense, it was in the air, and in some sense, you need some group to go do it.

Jeff Dean *00:22:36*

I like to think of a lot of ideas as being partially in the air, where there are a few different, maybe separate research ideas that one is squinting at when you’re trying to solve a new problem. You draw on those for some inspiration, and then there's some aspect that is not solved, and you need to figure out how to solve that. The combination of some morphing of the things that already exist and some new things lead to some new breakthrough or new research result that didn't exist before.

Dwarkesh Patel *00:22:57*

~~A~~re there key moments that stand out to you where you're looking at a research area, you come up with this idea, and you have this feeling of, "Holy shit, I can't believe that worked?"

Jeff Dean *00:23:06*

One thing I remember was in the early days of the Brain team. We were focused on “let’s see if we could build some infrastructure that lets us train really, really big neural nets”. At that time, we didn't have GPUs in our data centers; we just had CPUs. But we know how to make lots of CPUs work together.

So we built a system that enabled us to train pretty large neural nets through both model and data parallelism. We had a system for unsupervised learning on 10 million randomly selected YouTube frames. It was a spatially local representation, so it would build up unsupervised representations based on trying to reconstruct the thing from the high-level representations.

We got that working and training on 2,000 computers using 16,000 cores. After a little while, that model was actually able to build a representation at the highest level where one neuron would get excited by images of cats. It had never been told what a cat was, but it had seen enough examples of them in the training data of head-on facial views of cats that that neuron would turn on for that and not for much else.

Similarly, you'd have other ones for human faces and backs of pedestrians, and this kind of thing. That was kind of cool because it's from unsupervised learning principles, building up these really high-level representations. Then we were able to get very good results on the supervised ImageNet 20,000 category challenge that advanced the state of the art by 60% relative improvement, which was quite good at the time.

That neural net was probably 50x bigger than one that had been trained previously, and it got good results. So that sort of said to me, "Hey, actually scaling up neural nets seems like, I thought it would be a good idea and it seems to be, so we should keep pushing on that."

Dwarkesh Patel *00:25:14*

These examples illustrate how these AI systems fit into what you were just mentioning: that Google is fundamentally a company that organizes information. AI, in this context, is finding relationships between information, between concepts, to help get ideas to you faster, information you want to you faster.

Now we're moving with current AI models. Obviously, you can use BERT in Google Search and you can ask these questions. They are still good at information retrieval, but more fundamentally, they can write your entire code base for you and do actual work, which goes beyond just information retrieval.

So how are you thinking about that? Is Google still an information retrieval company if you're building an AGI? An AGI can do information retrieval, but it can do many other things as well.

Jeff Dean *00:26:14*

I think we're an "organize the world's information" company, and that's broader than information retrieval. Maybe: “organizing and creating new information from some guidance you give it”.

"Can you help me write a letter to my veterinarian about my dog? It's got these symptoms," and it'll draft that. Or, "Can you feed in this video, and can you produce a summary of what's happening in the video every few minutes?"

I think our multimodal capabilities are showing that it's more than just text. It's about understanding the world in all the different modalities that information exists in, both human ones but also non-human-oriented ones, like weird lidar sensors on autonomous vehicles, or genomic information, or health information.

And then, how do you extract and transform that into useful insights for people and make use of that in helping them do all kinds of things they want to do? Sometimes it's, "I want to be entertained by chatting with a chatbot." Sometimes it's, "I want answers to this really complicated question, there is no single source to retrieve from." You need to pull information from 100 web pages, figure out what's going on, and make an organized, synthesized version of that data.

Then dealing with multimodal things or coding-related problems. I think it's super exciting what these models are capable of, and they're improving fast, so I'm excited to see where we go.

Noam Shazeer *00:28:42*

I am also excited to see where we go. I think definitely organizing information is clearly a trillion-dollar opportunity, but a trillion dollars is not cool anymore. What's cool is a quadrillion dollars.

Obviously the idea is not to just pile up some giant pile of money, but it's to create value in the world, and so much more value can be created when these systems can actually go and do something for you, write your code, or figure out problems that you wouldn't have been able to figure out yourself.

To do that at scale, we're going to have to be very, very flexible and dynamic as we improve the capabilities of these models.

Jeff Dean *00:29:22*

Yeah, I'm pretty excited about a lot of fundamental research questions that come about because you see something could be substantially improved if we tried this approach or things in this rough direction. Maybe that'll work, maybe it won't.

But I also think there's value in seeing what we could achieve for end-users and then how can we work backwards from that to actually build systems that are able to do that. As one example: organizing information, that should mean any information in the world should be usable by anyone, regardless of what language they speak.

And that I think we've done some amount of, but it's not nearly the full vision of, "No matter what language you speak, out of thousands of languages, we can make any piece of content available to you and make it usable by you. Any video could be watched in any language." I think that would be pretty awesome. We're not quite there yet, but that's definitely things I see on the horizon that should be possible.

Dwarkesh Patel *00:30:26*

Speaking of different architectures you might try, I know one thing you're working on right now is longer context. If you think of Google Search, it's got the entire index of the internet in its context, but it's a very shallow search. And then obviously language models have limited context right now, but they can really think. It's like dark magic, in-context learning. It can really think about what it’s seeing.

How do you think about what it would be like to merge something like Google Search and something like in-context learning?

Jeff Dean *00:30:51*

Yeah, I'll take a first stab at it because —I've thought about this for a bit. One of the things you see with these models is they're quite good, but they do hallucinate and have factuality issues sometimes. Part of that is you've trained on, say, tens of trillions of tokens, and you've stirred all that together in your tens or hundreds of billions of parameters.

But it's all a bit squishy because you've churned all these tokens together. The model has a reasonably clear view of that data, but it sometimes gets confused and will give the wrong date for something.

Whereas information in the context window, in the input of the model, is really sharp and clear because we have this really nice attention mechanism in transformers. The model can pay attention to things, and it knows the exact text or the exact frames of the video or audio or whatever that it's processing.

Right now, we have models that can deal with millions of tokens of context, which is quite a lot. It's hundreds of pages of PDF, or 50 research papers, or hours of video, or tens of hours of audio, or some combination of those things, which is pretty cool. But it would be really nice if the model could attend to trillions of tokens.

Could it attend to the entire internet and find the right stuff for you? Could it attend to all your personal information for you? I would love a model that has access to all my emails, all my documents, and all my photos.

When I ask it to do something, it can sort of make use of that, with my permission, to help solve what it is I'm wanting it to do. But that's going to be a big computational challenge because the naive attention algorithm is quadratic. You can barely make it work on a fair bit of hardware for millions of tokens, but there's no hope of making that just naively go to trillions of tokens.

So, we need a whole bunch of interesting algorithmic approximations to what you would really want: a way for the model to attend conceptually to lots and lots more tokens, trillions of tokens. Maybe we can put all of the Google code base in context for every Google developer, all the world's source code in context for any open-source developer. That would be amazing.

Noam Shazeer *00:33:20*

It would be incredible. The beautiful thing about model parameters is they are quite memory-efficient at memorizing facts. You can probably memorize on the order of one fact or something per model parameter.

Whereas if you have some token in context, there are lots of keys and values at every layer. It could be a kilobyte, a megabyte of memory per token.

Jeff Dean *00:33:56*

You take a word and you blow it up to 10 kilobytes or something.

Noam Shazeer *00:33:59*

Yes. There's actually a lot of innovation going on around, okay, A, how do you minimize that? And B, what words do you need to have there? Are there better ways of accessing bits of that information?

Jeff seems like the right person to figure this out. Okay, what does our memory hierarchy look like from the SRAM all the way up to data center worldwide level?

Dwarkesh Patel *00:34:32*

I want to talk more about the thing you mentioned: look, Google is a company with lots of code and lots of examples. If you just think about that one use case and what that implies, so you've got the Google monorepo. Maybe you figure out the long context thing, you can put the whole thing in context, or you fine-tune on it. Why hasn't this been already done?

You can imagine the amount of code that Google has proprietary access to, even if you're just using it internally to make your developers more efficient and productive.

Jeff Dean *00:35:09*

To be clear, we have actually already done further training on a Gemini model on our internal code base for our internal developers. But that's different than attending to all of it because it sort of stirs together the code base into a bunch of parameters. Having it in context makes things clearer.

Even the further trained model internally is incredibly useful. Sundar, I think, has said that 25% of the characters that we're checking into our code base these days are generated by our AI-based coding models with kind of human oversight.

Dwarkesh Patel *00:35:49*

How do you imagine, in the next year or two, based on the capabilities you see around the horizon, your own personal work? What will it be like to be a researcher at Google? You have a new idea or something. With the way in which you're interacting with these models in a year, what does that look like?

Noam Shazeer *00:36:04*

Well, I assume we will have these models a lot better and hopefully be able to be much, much more productive.

Jeff Dean *00:36:15*

Yeah, in addition to kind of research-y context, anytime you're seeing these models used, I think they're able to make software developers more productive because they can kind of take a high-level spec or sentence description of what you want done and give a pretty reasonable first cut at that. From a research perspective, maybe you can say, "I'd really like you to explore this kind of idea similar to the one in this paper, but maybe let's try making it convolutional or something."

If you could do that and have the system automatically generate a bunch of experimental code, and maybe you look at it and you're like, "Yeah, that looks good, run that." That seems like a nice dream direction to go in.

It seems plausible in the next year or two years that you might make a lot of progress on that.

Dwarkesh Patel *00:38:08*

It seems under-hyped because you could have literally millions of extra employees, and you can immediately check their output, the employees can check each other's output, hey immediately stream tokens.

Jeff Dean *00:38:21*

Sorry, I didn't mean to underhype it. I think it's super exciting. I just don't like to hype things that aren't done yet.

Dwarkesh Patel *00:38:34*

I do want to play with this idea more because it seems like a big deal if you have something kind of like an autonomous software engineer, especially from the perspective of a researcher who's like, "I want to build the system." Okay, so let's just play with this idea. As somebody who has worked on developing transformative systems through your careers, the idea that instead of having to code something like whatever today's equivalent of MapReduce is or Tensorflow is, just like, "Here's how I want a distributed AI library to look. Write it up for me."

Do you imagine you could be 10x more productive? 100x more productive?

Jeff Dean *00:39:13*

I was pretty impressed. I think it was on Reddit that I saw we have a new experimental coding model that's much better at coding and math and so on. Someone external tried it, and they basically prompted it and said, "I'd like you to implement a SQL processing database system with no external dependencies, and please do that in C."

From what the person said, it actually did a quite good job. It generated a SQL parser and a tokenizer and a query planning system and some storage format for the data on disk and actually was able to handle simple queries. From that prompt, which is like a paragraph of text or something, to get even an initial cut at that seems like a big boost in productivity for software developers.

I think you might end up with other kinds of systems that maybe don't try to do that in a single semi-interactive, "respond in 40 seconds" kind of thing but might go off for 10 minutes and might interrupt you after five minutes saying, "I've done a lot of this, but now I need to get some input. Do you care about handling video or just images or something?" That seems like you'll need ways of managing the workflow if you have a lot of these background activities happening.

Dwarkesh Patel *00:40:44*

Can you talk more about that? What interface do you imagine we might need if you could literally have millions of employees you could spin up, hundreds of thousands of employees you could spin up on command, who are able to type incredibly fast, and who- It's almost like you go from 1930s trading of tickets or something to now modern Jane Street or something. You need some interface to keep track of all this that's going on, for the AIs to integrate into this big monorepo and leverage their own strengths, for humans to keep track of what's happening. Basically what is it like to be Jeff or Noam in three years working day-to-day?

Noam Shazeer *00:42:26*

It might be kind of similar to what we have now because we already have sort of parallelization as a major issue. We have lots and lots of really, really brilliant machine learning researchers, and we want them to all work together and build AI.

So actually, the parallelization among people might be similar to parallelization among machines. I think definitely it should be good for things that require a lot of exploration, like, "Come up with the next breakthrough."

If you have a brilliant idea that is just certain to work in the ML domain, then it has a 2% chance of working if you're brilliant. Mostly these things fail, but if you try 100 things or 1,000 things or a million things, then you might hit on something amazing. We have plenty of compute. Like modern top labs these days have probably a million times as much compute as it took to train Transformer.

Dwarkesh Patel *00:43:41*

Yeah, actually, so that's a really interesting idea. Suppose in the world today there are on the order of 10,000 AI researchers in this community coming up with a breakthrough-

Jeff Dean *00:43:52*

Probably more than that. There were 15,000 at NeurIPS last week.

Noam Shazeer *00:43:55*

Wow.

Dwarkesh Patel *00:43:57*

100,000, I don't know.

Jeff Dean *00:43:58*

Yeah, maybe. Sorry.

Dwarkesh Patel *00:44:00*

No, no, it's good to have the correct order of magnitude. The odds that this community every year comes up with a breakthrough on the scale of a Transformer is, let's say, 10%. Now suppose this community is a thousand times bigger, and it is, in some sense, like this sort of parallel search of better architectures, better techniques.

Do we just get like-

Jeff Dean 00:44:22

A breakthrough a day?

Dwarkesh Patel *00:44:23*

-breakthroughs every year or every day?

Noam Shazeer *00:44:25*

Maybe. Sounds potentially good.

Dwarkesh Patel *00:44:30*

But does that feel like what ML research is like? If you are able to try all these experiments…

Noam Shazeer *00:44:37*

It's a good question, because I don't know that folks haven't been doing that as much. We definitely have lots of great ideas coming along. Everyone seems to want to run their experiment at maximum scale, but I think that's a human problem.

Jeff Dean *00:44:55*

It's very helpful to have a 1/1000th scale problem and then vet 100,000 ideas on that, and then scale up the ones that seem promising.

Dwarkesh Patel *00:44:06*

So, one thing the world might not be taking seriously: people are aware that it's exponentially harder to make a model that's 100x bigger. It's 100x more compute, right? So people are worried that it's an exponentially harder problem to go from Gemini 2 to 3, or so forth.

But maybe people aren't aware of this other trend where Gemini 3 is coming up with all these different architectural ideas, trying them out, and you see what works, and you're constantly coming up with algorithmic progress that makes training the next one easier and easier. How far could you take that feedback loop?

Jeff Dean *00:45:43*

I think one thing people should be aware of is that the improvements from generation to generation of these models often are partially driven by hardware and larger scale, but equally and perhaps even more so driven by major algorithmic improvements and major changes in the model architecture, the training data mix, and so on, that really makes the model better per flop that is applied to the model, so I think that's a good realization. Then I think if we have automated exploration of ideas, we'll be able to vet a lot more ideas and bring them into the actual production training for next generations of these models.

That's going to be really helpful because that's sort of what we're currently doing with a lot of brilliant machine learning researchers: looking at lots of ideas, winnowing ones that seem to work well at small scale, seeing if they work well at medium scale, bringing them into larger scale experiments, and then settling on adding a whole bunch of new and interesting things to the final model recipe. If we can do that 100 times faster through those machine learning researchers just gently steering a more automated search process, rather than hand-babysitting lots of experiments themselves, that's going to be really, really good.

Noam Shazeer *00:47:03*

The one thing that doesn't speed up is experiments at the largest scale. You still end up doing these N = 1 experiments. Really, you just try to put a bunch of brilliant people in the room, have them stare at the thing, and figure out why this is working, why this is not working.

Jeff Dean *00:47:21*

For that, more hardware is a good solution. And better hardware.

Noam Shazeer *00:47:25*

Yes, we're counting on you.

Dwarkesh Patel *00:47:28*

So, naively, there's this software, there's this algorithmic side improvement that future AI can make. There's also the stuff you're working on. I'll let you describe it.

But if you get into a situation where just from a software level, you can be making better and better chips in a matter of weeks and months, and better AIs can presumably do that better, how does this feedback loop not just end up in, Gemini 3 taking two years, then Gemini 4 is- or the equivalent level jump is now six months, then level five is three months, then one month? You get to superhuman intelligence much more rapidly than you might naively think, because of this software, both on the hardware side and from the algorithmic side improvements.

Jeff Dean *00:48:26*

I've been pretty excited lately about how we could dramatically speed up the chip design process. As we were talking earlier, the current way in which you design a chip takes you roughly 18 months to go from "we should build a chip" to something that you then hand over to TSMC and then TSMC takes four months to fab it, and then you get it back and you put it in your data centers.

So that's a pretty lengthy cycle, and the fab time in there is a pretty small portion of it today. But if you could make that the dominant portion, so that instead of taking 12 to 18 months to design the chip with 150 people, you could shrink that to a few people with a much more automated search process, exploring the whole design space of chips and getting feedback from all aspects of the chip design process for the kind of choices that the system is trying to explore at the high level, then I think you could get perhaps much more exploration and more rapid design of something that you actually want to give to a fab.

That would be great because you can shrink fab time, you can shrink the deployment time by designing the hardware in the right way, so that you just get the chips back and you just plug them into some system. And that will then enable a lot more specialization, it will enable a shorter timeframe for the hardware design so that you don't have to look out quite as far into what kind of ML algorithms would be interesting. Instead, it's like you're looking at six to nine months from now, what should it be? Rather than two, two and a half years.

That would be pretty cool. I do think that fabrication time, if that's in your inner loop of improvement, you're going to like...

Dwarkesh Patel *00:50:19*

How long is it?

Jeff Dean *00:50:20*

The leading edge nodes, unfortunately, are taking longer and longer because they have more metal layers than previous, older nodes. So that tends to make it take anywhere from three to five months.

Dwarkesh Patel *00:50:32*

Okay, but that's how long training runs take anyways, right? So you could potentially do both at the same time.

Jeff Dean *00:50:38*

Potentially.

Dwarkesh Patel *00:50:39*

Okay, so I guess you can't get sooner than three to five months. But the idea that you could get- but also, yeah, you're rapidly developing new algorithmic ideas.

Noam Shazeer *00:50:47*

That can move fast.

Jeff Dean *00:50:48*

That can move fast, that can run on existing chips and explore lots of cool ideas.

Dwarkesh Patel *00:50:54*

So, isn't that a situation in which you're... I think people sort of expect like, ah, there's going to be a sigmoid. Again, this is not a sure thing. But just like, is this a possibility? The idea that you have sort of an explosion of capabilities very rapidly towards the tail end of human intelligence that gets smarter and smarter at a more and more rapid rate?

Noam Shazeer *00:51:17*

Quite possibly.

Jeff Dean *00:51:19*

Yeah. I like to think of it like this. Right now, we have models that can take a pretty complicated problem and can break it down internally in the model into a bunch of steps, can sort of puzzle together the solutions for those steps, and can often give you a solution to the entire problem that you're asking.

But it isn't super reliable, and it's good at breaking things down into five to ten steps, not 100 to 1,000 steps. So if you could go from, yeah, 80% of the time it can give you a perfect answer to something that's ten steps long to something that 90% of the time can give you a perfect answer to something that's 100 to 1,000 steps of sub-problem long, that would be an amazing improvement in the capability of these models. We're not there yet, but I think that's what we're aspirationally trying to get to.

Noam Shazeer *00:52:14*

We don't need new hardware for that, but we'll take it.

Jeff Dean *00:52:20*

Never look new hardware in the mouth.

Noam Shazeer *00:52:23*

One of the big areas of improvement in the near future is inference time compute, applying more compute at inference time. I guess the way I like to describe it is that even a giant language model, even if you're doing a trillion operations per token, which is more than most people are doing these days, operations cost something like 10 to the negative $18. And so you're getting a million tokens to the dollar.

I mean compare that to a relatively cheap pastime: you go out and buy a paper book and read it, you're paying 10,000 tokens to the dollar. Talking to a language model is like 100 times cheaper than reading a paperback.

So there is a huge amount of headroom there to say, okay, if we can make this thing more expensive but smarter, because we're 100x cheaper than reading a paperback, we're 10,000 times cheaper than talking to a customer support agent, or a million times or more cheaper than hiring a software engineer or talking to your doctor or lawyer. Can we add computation and make it smarter?

I think a lot of the takeoff that we're going to see in the very near future is of this form. We've been exploiting and improving pre-training a lot in the past, and post-training, and those things will continue to improve. But taking advantage of "think harder" at inference time is just going to be an explosion.

Jeff Dean *00:54:21*

Yeah, and an aspect of inference time is I think you want the system to be actively exploring a bunch of different potential solutions. Maybe it does some searches on its own, gets some information back, consumes that information, and figures out, oh, now I would really like to know more about this thing. So now it iteratively explores how to best solve the high-level problem you pose to this system.

And I think having a dial where you can make the model give you better answers with more inference time compute seems like we have a bunch of techniques now that can kind of do that. The more you crank up the dial, the more it costs you in terms of compute, but the better the answers get.

That seems like a nice trade-off to have, because sometimes you want to think really hard because it's a super important problem. Sometimes you probably don't want to spend enormous amounts of compute to compute “what's the answer to one plus one”. Maybe the system –

Dwarkesh Patel *00:55:22*

Shouldn’t decide to come up with new axioms of set theory or whatever!

Jeff Dean *00:55:25*

– should decide to use a calculator tool instead of a very large language model.

Dwarkesh Patel *00:55:31*

Interesting. So are there any impediments to taking inference time, like having some way in which you can just linearly scale up inference time compute? Or is this basically a problem that's sort of solved, and we know how to throw 100x compute, 1000x compute, and get correspondingly better results?

Noam Shazeer *00:55:50*

We're working out the algorithms as we speak. So I believe we'll see better and better solutions to this as these many more than 10,000 researchers are hacking at it, many of them at Google.

Jeff Dean *00:56:06*

I think we do see some examples in our own experimental work of things where if you apply more inference time compute, the answers are better than if you just apply 10x, you can get better answers than x amount of computed inference time. And that seems useful and important.

But I think what we would like is when you apply 10x to get even a bigger improvement in the quality of the answers than we're getting today. And so that's about designing new algorithms, trying new approaches, figuring out how best to spend that 10x instead of x to improve things.

Dwarkesh Patel *00:56:44*

Does it look more like search, or does it look more like just keeping going in the linear direction for a longer time?

Jeff Dean *00:56:49*

I really like Rich Sutton's paper that he wrote about the Bitter Lesson and the Bitter Lesson effectively is this nice one-page paper but the essence of it is you can try lots of approaches, but the two techniques that are incredibly effective are learning and search.

You can apply and scale those algorithmically or computationally, and you often will then get better results than any other kind of approach you can apply it to a pretty broad variety of problems.

Search has got to be part of the solution to spending more inference time. Maybe you explore a few different ways of solving this problem, and that one didn't work, but this one worked better. I'm going to explore that a bit more.

Dwarkesh Patel *00:57:36*

How does this change your plans for future data center planning and so forth? Where can this kind of search be done asynchronously? Does it have to be online, offline? How does that change how big of a campus you need and those kinds of considerations?

Jeff Dean *00:57:55*

One general trend is it's clear that inference time compute, you have a model that's pretty much already trained and you want to do inference on, it is going to be a growing and important class of computation. Maybe you want to specialize hardware more around that.

Actually, the first TPU was specialized for inference and wasn't really designed for training. Then subsequent TPUs were really designed more around training and also for inference.

But it may be that when you have something where you really want to crank up the amount of compute you use at inference time, that even more specialized solutions will make a lot of sense.

Dwarkesh Patel *00:58:38*

Does that mean you can accommodate more asynchronous training?

Jeff Dean *00:58:41*

Training? Or inference?

Dwarkesh Patel *00:58:42*

Or just you can have the different data centers don't need to talk to each other, you can just have them do a bunch of...

Jeff Dean *00:58:52*

I like to think of it as, is the inference that you're trying to do latency-sensitive? Like a user is actively waiting for it, or is it a background thing? Maybe I have some inference tasks that I'm trying to run over a whole batch of data, but it's not for a particular user. It's just I want to run inference on it and extract some information.

There's probably a bunch of things that we don't really have very much of right now, but you're seeing inklings of it in our deep research tool that we just released, like a week ago. You can give it a pretty complicated, high-level task like, "Hey, can you go off and research the history of renewable energy and all the trends in costs for wind and solar and other kinds of techniques, and put it in a table and give me a full eight-page report?" And it will come back with an eight-page report with like 50 entries in the bibliography.

It's pretty remarkable. But you're not actively waiting for that for one second. It takes like a minute or two to go do that.

And I think there's going to be a fair bit of that kind of compute, and that's the kind of thing where you have some UI questions around. Okay, if you're going to have a user with 20 of these kind of asynchronous tasks in the background happening, and maybe each one of them needs to get more information from the user, like, "I found your flights to Berlin, but there's no non-stop ones. Are you okay with a non-stop one?" How does that flow work when you kind of need a bit more information, and then you want to put it back in the background for it to continue doing, you know, finding the hotels in Berlin or whatever? I think it's going to be pretty interesting, and inference will be useful.

Noam Shazeer *01:00:33*

Inference will be useful. There's also a compute efficiency in inference that you don't have in training. In general, transformers can use the sequence length as a batch during training, but they can't really in inference, because when you're generating one token at a time, so there may be different hardware and inference algorithms that we design for the purposes of being efficient at inference.

Jeff Dean *01:01:02*

Yeah, as a good example of an algorithmic improvement is the use of drafter models. So you have a really small language model that you do one token at a time when you're decoding, and it predicts four tokens. Then you give that to the big model and you say, "Okay, here are the four tokens the little model came up with. Check which ones you agree with."

If you agree with the first three, then you just advance. Then you've basically been able to do a four-token width parallel computation instead of a one-token width computation in the big model. Those are the kinds of things that people are looking at to improve inference efficiency, so you don't have this single-token decode bottleneck.

Noam Shazeer *01:01:46*

Right, basically the big model's being used as a verifier.

Jeff Dean *01:01:48*

Right, “can you verify”, yeah.

Noam Shazeer *01:01:50*

[inaudible] generator and verification you can do.

Jeff Dean *01:01:52*

Right. "Hello, how are you?" That sounds great to me. I'm going to advance past that.

Dwarkesh Patel *01:01:56*

So, a big discussion has been about how we're already tapping out nuclear power plants in terms of delivering power into one single campus. Do we have to have just two gigawatts in one place, five gigawatts in one place, or can it be more distributed and still be able to train a model? Does this new regime of inference scaling make different considerations there plausible? How are you thinking about multi-data center training now?

Jeff Dean *01:02:31*

We're already doing it. We're pro multi-data center training. I think in the Gemini 1.5 tech report, we said we used multiple metro areas and trained with some of the compute in each place. And then a pretty long latency but high bandwidth connection between those data centers, and that works fine.

Training is kind of interesting because each step in a training process is usually, for a large model, is usually a few seconds or something, at least. So, the latency of it being 50 milliseconds away doesn't matter that much.

Noam Shazeer *01:03:06*

Just the bandwidth.

Jeff Dean *01:03:08*

Yeah, just bandwidth.

Noam Shazeer *01:03:10*

As long as you can sync all of the parameters of the model across the different data centers and then accumulate all the gradients, in the time it takes to do one step, you're pretty good.

Jeff Dean *01:03:21*

And then we have a bunch of work, even from early Brain days, when we were using CPU machines and they were really slow. We needed to do asynchronous training to help scale, where each copy of the model would do some local computation, send gradient updates to a centralized system, and then apply them asynchronously. Another copy of the model would be doing the same thing.

It makes your model parameters wiggle around a bit, and it makes people uncomfortable with the theoretical guarantees, but it actually seems to work in practice.

Noam Shazeer *01:03:56*

It was so pleasant to go from asynchronous to synchronous because your experiments are now replicable, rather than your results depend on whether there was a web crawler running on the same machine. So, I am so much happier running on TPU pods.

Jeff Dean *01:04:20*

I love asynchrony. It just lets you scale so much more.

Noam Shazeer *01:04:22*

With these two iPhones and an Xbox or whatever.

Jeff Dean *01:04:25*

Yeah, what if we could give you asynchronous but replicable results?

Noam Shazeer *01:04:29*

Ooh.

Jeff Dean *01:04:31*

So, one way to do that is you effectively record the sequence of operations, like which gradient update happened and when and on which batch of data. You don't necessarily record the actual gradient update in a log or something, but you could replay that log of operations so that you get repeatability. Then I think you'd be happy.

Noam Shazeer *01:04:53*

Possibly. At least you could debug what happened, but you wouldn't be able to necessarily compare two training runs. Because, okay, I made one change in the hyperparameter, but also I had a-

Jeff Dean *01:05:08*

Web crawler.

Noam Shazeer *01:05:09*

-web crawler messing up, and there were a lot of people streaming the Super Bowl at the same time.

Jeff Dean *01:05:19*

The thing that led us to go from asynchronous training on CPUs to fully synchronous training is the fact that we have these super fast TPU hardware chips and pods, which have incredible amounts of bandwidth between the chips in a pod. Then, scaling beyond that, we have really good data center networks and even cross-metro area networks that enable us to scale to many, many pods in multiple metro areas for our largest training runs. We can do that fully synchronously.

As Noam said, as long as the gradient accumulation and communication of the parameters across metro areas happens fast enough relative to the step time, you're golden. You don't really care. But I think as you scale up, there may be a push to have a bit more asynchrony in our systems than we have now because we can make it work, our ML researchers have been really happy how far we've been able to push synchronous training because it is an easier mental model to understand. You just have your algorithm sort of fighting you, rather than the asynchrony and the algorithm kind of battling you.

Noam Shazeer *01:06:28*

As you scale up, there are more things fighting you. That's the problem with scaling, that you don't always know what it is that's fighting you. Is it the fact that you've pushed quantization a little too far in some place or another? Or is it your data?

Jeff Dean *01:06:46*

Maybe it's your adversarial machine MUQQ17 that is setting the seventh bit of your exponent and all your gradients or something.

Noam Shazeer *01:06:56*

Right. And all of these things just make the model slightly worse, so you don't even know that the thing is going on.

Jeff Dean *01:07:04*

That's actually a bit of a problem with neural nets, is they're so tolerant of noise. You can have things set up kind of wrong in a lot of ways, and they just figure out ways to work around that or learn.

Noam Shazeer *01:07:15*

You could have bugs in your code. Most of the time that does nothing. Some of the time it makes your model worse. Some of the time it makes your model better. Then you discover something new because you never tried this bug at scale before because you didn't have the budget for it.

Dwarkesh Patel *01:07:33*

What practically does it look like to debug or decode? You've got these things, some of which are making the model better, some of which are making it worse. When you go into work tomorrow, how do you figure out what the most salient inputs are?

Noam Shazeer *01:07:50*

At small scale, you do lots of experiments. There's one part of the research that involves, okay, I want to invent these improvements or breakthroughs in isolation. In which case you want a nice simple code base that you can fork and hack, and some baselines.

My dream is I wake up in the morning, come up with an idea, hack it up in a day, run some experiments, get some initial results in a day. Like okay this looks promising, these things worked, and these things didn't work.

I think that is very achievable because-

Jeff Dean *01:08:34*

At small scale.

Noam Shazeer *01:08:35*

At small scale, as long as you keep a nice experimental code base.

Jeff Dean *01:08:41*

Maybe an experiment takes an hour to run or two hours, not two weeks.

Noam Shazeer *01:08:45*

It's great. So there's that part of the research, and then there's some amount of scaling up. Then you have the part which is integrating, where you want to stack all the improvements on top of each other and see if they work at large scale, and see if they work all in conjunction.

Jeff Dean *01:09:02*

Right, how do they interact? Right, you think maybe they're independent, but actually maybe there's some funny interaction between improving the way in which we handle video data input and the way in which we update the model parameters. Maybe that interacts more for video data than some other thing.

There are all kinds of interactions that can happen that you maybe don't anticipate. So you want to run these experiments where you're then putting a bunch of things together and then periodically making sure that all the things you think are good are good together. If not, understanding why they're not playing nicely.

Dwarkesh Patel *01:09:41*

Two questions. One, how often does it end up being the case that things don't stack up well together? Is it like a rare thing or does it happen all the time?

Noam Shazeer *01:09:52*

It happens 50% of the time.

Jeff Dean *01:09:55*

Yeah, I mean, I think most things you don't even try to stack because the initial experiment didn't work that well, or it showed results that aren't that promising relative to the baseline. Then you sort of take those things and you try to scale them up individually.

Then you're like, "Oh yeah, these ones seem really promising." So I'm going to now include them in something that I'm going to now bundle together and try to advance and combine with other things that seem promising. Then you run the experiments and then you're like, "Oh, well, they didn't really work that well. Let's try to debug why."

Noam Shazeer *01:10:28*

And then there are trade offs, because you want to keep your integrated system as clean as you can, because complexity –

Jeff Dean *01:10:38*

Codebase-wise.

Noam Shazeer *01:10:39*

– yeah codebase and algorithmically. Complexity hurts, complexity makes things slower, introduces more risk.

And then at the same time you want it to be as good as possible. And of course, every individual researcher wants his inventions to go into it. So there are definitely challenges there, but we've been working together quite well.

Dwarkesh Patel *01:11:05*

Okay, so then going back to the whole dynamic “you find better and better algorithmic improvements and the models get better and better over time”, even if you take the hardware part out of it. Should the world be thinking more about, and should *you guys* be thinking more about this?

There's one world where AI is a thing that takes two decades to slowly get better over time and you can sort of refine things over. If you've kind of messed something up, you fix it, and it's not that big a deal, right? It's like not that much better than the previous version you released.

There's another world where you have this big feedback loop, which means that the two years between Gemini 4 and Gemini 5 are the most important years in human history. Because you go from a pretty good ML researcher to superhuman intelligence because of this feedback loop. To the extent that you think that the second world is plausible, how does that change how you sort of approach these greater and greater levels of intelligence?

Noam Shazeer *01:12:14*

I've stopped cleaning my garage because I'm waiting for the robots. So probably I'm more in the second camp of what we're going to see, a lot of acceleration.

Jeff Dean *01:12:24*

Yeah, I mean, I think it's super important to understand what's going on and what the trends are. And I think right now the trends are the models are getting substantially better generation over generation. I don't see that slowing down in the next few generations probably.

So that means the models say two to three generations from now are going to be capable of... Let's go back to the example of breaking down a simple task into 10 sub pieces and doing it 80% of the time, to something that can break down a task, a very high level task, into 100 or 1,000 pieces and get that right 90% of the time. That's a major, major step up in what the models are capable of.

So I think it's important for people to understand what is happening in the progress in the field. And then those models are going to be applied in a bunch of different domains. I think it's really good to make sure that we, as a society, get the maximal benefits from what these models can do to improve things. I'm super excited about areas like education and healthcare, making information accessible to all people.

But we also realize that they could be used for misinformation, they could be used for automated hacking of computer systems, and we want to put as many safeguards and mitigations and understand the capabilities of the models in place as we can. I think Google as a whole has a really good view to how we should approach this. Our Responsible AI principles actually are a pretty nice framework for how to think about trade offs of making better and better AI systems available in different contexts and settings, while also sort of making sure that we're doing the right thing in terms of making sure they're safe and not saying toxic things and things like that.

Dwarkesh Patel *01:14:21*

I guess the thing that stands out to me, if you were zooming out and looking at this period of human history, if we're in the world where, look, if you do post-training on Gemini 3 badly, it can do some misinformation – but then you fix the post training. It's a bad mistake, but it's a fixable mistake, right?

Noam Shazeer *01:14:40*

Right.

Dwarkesh Patel *01:14:40*

Whereas if you have this feedback loop dynamic, which is a possibility, then the mistake of the thing that catapults this intelligence explosion is misaligned, is not trying to write the code you think it's trying to write, and [instead] optimizing for some other objective.

And on the other end of this very rapid process that lasts a couple of years, maybe less, you have things that are approaching Jeff Dean or beyond level, or Noam Shazeer or beyond level. And then you have millions of copies of Jeff Dean level programmers, and- anyways, that seems like a harder to recover mistake.

Noam Shazeer *01:15:29*

As these systems do get more powerful, you have to be more and more careful.

Jeff Dean *01:15:37*

One thing I would say is, there are extreme views on either end. There's, "Oh my goodness, these systems are going to be so much better than humans at all things, and we're going to be kind of overwhelmed." And then there's, "These systems are going to be amazing, and we don't have to worry about them at all."

I think I'm somewhere in the middle. I've been a co-author on a paper called "Shaping AI," which is, you know, those two extreme views often kind of view our role as kind of laissez-faire, like we're just going to have the AI develop in the path that it takes.

And I think there's actually a really good argument to be made that what we're going to do is try to shape and steer the way in which AI is deployed in the world so that it is, you know, maximally beneficial in the areas that we want to capture and benefit from, in education, some of the areas I mentioned, healthcare.

And steer it as much as we can away- maybe with policy-related things, maybe with technical measures and safeguards- away from, you know, the computer will take over and have unlimited control of what it can do. So I think that's an engineering problem: how do you engineer safe systems?

I think it's kind of the modern equivalent of what we've done in older-style software development. Like if you look at, you know, airplane software development, that has a pretty good record of how do you rigorously develop safe and secure systems for doing a pretty risky task?

Dwarkesh Patel *01:17:18*

The difficulty there is that there's not some feedback loop where the 737, you put it in a box with a bunch of compute for a couple of years, and it comes out with the version 1000.

Noam Shazeer *01:17:27*

I think the good news is that analyzing text seems to be easier than generating text. So I believe that the ability of language models to actually analyze language model output and figure out what is problematic or dangerous will actually be the solution to a lot of these control issues.

We are definitely working on this stuff. We've got a bunch of brilliant folks at Google working on this now. And I think it's just going to be more and more important, both from a “do something good for people” standpoint, but also from a business standpoint, that you are, a lot of the time, limited in what you can deploy based on keeping things safe.

And so it becomes very, very important to be really, really good at that.

Dwarkesh Patel *01:18:48*

Yeah, obviously, I know you guys take the potential benefits and costs here seriously, and it's truly remarkable. I know you guys get credit for it, but not enough. I think there's just, there are so many different applications that you have put out for using these models to make the different areas you talked about better.

Um, but I do think that… again, if you have a situation where plausibly there's some feedback loop process, on the other end, you have a model that is as good as Noam Shazeer, as good as Jeff Dean.

If there's an evil version of you running around, and suppose there's a million of them, I think that's really, really bad. That could be much, much worse than any other risk, maybe short of nuclear war or something. Just think about it, like a million evil Jeff Deans or something.

Jeff Dean *01:19:47*

Where do we get the training data?

Dwarkesh Patel *01:20:20*

But, to the extent that you think that's a plausible output of some quick feedback loop process, what is your plan of okay, we've got Gemini 3 or Gemini 4, and we think it's helping us do a better job of training future versions, it's writing a bunch of the training code for us. From this point forward, we just kind of look over it, verify it.

Even the verifiers you talked about of looking at the output of these models will eventually be trained by, or a lot of the code will be written by the AIs you make. What do you want to know for sure before we have the Gemini 4 help us with the AI research? We really want to make sure, we want to run this test on it before we let it write our AI code for us.

Jeff Dean *01:21:34*

I mean, I think having the system explore algorithmic research ideas seems like something where there's still a human in charge of that. Like, it's exploring the space, and then it's going to, like, get a bunch of results, and we're going to make a decision, like, are we going to incorporate this particular, you know, learning algorithm or change to the system into kind of the core code base?

And so I think you can put in safeguards like that that enable us to get the benefits of the system that can sort of improve or kind of self-improve with human oversight, uh, without necessarily letting the system go full-on self-improving without any any notion of a person looking at what it's doing, right? That's the kind of engineering safeguards I'm talking about, where you want to be kind of looking at the characteristics of the systems you're deploying, not deploy ones that are harmful by some measures and some ways, and you have an understanding of what its capabilities are and what it's likely to do in certain scenarios. So, you know, I think it's not an easy problem by any means, but I do think it is possible to make these these systems safe.

Noam Shazeer *01:36:56*

Yeah. I mean, I think we are also going to use these systems a lot to check themselves, check other systems. Even as a human, it is easier to recognize something than to generate it.

Jeff Dean *01:37:14*

One thing I would say is if you expose the model's capabilities through an API or through a user interface that people interact with, I think then you have a level of control to understand how is it being used and put some boundaries on what it can do. And that I think is one of the tools in the arsenal of how do you make sure that what it's going to do is sort of acceptable by some set of standards you've set out in your mind?

Noam Shazeer *01:37:44*

Yeah. I mean, I think the goal is to to empower people, but for the most part we should be mostly letting people do things with these systems that make sense and closing off as few parts of the space as we can. But yeah, if you let somebody take your thing and create a million evil software engineers, then that doesn't empower people because they're going to hurt others with a million evil software engineers.

So I'm against that.

Jeff Dean *01:38:14*

Me too. I'll go on.

Dwarkesh Patel *01:38:16*

All right, let's talk about a few more fun topics. Make it a little lighter. Over the last 25 years, what was the most fun time? What period of time do you have the most nostalgia over?

Jeff Dean *01:38:30*

At work, you mean?

Noam Shazeer *01:38:31*

Yeah. At work. Yeah.

Jeff Dean *01:38:32*

I think the early sort of four or five years at Google when I was one of a handful of people working on search and crawling and indexing systems, our traffic was growing tremendously fast. We were trying to expand our index size and make it so we updated it every minute instead of every month, or two months if something went wrong.

Seeing the growth in usage of our systems was really just personally satisfying. Building something that is used by two billion people a day is pretty incredible.

But I would also say equally exciting is working with people on the Gemini team today. I think the progress we've been making in what these models can do over the last year and a half is really fun. People are really dedicated, really excited about what we're doing.

I think the models are getting better and better at pretty complex tasks. Like if you showed someone using a computer 20 years ago what these models are capable of, they wouldn't believe it. And even five years ago, they might not believe it. And that's pretty satisfying.

I think we'll see a similar growth in usage of these models and impact in the world.

Noam Shazeer *01:39:48*

Yeah, I'm with you. Early days were super fun. Part of that is just knowing everybody and the social aspect, and the fact that you're just building something that millions and millions of people are using.

Same thing today. We got that whole nice micro kitchen area where you get lots of people hanging out. I love being in person, working with a bunch of great people, and building something that's helping millions to billions of people. What could be better?

Dwarkesh Patel *01:40:21*

What was this micro kitchen?

Jeff Dean *01:40:23*

Oh, we have a micro kitchen area in the building we both sit in. It's the new, so-named Gradient Canopy. It used to be named Charleston East, and we decided we needed a more exciting name because it's a lot of machine learning researchers and AI research happening in there.

There's a micro kitchen area that we've set up with, normally it's just like an espresso machine and a bunch of snacks, but this particular one has a bunch of space in it. So we've set up maybe 50 desks in there, and so people are just hanging out in there. It's a little noisy because people are always grinding beans and brewing espresso, but you also get a lot of face-to-face ideas of connections, like, "Oh, I've tried that. Did you think about trying this in your idea?" Or, "Oh, we're going to launch this thing next week. How's the load test looking?" There's just lots of feedback that happens.

And then we have our Gemini chat room for people who are not in that micro kitchen. We have a team all over the world, and there's probably 120 chat rooms I'm in related to Gemini things. In this particular very focused topic, we have seven people working on this, and there are exciting results being shared by the London colleagues.

When you wake up, you see what's happening in there, or it's a big group of people focused on data, and there are all kinds of issues happening in there. It's just fun.

Dwarkesh Patel *01:41:50*

What I find remarkable about some of the calls you guys have made is you're anticipating a level of demand for compute, which at the time wasn't obvious or evident. TPUs being a famous example of this, or the first TPU being an example of this.

That thinking you had in, I guess, 2013 or earlier, if you think about it that way today and you do an estimate of, look, we're going to have these models that are going to be a backbone of our services, and we're going to be doing constant inference for them. We're going to be training future versions. And you think about the amount of compute we'll need by 2030 to accommodate all these use cases, where does the Fermi estimate get you?

Jeff Dean *01:42:30*

Yeah, I mean, I think you're going to want a lot of inference. Compute is the rough, highest-level view of these capable models because if one of the techniques for improving their quality is scaling up the amount of inference compute you use, then all of a sudden what's currently like one request to generate some tokens now becomes 50 or 100 or 1000 times as computationally intensive, even though it's producing the same amount of output.

And you're also going to then see tremendous scaling up of the uses of these services as not everyone in the world has discovered these chat-based conversational interfaces where you can get them to do all kinds of amazing things. Probably 10% of the computer users in the world have discovered that today, or 20%. As that pushes towards 100% and people make heavier use of it, that's going to be another order of magnitude or two of scaling.

And so you're now going to have two orders of magnitude from that, two orders of magnitude from that. The models are probably going to be bigger, you'll get another order of magnitude or two from that. And there's a lot of inference compute you want. So you want extremely efficient hardware for inference for models you care about.

Dwarkesh Patel *01:43:52*

In flops, global total global inference in 2030?

Noam Shazeer *01:43:58*

I think just more is always going to be better. If you just kind of think about, okay, what fraction of world GDP will people decide to spend on AI at that point? And then, like, okay, what do the AI systems look like?

Well, maybe it's some sort of personal assistant-like thing that is in your glasses and can see everything around you and has access to all your digital information and the world's digital information. And maybe it's like you're Joe Biden, and you have the earpiece in the cabinet that can advise you about anything in real-time and solve problems for you and give you helpful pointers. Or you could talk to it, and it wants to analyze anything that it sees around you for any potential useful impact that it has on you.

So I mean, I can imagine, okay, and then say it's like your personal assistant or your personal cabinet or something, and that every time you spend 2x as much money on compute, the thing gets like 5, 10 IQ points smarter or something like that. And, okay, would you rather spend $10 a day and have an assistant or $20 a day and have a smarter assistant? And not only is it an assistant in life but an assistant in getting your job done better because now it makes you from a 10x engineer to a 100x or 10 millionx engineer?

Okay, so let's see: from first principles, right? So people are going to want to spend some fraction of world GDP on this thing. The world GDP is almost certainly going to go way, way up, two orders of magnitude higher than it is today, due to the fact that we have all of these artificial engineers working on improving things.

Probably we'll have solved unlimited energy and carbon issues by that point. So we should be able to have lots of energy. We should be able to have millions to billions of robots building us data centers. Let's see, the sun is what, 10 to the 26 watts or something like that?

I'm guessing that the amount of compute being used for AI to help each person will be astronomical.

Jeff Dean *01:46:48*

I would add on to that. I'm not sure I agree completely, but it's a pretty interesting thought experiment to go in that direction. And even if you get partway there, it's definitely going to be a lot of compute.

And this is why it's super important to have as cheap a hardware platform for using these models and applying them to problems that Noam described, so that you can then make it accessible to everyone in some form and have as low a cost for access to these capabilities as you possibly can.

And I think that's achievable by focusing on hardware and model co-design kinds of things, we should be able to make these things much, much more efficient than they are today.

Dwarkesh Patel *01:47:36*

Is Google's data center build-out plan over the next few years aggressive enough given this increase in demand you're expecting?

Jeff Dean *01:47:46*

I'm not going to comment on our future capital spending because our CEO and CFO would prefer I probably not. But I will say, you can look at our past capital expenditures over the last few years and see that we're definitely investing in this area because we think it's important.

We are continuing to build new and interesting, innovative hardware that we think really helps us have an edge in deploying these systems to more and more people, both training them and also, how do we make them usable by people for inference?

Dwarkesh Patel *01:48:21*

One thing I've heard you talk a lot about is continual learning, the idea that you could just have a model which improves over time rather than having to start from scratch. Is there any fundamental impediment to that? Because theoretically, you should just be able to keep fine-tuning a model. What does that future look like to you?

Jeff Dean *01:48:40*

Yeah, I've been thinking about this more and more. I've been a big fan of models that are sparse because I think you want different parts of the model to be good at different things. We have our Gemini 1.5 Pro model, and other models are mixture-of-experts style models where you now have parts of the model that are activated for some token and parts that are not activated at all because you've decided this is a math-oriented thing, and this part's good at math, and this part's good at understanding cat images. So, that gives you this ability to have a much more capable model that's still quite efficient at inference time because it has very large capacity, but you activate a small part of it.

But I think the current problem, well, one limitation of what we're doing today is it's still a very regular structure where each of the experts is the same size. The paths merge back together very fast. They don't go off and have lots of different branches for mathy things that don't merge back together with the kind of cat-image thing.

I think we should probably have a more organic structure in these things. I also would like it if the pieces of those model of the model could be developed a little bit independently. Like right now, I think we have this issue where we're going to train a model. So, we do a bunch of preparation work on deciding the most awesome algorithms we can come up with and the most awesome data mix we can come up with.

But there's always trade-offs there, like we'd love to include more multilingual data, but that might come at the expense of including less coding data, and so, the model's less good at coding but better at multilingual, or vice versa. I think it would be really great if we could have a small set of people who care about a particular subset of languages go off and create really good training data, train a modular piece of a model that we can then hook up to a larger model that improves its capability in, say, Southeast Asian languages or in reasoning about Haskell code or something.

Then, you also have a nice software engineering benefit where you've decomposed the problem a bit compared to what we do today, which is we have this kind of a whole bunch of people working. But then, we have this kind of monolithic process of starting to do pre-training on this model.

If we could do that, you could have 100 teams around Google. You could have people all around the world working to improve languages they care about or particular problems they care about and all collectively work on improving the model. And that's kind of a form of continual learning.

Noam Shazeer *01:51:27*

That would be so nice. You could just glue models together or rip out pieces of models and shove them into other...

Jeff Dean *01:51:33*

Upgrade this piece without throwing out the thing...

Noam Shazeer *01:51:36*

...or you just attach a fire hose, and you suck all the information out of this model, shove it into another model. There is, I mean, the countervailing interest there is sort of science, in terms of, okay, we're still in the period of rapid progress, so, if you want to do sort of controlled experiments, and okay, I want to compare this thing to that thing because that then is helping us figure out what to build. In that interest, it's often best to just start from scratch so you can compare one complete training run to another complete training run at the practical level because it helps us figure out what to build in the future. It's less exciting but does lead to rapid progress.

Jeff Dean *01:52:32*

Yeah, I think there may be ways to get a lot of the benefits of that with a version system of modularity. I have a frozen version of my model, and then I include a different variant of some particular module, and I want to compare its performance or train it a bit more. Then, I compare it to the baseline of this thing with now version N prime of this particular module that does Haskell interpretation.

Noam Shazeer *01:52:58*

Actually, that could lead to faster research progress, right? You've got some system, and you do something to improve it. And if that thing you're doing to improve it is relatively cheap compared to training the system from scratch, then it could actually make research much, much cheaper and faster.

Jeff Dean *01:53:16*

Yeah, and also more parallelizable, I think, across people.

Noam Shazeer *01:53:24*

Okay, let's figure it out and do that next.

Dwarkesh Patel *01:53:29*

So, this idea that is sort of casually laid out there would actually be a big regime shift compared to how things are done today. If you think the way things are headed, this is a sort of very interesting prediction about... You just have this blob where things are getting pipelined back and forth – and if you want to make something better, you can do like a sort of surgical incision almost.

Jeff Dean *01:53:55*

Right, or grow the model, add another little bit of it here. Yeah, I've been sort of sketching out this vision for a while in Pathways...

Noam Shazeer *01:54:04*

Yeah, you've been building the...

Jeff Dean *01:54:05*

...and we've been building the infrastructure for it. So, a lot of what Pathways, the system, can support is this kind of twisty, weird model with asynchronous updates to different pieces. And we're using Pathways to train our Gemini models, but we're not making use of some of its capabilities yet. But maybe we should.

Noam Shazeer *01:54:24*

Ooh maybe.

Dwarkesh Patel *01:54:27*

This is so interesting, and I don't want to lose this thread, but give me one moment.

Noam Shazeer *01:54:33*

There have been times, like the way the TPU pods were set up. I don't know who did that, but they did a pretty brilliant job. The low-level software stack and the hardware stack, okay, you've got your nice regular high-performance hardware, you've got these great torus-shaped interconnects, and then you've got the right low-level collectives, the all-reduces, et cetera, which I guess came from supercomputing, but it turned out to be kind of just the right thing to build distributed deep learning on top of.

Dwarkesh Patel *01:39:15*

Okay, so a couple of questions. One, suppose Noam makes another breakthrough, and now we've got a better architecture. Would you just take each compartment and distill it into this better architecture? And that's how it keeps improving over time?

Jeff Dean *01:39:33*

I do think distillation is a really useful tool because it enables you to transform a model in its current model architecture form into a different form. Often, you use it to take a really capable but large and unwieldy model and distill it into a smaller one that maybe you want to serve with really good, fast latency inference characteristics.

But I think you can also view this as something that's happening at the module level. Maybe there'd be a continual process where you have each module, and it has a few different representations of itself. It has a really big one. It's got a much smaller one that is continually distilling into the small version.

And then the small version, once that's finished, you sort of delete the big one and you add a bunch more parameter capacity. Now, start to learn all the things that the distilled small one doesn't know by training it on more data, and then you kind of repeat that process. If you have that kind of running a thousand different places in your modular model in the background, that seems like it would work reasonably well.

Dwarkesh Patel *01:40:42*

This could be a way of doing inference scaling, like the router decides how much do you want the big one.

Jeff Dean *01:40:47*

Yeah, you can have multiple versions. Oh, this is an easy math problem, so I'm going to route it to the really tiny math distilled thing. Oh, this one's really hard, so...

Dwarkesh Patel *01:40:56*

One, at least from public research, it seems like it's often hard to decode what each expert is doing in mixture of expert type models. If you have something like this, how would you enforce the kind of modularity that would be visible and understandable to us?

Noam Shazeer *01:41:13*

Actually, in the past, I found experts to be relatively easy to understand. I mean, the first Mixture of Experts paper, you could just look at the experts.

Dwarkesh Patel *01:41:24*

“I don’t know, I'm only the inventor of Mixture of Experts.”

Noam Shazeer *01:57:25*

Yeah – oh, the what?

Jeff Dean *01:41:28*

Yeah, yeah.

Noam Shazeer *01:41:30*

Yeah, you could just see, okay, this expert, like we did, you know, a thousand, two thousand experts. Okay, and this expert, was getting words referring to cylindrical objects.

Jeff Dean *01:41:42*

This one's super good at dates.

Noam Shazeer *01:41:44*

Yeah.

Jeff Dean *01:41:45*

Talking about times.

Noam Shazeer *01:41:46*

Yeah, pretty easy to do.

Not that you would need that human understanding to figure out how to work the thing at runtime because you just have some sort of learned router that's looking at the example.

Jeff Dean *01:42:04*

One thing I would say is there is a bunch of work on interpretability of models and what are they doing inside. Sort of expert-level interpretability is a sub-problem of that broader area. I really like some of the work that my former intern, Chris Olah, and others did at Anthropic, where they trained a very sparse autoencoder and were able to deduce what characteristics some particular neuron in a large language model has, so they found a Golden Gate Bridge neuron that's activated when you're talking about the Golden Gate Bridge. And I think you could do that at the expert level, you could do that at a variety of different levels and get pretty interpretable results, and it's a little unclear if you necessarily need that. If the model is just really good at stuff, we don't necessarily care what every neuron in the Gemini model is doing, as long as the collective output and characteristics of the overall system are good. That's one of the beauties of deep learning, is you don't need to understand or hand-engineer every last feature.

Dwarkesh Patel *01:43:13*

Man, there are so many interesting implications of this that I could just keep asking you about this- I would regret not asking you more about this, so I'll keep going. One implication is, currently, if you have a model that has some tens or hundreds of billions of parameters, you can serve it on a handful of GPUs.

In this system, where any one query might only make its way through a small fraction of the total parameters, but you need the whole thing loaded into memory, the specific kind of infrastructure that Google has invested in with these TPUs that exist in pods of hundreds or thousands would be immensely valuable, right?

Noam Shazeer *01:44:02*

For any sort of even existing mixtures of experts, you want the whole thing in memory. I guess there's kind of this misconception running around with Mixture of Experts that, okay, the benefit is that you don't even have to go through those weights in the model.

If some expert is unused, it doesn't mean that you don't have to retrieve that memory because, really, in order to be efficient, you're serving at very large batch sizes.

Jeff Dean *01:44:36*

Of independent requests.

Noam Shazeer *01:44:38*

Right, of independent requests. So it's not really the case that, okay, at this step, you're either looking at this expert or you're not looking at this expert.

Because if that were the case, then when you did look at the expert, you would be running it at batch size one, which is massively inefficient. Like you've got modern hardware, the operational intensities are whatever, hundreds. So that's not what's happening. It's that you are looking at all the experts, but you only have to send a small fraction of the batch through each one.

Jeff Dean *01:45:17*

Right, but you still have a smaller batch at each expert that then goes through. And in order to get kind of reasonable balance, one of the things that the current models typically do is they have all the experts be roughly the same compute cost, and then you run roughly the same size batches through them in order to propagate the very large batch you're doing at inference time and have good efficiency.

But I think you often in the future might want experts that vary in computational cost by factors of 100 or 1000. Or maybe paths that go for many layers on one case, and a single layer or even a skip connection in the other case. And there, I think you're going to want very large batches still, but you're going to want to push things through the model a little bit asynchronously at inference time, which is a little easier than training time.

That's part of one of the things that Pathways was designed to support. You have these components, and the components can be variable cost and you kind of can say, for this particular example, I want to go through this subset of the model, and for this example, I want to go through this subset of the model and have the system kind of orchestrate that.

Dwarkesh Patel *01:46:39*

It also would mean that it would take companies of a certain size and sophistication to be able to... Right now, anybody can train a sufficiently small enough model. But if it ends up being the case that this is the best way to train future models, then you would need a company that can basically have a data center serving a single quote, unquote “blob” or model. So it would be an interesting change in paradigms in that way as well.

Noam Shazeer *01:47:10*

You definitely want to have at least enough HBM to put your whole model. So depending on the size of your model, most likely that's how much HBM you'd want to have at a minimum.

Jeff Dean *01:47:28*

It also means you don't necessarily need to grow your entire model footprint to be the size of a data center. You might want it to be a bit below that.

And then have potentially many replicated copies of one particular expert that is being used a lot, so that you get better load balancing. This one's being used a lot because we get a lot of math questions, and this one is an expert on Tahitian dance, and it is called on really rarely.

That one, maybe you even page out to DRAM rather than putting it in HBM. But you want the system to figure all this stuff out based on load characteristics.

Dwarkesh Patel *01:48:09*

Right. Now, language models, obviously, you put in language, you get language out. Obviously, it's multimodal.

But the Pathways blog post talks about so many different use cases that are not obviously of this kind of auto-regressive nature going through the same model. Could you imagine, basically, Google as a company, the product is like Google Search goes through this, Google Images goes through this, Gmail goes through it?

Just like the entire server is just this huge mixture of experts, specialized?

Jeff Dean *01:48:45*

You're starting to see some of this by having a lot of uses of Gemini models across Google that are not necessarily fine-tuned. They're just given instructions for this particular use case in this feature in this product setting.

So, I definitely see a lot more sharing of what the underlying models are capable of across more and more services. I do think that's a pretty interesting direction to go, for sure.

Dwarkesh Patel *01:49:14*

Yeah, I feel like people listening might not register how interesting a prediction this is about where AI is going. It's like sort of getting Noam on a podcast in 2018 and being like, "Yeah, so I think language models will be a thing."

It's like, if this is where things go, this is actually incredibly interesting.

Jeff Dean *01:49:36*

Yeah, and I think you might see that might be a big base model. And then you might want customized versions of that model with different modules that are added onto it for different settings that maybe have access restrictions.

Maybe we have an internal one for Google use, for Google employees, that we've trained some modules on internal data, and we don't allow anyone else to use those modules, but we can make use of it. Maybe other companies, you add on other modules that are useful for that company setting and serve it in our cloud APIs.

Dwarkesh Patel *01:50:09*

What is the bottleneck to making this sort of system viable? Is it systems engineering? Is it ML?

Jeff Dean *01:50:17*

It's a pretty different way of operating than our current Gemini development. So, I think we will explore these kinds of areas and make some progress on them.

But we need to really see evidence that it's the right way, that it has a lot of benefits. Some of those benefits may be improved quality, some may be less concretely measurable, like this ability to have lots of parallel development of different modules. But that's still a pretty exciting improvement because I think that would enable us to make faster progress on improving the model's capabilities for lots of different distinct areas.

Noam Shazeer *01:51:00*

Even the data control modularity stuff seems really cool because then you could have the piece of the model that's just trained for me. It knows all my private data.

Jeff Dean *01:51:09*

Like a personal module for you would be useful. Another thing might be you can use certain data in some settings but not in other settings.

Maybe we have some YouTube data that's only usable in a YouTube product surface but not in other settings. So, we could have a module that is trained on that data for that particular purpose.

Dwarkesh Patel *01:51:29*

Yeah.

Noam Shazeer *01:51:32*

We're going to need a million automated researchers to invent all of this stuff.

Jeff Dean *01:51:39*

It's going to be great.

Dwarkesh Patel *01:51:41*

Yeah, well the thing itself, you build the blob, and it tells you how to make the blob better.

Jeff Dean *01:51:47*

Blob 2.0. Or maybe they're not even versions, it's just like an incrementally growing blob.

Dwarkesh Patel *01:51:56*

Yeah, that's super fascinating. Okay, Jeff, motivate for me, big picture: why is this a good idea? Why is this the next direction?

Jeff Dean *01:52:06*

Yeah, this notion of an organic, not quite so carefully mathematically constructed machine learning model is one that's been with me for a little while. I feel like in the development of neural nets, the artificial neurons, inspiration from biological neurons is a good one and has served us well in the deep learning field.

We've been able to make a lot of progress with that. But I feel like we're not necessarily looking at other things that real brains do as much as we perhaps could, and that's not to say we should exactly mimic that because silicon and wetware have very different characteristics and strengths. But I do think one thing we could draw more inspiration from is this notion of having different specialized portions, sort of areas of a model of a brain that are good at different things.

We have a little bit of that in Mixture of Experts models, but it's still very structured. I feel like this kind of more organic growth of expertise, and when you want more expertise of that, you add some more capacity to the model there and let it learn a bit more on that kind of thing.

Also this notion of adapting the connectivity of the model to the connectivity of the hardware is a good one. I think you want incredibly dense connections between artificial neurons in the same chip and the same HBM because that doesn't cost you that much. But then you want a smaller number of connections to nearby neurons. So, like a chip away, you should have some amount of connections and then, like many, many chips away, you should have a smaller number of connections where you send over a very limited kind of bottlenecky thing: the most important things that this part of the model is learning for other parts of the model to make use of. And even across multiple TPU pods, you'd like to send even less information but the most salient kind of representations. And then across metro areas, you'd like to send even less.

Dwarkesh Patel *01:54:23*

Yeah, and then that emerges organically.

Jeff Dean *01:54:26*

Yeah, I'd like that to emerge organically. You could hand-specify these characteristics, but I think you don't know exactly what the right proportions of these kinds of connections are so you should just let the hardware dictate things a little bit. Like if you're communicating over here and this data always shows up really early, you should add some more connections, then it'll take longer and show up at just the right time.

Dwarkesh Patel *01:54:48*

Oh here's another interesting implication: Right now, we think about the growth in AI use as a sort of horizontal- so, suppose you're like, how many AI engineers will Google have working for it? You think about how many instances of Gemini 3 will be working at one time.

If you have this, whatever you want to call it, this blob, and it can sort of organically decide how much of itself to activate, then it's more of, if you want 10 engineers worth of output, it just activates a different pattern or a larger pattern. If you want 100 engineers of output, it's not like calling more agents or more instances, it's just calling different sub-patterns.

Jeff Dean *01:55:34*

I think there's a notion of how much compute do you want to spend on this particular inference, and that should vary by factors of 10,000 for really easy things and really hard things, maybe even a million. It might be iterative, you might make a pass through the model, get some stuff, and then decide you now need to call on some other parts of the model.

The other thing I would say is this sounds super complicated to deploy because it's this weird, constantly evolving thing with maybe not super optimized ways of communicating between pieces, but you can always distill from that. If you say, "This is the kind of task I really care about, let me distill from this giant kind of organic thing into something that I know can be served really efficiently," you could do that distillation process whenever you want, once a day, once an hour. That seems like it'd be kind of good.

Noam Shazeer *01:56:32*

Yeah, we need better distillation.

Jeff Dean *01:56:34*

Yeah.

Noam Shazeer *01:56:34*

Anyone out there who invents amazing distillation techniques that instantly distill from a giant blob onto your phone, that would be wonderful.

Dwarkesh Patel *01:56:43*

How would you characterize what's missing from current distillation techniques?

Noam Shazeer *01:56:46*

Well, I just want it to work faster.

Jeff Dean *01:56:49*

A related thing is I feel like we need interesting learning techniques during pre-training. I'm not sure we're extracting the maximal value from every token we look at with the current training objective. Maybe we should think a lot harder about some tokens.

When you get to "the answer is," maybe the model should, at training time, do a lot more work than when it gets to "the".

Noam Shazeer *01:57:16*

Right. There's got to be some way to get more from the same data, make it learn it forwards and backwards.

Jeff Dean *01:57:24*

And every which way. Hide some stuff this way, hide some stuff that way, make it infer from partial information. I think people have been doing this in vision models for a while. You distort the model or you hide parts of it and try to make it guess the bird from half, like that it's a bird from this upper corner of the image or the lower left corner of the image.

That makes the task harder, and I feel like there's an analog for more textual or coding-related data where you want to force the model to work harder. You'll get more interesting observations from it.

Noam Shazeer *01:58:03*

Yeah, the image people didn't have enough labeled data so they had to invent all this stuff.

Jeff Dean *01:58:08*

And they invented -- I mean, dropout was invented on images, but we're not really using it for text mostly. That's one way you could get a lot more learning in a more large-scale model without overfitting is just make like 100 epochs over the world's text data and use dropout.

But that's pretty computationally expensive, but it does mean we won't run it. Even though people are saying, "Oh no, we're almost out of textual data," I don't really believe that because I think we can get a lot more capable models out of the text data that does exist.

Noam Shazeer *01:58:44*

I mean, a person has seen a billion tokens.

Jeff Dean *01:58:47*

Yeah, and they're pretty good at a lot of stuff.

Dwarkesh Patel *01:58:54*

So obviously human data efficiency sets a lower bound on how, or I guess, upper bound, one of them, maybe not.

Jeff Dean *01:59:04*

It's an interesting data point.

Dwarkesh Patel *01:59:05*

Yes. So there's a sort of modus ponens, modus tollens thing here. One way to look at it is, look, LLMs have so much further to go, therefore we project orders of magnitude improvement in sample efficiency just if they could match humans. Another is, maybe they're doing something clearly different given the orders of magnitude difference. What's your intuition of what it would take to make these models as sample efficient as humans are?

Jeff Dean *01:59:33*

Yeah, I think we should consider changing the training objective a little bit. Just predicting the next token from the previous ones you've seen seems like not how people learn. It's a little bit related to how people learn, I think, but not entirely. A person might read a whole chapter of a book and then try to answer questions at the back, and that's a different kind of thing.

I also think we're not learning from visual data very much. We're training a little bit on video data, but we're definitely not anywhere close to thinking about training on all the visual inputs you could get. So you have visual data that we haven't really begun to train on.

Then I think we could extract a lot more information from every bit of data we do see. I think one of the ways people are so sample efficient is they explore the world and take actions in the world and observe what happens. You see it with very small infants picking things up and dropping them; they learn about gravity from that. And that's a much harder thing to learn when you're not initiating the action.

I think having a model that can take actions as part of its learning process would be just a lot better than just sort of passively observing a giant dataset.

Dwarkesh Patel *02:00:50*

Is Gato the future, then?

Jeff Dean *02:00:53*

Something where the model can observe and take actions and observe the corresponding results seems pretty useful.

Noam Shazeer *02:01:04*

I mean, people can learn a lot from thought experiments that don't even involve extra input. Einstein learned a lot of stuff from thought experiments, or like Newton went into quarantine and got an apple dropped on his head or something and invented gravity. And like mathematicians -- math didn't have any extra input.

Chess, okay, you have the thing play chess against itself and it gets good at chess. That was DeepMind, but also all it needs is the rules of chess. So there's actually probably a lot of learning that you can do even without external data, and then you can make it in exactly the fields that you care about. Of course, there is learning that will require external data, but maybe we can just have this thing talk to itself and make itself smarter.

Dwarkesh Patel *02:02:03*

So here's the question I have. What you've just laid out over the last hour is potentially just like the big next paradigm shift in AI. That's a tremendously valuable insight, potentially. Noam, in 2017 you released the Transformer paper on which tens, if not hundreds, of billions of dollars of market value is based in other companies, not to mention all this other research that Google has released over time, which you've been relatively generous with.

In retrospect, when you think about divulging this information that has been helpful to your competitors, in retrospect is it like, "Yeah, we'd still do it," or would you be like, "Ah, we didn't realize how big a deal Transformer was. We should have kept it indoors." How do you think about that?

Noam Shazeer *02:02:51*

It's a good question because I think probably we did need to see the size of the opportunity, often reflected in what other companies are doing. And also it's not a fixed pie. The current state of the world is pretty much as far from fixed pie as you can get.

I think we're going to see orders of magnitude of improvements in GDP, health, wealth, and anything else you can think of. So I think it's definitely been nice that Transformer has got around.

Jeff Dean *02:03:39*

It's transformative.

Noam Shazeer *02:03:51*

Woo. Thank God Google's doing well as well. So these days we do publish a little less of what we're doing.

Jeff Dean *02:03:54*

There's always this trade-off: should we publish exactly what we're doing right away? Should we put it in the next stages of research and then roll it out into production Gemini models and not publish it at all? Or is there some intermediate point?

And for example, in our computational photography work in Pixel cameras, we've often taken the decision to develop interesting new techniques, like the ability to do super good night sight vision for low-light situations or whatever, put that into the product and then published a real research paper about the system that does that after the product is released.

Different techniques and developments have different treatments. Some things we think are super critical we might not publish. Some things we think are really interesting but important for improving our products; we'll get them out into our products and then make a decision: did we publish this or do we give kind of a lightweight discussion of it, but maybe not every last detail?

Other things I think we publish openly and try to advance the field and the community because that's how we all benefit from participating. I think it's great to go to conferences like NeurIPS last week with 15,000 people all sharing lots and lots of great ideas. We publish a lot of papers there as we have in the past, and see the field advance is super exciting.

Dwarkesh Patel *02:05:29*

How would you account for... so obviously Google had all these insights internally rather early on, including the top researchers. And now Gemini 2 is out. We didn't get a chance much to talk about it, but people know it's a really great model.

Jeff Dean *02:05:53*

Such a good model. As we say around the micro-kitchen, “such a good model, such a good model”.

Dwarkesh Patel *02:05:57*

So it's top in LMSYS Chatbot Arena. And so now Google's on top. But how would you account for basically coming up with all the great insights for a couple of years? Other competitors had models that were better for a while despite that.

Jeff Dean *02:06:16*

We've been working on language models for a long time. Noam's early work on spelling correction in 2001, the work on translation, very large-scale language models in 2007, and seq2seq and word2vec and more recent Transformers and then BERT.

Things like the internal Meena system that was actually a chatbot-based system designed to kind of engage people in interesting conversations. We actually had an internal chatbot system that Googlers could play with even before ChatGPT came out. And actually, during the pandemic, a lot of Googlers would enjoy spending, you know, everyone was locked down at home, and so they enjoyed spending time chatting with Meena during lunch because it was like a nice, you know, lunch partner.

I think one of the things we were a little, our view of things from a search perspective was these models hallucinate a lot, they don't get things right a lot of the time- or some of the time- and that means that they aren't as useful as they could be and so we'd like to make that better. From a search perspective, you want to get the right answer 100% of the time, ideally and be very high on factuality. These models were not near that bar.

I think what we were a little unsure about is that they were incredibly useful. Oh and they also had all kinds of safety issues, like they might say offensive things and we had to work on that aspect and get that to a point where we were comfortable releasing the model. But I think what we didn't quite appreciate was how useful they could be for things you wouldn't ask a search engine, right? Like, help me write a note to my veterinarian, or like, can you take this text and give me a quick summary of it? I think that's the kind of thing we've seen people really flock to in terms of using chatbots as amazing new capabilities rather than as a pure search engine.

So I think we took our time and got to the point where we actually released quite capable chatbots and have been improving them through Gemini models quite a bit. I think that's actually not a bad path to have taken. Would we like to have released the chatbot earlier? Maybe. But I think we have a pretty awesome chatbot with awesome Gemini models that are getting better all the time. And that's pretty cool.

Dwarkesh Patel *02:08:54*

Okay, final question. So we've discussed some of the things you guys have worked on over the last 25 years, and there are so many different fields, right? You start off with search and indexing to distributed systems, to hardware, to AI algorithms. And genuinely, there are a thousand more, just go on either of their Google Scholar pages or something. What is the trick to having this level of, not only career longevity where you're having many decades of making breakthroughs, but also the breadth of different fields, both of you, in either order, what's the trick to career longevity and breadth?

Jeff Dean *02:09:46*

One thing that I like to do is to find out about a new and interesting area, and one of the best ways to do that is to pay attention to what's going on, talk to colleagues, pay attention to research papers that are being published, and look at the kind of research landscape as it's evolving.

Be willing to say, "Oh, chip design. I wonder if we could use reinforcement learning for some aspect of that." Be able to dive into a new area, work with people who know a lot about a different domain or AI for healthcare or something. I've done a bit of working with clinicians about what are the real problems, how could AI help? It wouldn't be that useful for this thing, but it would be super useful for this.

Getting those insights, and often working with a set of five or six colleagues who have different expertise than you do. It enables you to collectively do something that none of you could do individually. Then some of their expertise rubs off on you and some of your expertise rubs off on them, and now you have this bigger set of tools in your tool belt as an engineering researcher to go tackle the next thing.

I think that's one of the beauties of continuing to learn on the job. It's something I treasure. I really enjoy diving into new things and seeing what we can do.

Noam Shazeer *02:11:10*

I'd say probably a big thing is humility, like I'd say I'm the most humble. But seriously, to say what I just did is nothing compared to what I can do or what can be done. And to be able to drop an idea as soon as you see something better, like you or somebody with some better idea, and you see how maybe what you're thinking about, what they're thinking about or something totally different can conceivably work better.

I think there is a drive in some sense to say, "Hey, the thing I just invented is awesome, give me more chips." Particularly if there's a lot of top-down resource assignment. But I think we also need to incentivize people to say, "Hey, this thing I am doing is not working at all. Let me just drop it completely and try something else."

Which I think Google Brain did quite well. We had the very kind of bottoms-up UBI kind of chip allocation.

Dwarkesh Patel *02:12:39*

You had a UBI?

Noam Shazeer *02:12:41*

Yeah, it was like basically everyone had one credit and you could pool them.

Gemini has been mostly top-down, which has been very good in some sense because it has led to a lot more collaboration and people working together. You less often have five groups of people all building the same thing or building interchangeable things.

But on the other hand, it does lead to some incentive to say, "Hey, what I'm doing is working great." And then, as a lead, you hear hundreds of groups, and everything is, "So you should give them more chips." There's less of an incentive to say, "Hey, what I'm doing is not actually working that well. Let me try something different."

So I think going forward, we're going to have some amount of top-down, some amount of bottom-up, so as to incentivize both of these behaviors: collaboration and flexibility. I think both those things lead to a lot of innovation.

Jeff Dean *02:13:49*

I think it's also good to articulate interesting directions you think we should go. I have an internal slide deck called "Go, Jeff, Wacky Ideas." I think those are a little bit more product-oriented things, like, "Hey, I think now that we have these capabilities, we could do these 17 things."

I think that's a good thing because sometimes people get excited about that and want to start working with you on one or more of them. And I think that's a good way to bootstrap where we should go without necessarily ordering people, "We must go here."

Dwarkesh Patel *02:14:32*

Alright, this was great.

Jeff Dean *02:14:34*

Yeah.

Dwarkesh Patel *02:14:34*

Thank you, guys.

Jeff Dean *02:14:35*

Appreciate you taking the time, it was great chatting. That was awesome.