Introduction

Patrick: [00:00:56] **Last week, we heard from 99-year-old Charlie Munger on the show. Today, my guest is 21-year-old Gavin Uberti, who dropped out of Harvard to build Etched, which is one of the most fascinating companies I've seen recently. The topic of our conversation is the ongoing revolution in artificial intelligence and more specifically, the chips and technology that powers these incredible models. To date, general purpose AI chips like NVIDIA GPUs have powered the revolution, but Gavin's bet is that purpose-built chips hard coded for the underlying model architecture will dramatically reduce the latency and cost of running models like GPT-4.**

Gavin thinks that we're about to embark on what he calls the largest infrastructure build-out since the Industrial Revolution, and I won't spoil what he thinks this will unlock for all of us. It is so uplifting to me that someone so young can be working on something so big. Please enjoy this great conversation with Gavin Uberti.

More Interesting Than Exploring the Stars

Patrick: [00:01:50] **Gavin, it's really hard to know how to tackle this enormous topic, but you've become one of my favorite people to learn about the future of artificial intelligence, models, hardware, software, data centers, a million things that effect tons of public equities, tons of soon-to-be applications.**

Maybe an appropriate place to begin would just be one of your early investors told me you said this interesting line to him, which was something like you felt like you were born too late to explore the world but too early to explore the stars, and you were sort of depressed by this. But then maybe this whole world of super intelligence and AI allowed something in between or even more interesting than the stars. I would love to hear the origin of that reflection of yours and what you think about it now.

Gavin: [00:02:34] I strongly agree with this. One can't help but think that 100 years ago, 200 years ago, there were so many mysteries left to be solved, that low-hanging fruit, so to speak. Somebody smart go into their -- whatever they had instead of a garage, think for a couple of years and come up with optics or physics. You can go and explore continents.

Then as technology developed, that became no longer feasible. That stuff had already been done. And yes, you could still do interesting in novel work, but it often required going to school for 10 years before you could learn all the mathematics you needed to go do new mathematics, say, chemistry. You had to go study all that had come before, and someday, mankind of will go to the stars or be able to exploit other planets and that fruit will be low hanging too, but that stuff is very expensive today.

But I think AI is very unique because unlike mathematics and unlike chemistry, the field is so young that we haven't had 100 years to develop it, and we haven't had 100 years to solve these low-hanging problems. A single person can make a major, major impact.

And this got even better with the transformer, all this previous technology of convolutional networks and the support vector machine was thrown away in 2020 when GPT-3 came out showing that transformers really are the best way to do things. I am really emboldened by this transformer revolution. I think it is the most interesting problem of our time. And I think that it is very accessible because there's no 100 years of history built up around it.

Patrick: [00:04:09] **Can you give me your interpretation of the idea of super intelligence?**

Gavin: [00:04:14] I don't think super intelligence is some black and white thing. There are some people who say that one day the machine will be as smart as us, and the next, they'll be 100x smarter. I don't think that's true. And the reason I don't is because to go from GPT to GPT-3 to GPT-4 requires spending exponentially more compute. So I don't think you get to this crazy god-like super intelligence just overnight. I think you have to go build an enormous data center to support a thing like that, way, way bigger than anything on Earth today.

I don't believe in this concept of fast takeoff. I think that we're going to spend two years building GPT-5. And then we're going to build new data centers for GPT-6, and that will take three, five years. And then we'll keep iterating on that, slowly building better and better and better ones. And eventually, they'll begin to seem human level. They'll begin to show human-level agenticness, and then they'll become better than humans in some avenues.

GPT-4 already is a better lawyer than I am. So I think super intelligence is much more of a spectrum than it is a binary thing. And eventually, we'll look and we'll say, wow, GPT-12 is obviously super intelligent. This thing is way smarter than any human. It will be clear in retrospect. But when you're in the middle of that curve, it looks flat.

Patrick: [00:05:30] **What role do you think humans will play at that stage? I mean, obviously, like safety and alignment, all these things are euphemisms for managing fear of something that is 100x smarter than us even if that takes much longer than we think. There is no fast takeoff.**

Gavin: [00:05:4] I am super excited.

Patrick: [00:05:47] **Yes. What's the balance of excitement and fear that you have?**

Gavin: [00:05:50] It is very hard on the excitement side. I cannot wait to see what kind of books, what kind of movies, what kinds of video games a machine that's 10x smarter than me can build. If you look at the film -- somebody like Spielberg, a great director, versus some Joe Blow like myself, you say, wow, that is incredible. Imagine what a director 10x better than Spielberg could put together. Wouldn't you love to see that?

So I think there's so much to be excited about. AI is making breakthroughs in science and technology, providing some sort of universal basic income. And yes, that stuff's probably more impactful, but like viscerally, I am most excited to see what kinds of art and games and movies and TV shows these things put together.

Patrick: [00:06:36] **In that world, is there a role for people?**

Gavin: [00:06:38] Of course.

Patrick: [00:06:40] **Are we going to do anything but be the ones that are entertained?**

Gavin: [00:06:43] No, I think chess is a great analogy here. The machines have been super human at chess

for 25 years, and back when that Deep Blue first beat Gary Kasparov, people said chess is going to die whether through a lack of interest or through rampant cheating. And the opposite has happened. Chess has become more popular today than it was back then.

Go as well. Machines became super intelligent at Go in 2016, and yet there was no death of all the human Go players. People just said these things are better at it than us, but there's still a lot of value to be extracted by humans playing other humans. And to even go a step back, machines have been faster than humans for a long time. I cannot outrun a car, but yet, I still go running. People still run marathons. There is so much value, so much fulfillment in doing a task against other humans even if the machines are, in some sense, better.

The Building Blocks Behind AI Models

Patrick: [00:07:36] **Can you explain a lot of the basic terminology to us? I want to step back and think about this as an opportunity to really learn the basics of AI. It's the component technologies that make this possible -- so that we have those building blocks to talk about what they might do for us in the world, and you've got lots of really interesting ideas about what might happen.**

But let's start with the transformer. So what is a transformer? What does architecture mean in this context if you're baking one of these things on to the silicon? Start with the basic terminology of why the transformer is such a key piece of technology or a key idea that's driven so much of what's exploded in the last year.

Gavin: [00:08:15] Well, at a high level, transformer is a sequence-to-sequence model. You put in a sequence. You get out a sequence and the way that you're trying to determine what conversion is done. But that's not a very satisfying answer, so let's go one level deeper. One of the other real breakthroughs over the transformer was realizing that it takes a lot of data to teach a machine how to be helpful, how to be honest, how to give these useful responses. But we don't have to train it on just helpful, honest and useful responses.

Trick they found is imagine we take a machine and we give it the first 100 words of a document, and we say, given these first 100 words, guess what the next word is. And then given those 101 words, guess what the next word is. And we play this game again and again and again trillions of times. Now at the end of that, you're going to get a machine that is bad at guessing the next word. Maybe it gets right 18% of the time, but to even get it right 18% of the time, you have to have a huge number of concepts stored within that machine.

So they take these transformer models. They feed them a huge amount of text playing this guess the next word game, and at the end, you get a model that is able to predict the next word in science papers by understanding some of the science, the next word of Internet stuff having read a lot of web pages and so on and so forth.

And the second step on top of that is what they call RLHF. When you take the model that has been pretrained with predicting the next word and has all these concepts stored somewhere inside of it and then you ask it to, hey, mimic this text that is helpful and honest, you give it some examples of a debugging code on being helpful. And the second piece requires a lot more extensive data, takes much less computational time than that first step. So after you stack these 2 things, you get your GPT model that takes in a word, output the next word that it thinks is helpful, honest system would say.

And I think a good analogy here is children going to grade school. We want to go teach kids how to be good

investment bankers, how to be good computer programmers. But you can't really just throw a toddler and do a computer programming course and expect them to get it right. That's too hard. We need to get them to have some baseline stuff, and this baseline stuff has to be easy to grade. It doesn't necessarily have to be super relevant.

So you put the kid in grade school for 13 years, having them do tasks that are not that related but easy to get a feedback on, much like the transformer goes to the pretraining step, where it takes in a bunch of words, predicts the next one. It's not super relevant, but it is easy to grade. And after grade school or after pretraining, you have a model that understands some core concepts. And then and only then you send into college or a RLHF where you can put these other concepts on top of this and make it into a helpful, honest assistant or a good computer programmer.

Patrick: [00:11:20] **It's pretty amazing how cogent and incredible some of the answers you get when you consider that what is going on is that it is predicting the next token or the next word over and over again, that it's not thinking ahead of that token. It's sort of constantly guessing just the next one. This seems like an extraordinary limitation.**

What other kinds of models -- you've talked to me about tree search before. I'd love to hear your thoughts on that. I can't believe how powerful this thing is just by predicting a little bit ahead. Do you think we'll be able to get to a point where we predict a lot ahead and have yet more impressive answers?

Gavin: [00:11:54] I do want to clarify a misconception a little bit. Just because the model is upwards word by word, doesn't mean that they're not planning ahead. They have all these internal layers, and there's very strong evidence that there is a little bit of planning going on much like when you and I talk. I give one word at a time, but I'm able to make internal plans that you can't see about what I'm going to say after that.

And as you would expect, you have some internal plan about what words come next. It makes you better at guessing the next word. So this pretraining helps to develop this skill. Now that said, I do think there is a lot of work to be done to make these models not as autoregressive as they are right now.

Here's a good analogy. Imagine a chess engine like a Stockfish. The way these work, the neural ones that are the most advanced today, is not by just looking at a board and guessing the next position. Instead, they're going to try a huge number of possible sets of moves. It'll play hundreds, thousands, maybe millions of the games against yourself and your simulated opponent.

And at the end of all of those, you'll say, okay, this position I find myself in, how much do I like that. Then you kind of backtrack saying, okay, based on where I want to be, this was the best tree to get there. And I think that language modeling might go a similar direction, where the model makes a couple of hundred plans of, hey, here are the next 100 things I could say. I like this path the best, so I'm going to go pick that.

However, this hasn't really caught on quite yet. There's a concept called a beam search into literature that does some of this. Beam search, I think, is much more primitive. It usually has, at most, four beams. They don't look very far ahead. The first small models can be kind of helpful. And beam search also requires 4x more compute, which is a very large overhead to pay. So this beam search, tree search stuff has largely not caught on yet. Although with the rumors of the Q* learning from OpenAI, if you believe them, that does suggest that they are doing similar tree search stuff in there. That's where the star comes from.

Patrick: [00:13:59] **Maybe describe, just for the uninitiated, what Q* is, the rumor, I should say, around Q*.**

Gavin: [00:14:05] Well, allegedly, it is the combination of Q-learning into A* tree search. This Q-learning being a way to do stuff on top of RLHF and the A* being, hey, we're going to pick which of these tasks is the best. That said, I don't want to put too much credence in this. It really is just a rumor, and there are so many ways a thing like this could work.

Future Use Cases

Patrick: [00:14:25] **Talk to me a little bit about how you see the future unfolding from where we sit with a handful of very impressive models. We'll just use GPT-4 as the main one that I think people know about and have used before. I'm using this 50, 100 times a day, pretty much constantly, lots of really great use cases. But for the most part, I mean, the chat window in their OpenAI's window, I'm putting something in. I'm taking something out, and that's kind of the extent of my use.**

It's super valuable. I'm happy to pay whatever it is, $20 a month for it. But I think you see a world where these models get used in a lot more places. So I'd love you to just sort of open our minds to how you think the future might or should unfold and what the constellation of technologies will be that are required to get us there.

Gavin: [00:15:09] Well, I think the right way to think about where this tech will go is to think about how this tech got here. And there are a number of innovations that have brought us to this point, but the biggest, by far, in my mind, is scale. We have gone from a 1.5 billion parameter GPT-2 model to a 175 billion parameter GPT-3 model to a multitrillion parameter GPT-4 model. And companies seem set to spend 10x more cash training their next-generation models as well. Making these models bigger makes them better.

And now I want to call this out because this is not how AI used to work. Previously, you took a stats class in like 2012. You hear about overfitting. If you put too many parameters in your model, it will fit the data too closely, and it will get worse results when you actually test it and use it in real-world use cases.

Imagine that you're sort of teaching the test. You give it the answers to the test, and that makes it do very well on the test, but it's not going to do very well in real life. But for whatever reason, transformers seem to not fall into this trap quite so much. They are, to quote one of my teachers in college, the first capitalist AIs. Adding more money makes them better.

So based on that, I think we are going to see GPT-5 next-gen models from Google and Anthropic be smarter than GPT-4 is today, and that's going to unlock some interesting use cases. ASICs are a thing people have talked about for a long time. Imagine the GPT-4 thinking to itself and taking actions autonomously. And to date, people haven't had very much success here and because GPT-4 is not quite smart enough to get out of loops and to go do the kind of planning needed to operating on its own.

But I think we're almost there. And I think when we scale up with GPT-5 and models beyond that, we will see ASIC use case becoming more mainstream. And I also believe that this chatbot interface that I love as well is definitely not the end use case for these models. Any scenario where you are reading chat via language model, writes token by token by token is not a use case that is expensive. The model is probably going to be cheaper than human eyeballs. Human time is very expensive and the first thing you think to try.

But there are real technological limitations that stop other use cases from becoming the standard. Take, for example, speech, this podcast we're doing right now. I'd love to be able to talk to GPT-4 much like I talk to a human. It makes such a difference at talking versus descending text, so much easier to be misinterpreted then.

And while people have tried to build transformer speech-to-speech models, it's challenging because of latency. You talk to it, and there's this pause before you get your response back. And that really breaks the flow of the conversation. So I think we need tech that will enable stuff like tree search by bringing the cost of running these models down by a factor of 100.

I think we need tech that will bring the latency down by an order of magnitude, so we can go have conversations with these models. I think we need tech, allows them to generate 1,000 tokens per second and make that code writing take 2 seconds instead of 20. So I think there are so many use cases beyond this chatbot interface that are waiting to be built. The tech is not quite there yet.

Patrick: [00:18:37] **Can you talk a bit about robotics? And by robotics, I mean like physical machinery technology. I know you did some stuff with the robotics in high school. What's interesting to you about robotics vis-a-vis these models and performance in latency specifically?**

Gavin: [00:18:54] I think robotics is a very interesting use case. That's currently kind of stuck in the before times. Robotics is a field that works very heavily on feedback loops, on hard coding algorithms while having some wild loop that says, okay, move the arm until it goes and welds and then do this again and -- old tech kind of stuff.

People have tried to go and put some AI models into these robots, but it's very hard because of this lack of training data. For images, you can go feed it 1 million things from ImageNet, 1 million frames. For robotics, you can't have it break the robot a 1 million times before it figures out how to walk. That's just not affordable.

And while some people have tried to build the simulations, where the robot can go interact with the world, these don't work super well because of the differences between a simulation and the real world. I mean there a lot to understand the real world without actually having to break itself 100,000 times. But transformers, I think, are an interesting way to solve this problem.

There's a great paper out at Google, 2 years ago, I think, RT-1, proposed a robotics multimodal transformer. One of the other beauties of a transformer is that they can take in inputs in almost any modality and the modalities of things like text or images or video or in this case, a robotic sensor input. And you can train a model on many different inputs.

In this case, they pretrained the model with a huge amount of text, where they have built some understanding of the world from this guess the next word game and then on top of that, added in this robotics encoder stuff. So I think that robotics is a super interesting use case for these models. I think that when transformers are put onto robots, they won't be trained just with their robotics inputs. They'll be pretrained like today's, and you'll add those other sensors later.

It's tough though. You also need some very good response times for a robotics model that, for text, it's okay if it takes a second to go off your machine, be processed and then come back. A second is too long if you're

about to follow. So latency is so, so critical for any kind of application here. It's not quite there yet.

There's been some limited adoption. In the Tesla self-driving cars today, they use vision transformers, so much like a language model, but it takes images instead. The internals are almost the same. But they haven't hit the mainstream for robotics largely because of this latency problem.

Patrick: [00:21:25] **It seems like latency is the thing to focus on. I mean there's two roadblocks, I guess, to more huge applications, companies' use cases being built on top of this technology. One is just how smart the thing is, and that seems to be on a one-way trajectory. I'd love to hear what you think the limits to that are.**

Then the second seems to be just latency. So many of the things you described, I would say we need real-time AI or something, whether that's conversation, translation, robotics. So many of the things you've described seem like real time is the thing that's really missing here. So maybe describe how we can get over that hurdle.

Gavin: [00:22:00] I strongly agree that real-time AI is the bottleneck here. And to go a step deeper, what do we mean when we say real-time AI? What do we mean when we say we need better latency? We measure it in 2 ways: first, the number of milliseconds between when you submit your response and when you get your first token back, so you're hitting enter, and you're seeing the first answer from ChatGPT; and then the time between subsequent tokens.

Now why does the first token take so much longer? In theory, it's only running the same model. Well, that's because it's running the model on the entire prompt. If you give it, say, 200 token prompt and then it has to run all those 200 tokens before it can give you your answer back. And it's actually a little bit worse than this because imagine that you have some conversation six messages long. It's very expensive to go store all that in RAM while it's waiting for you to write your response. It's usually cheaper just to recompute it.

So if you hit enter, your chat's already 6 messages deep, you and ChatGPT talking to each other. It's going to have to feed that entire message history back into the model before it gives you your next tokens back out. So instead of being 200 tokens, even if that was your last message, it has to go compute this model on 1,200 tokens. So if we want to bring this initial delay down, we're going to need chips that have way more compute and are able to use that really efficiently.

And I think the best approach here is model-specific chips. Imagine a chip where you take the transformer model, this family, and burn it into the silicon. Because there's no flexible ways to read memory, you can fit order of magnitude more compute and use it more than 90% utilization. This lets you solve this problem and get the responses back in milliseconds instead of seconds.

Patrick: [00:23:53] **Talk a little bit more about the actual machinery of that way of doing things. This makes me think of ASICs from the Bitcoin era when those got so popular. Maybe describe what an ASIC is. But I remember these chips that became ever more specialized that literally just Bitcoin mining, just guessing hashes basically over and over again as fast as humanly possible.**

And anyone that had an ASIC outperformed someone that had a GPU, outperformed somebody that had a CPU. So maybe just talk us through. Everyone's heard those 3 terms probably but may not know like what the differences are and what the trade-offs are. What are ASICs and how does this work?

Gavin: [00:24:28] Well, let me give it to you in the context of Bitcoin. In the very old days, back when the Bitcoin algorithm was first devised, people ran it on CPUs. And to go mine a Bitcoin, like you're saying, you have to solve a very hard math problem, and the only way to solve this math problem is by guessing and checking answers very fast.

A CPU can do many things. It can do this, too, but it's not very fast about it. CPU mining will originally be a standard. People then said, hey, let's run this thing on GPUs. A GPU has several hundred mini CPUs scattered across the silicon, and because of this, it's able to go do many things in parallel. Each one of these cores on a GPU can do all kinds of different tasks.

Once people wrote GPU miners, CPU mining became so bad by comparison that you'd lose money running it. But then people said, well, what if we took this a step further. On a GPU, only a tiny fraction of that GPU is used for mining Bitcoin. We don't need the caches. We don't need the I/Os. We don't need the crazy instruction processing in L0 caches. What if we specialize a chip, put every transistor on that chip, so it would just mine Bitcoin.

And then MicroBT did this in the 2012, 2013 era. We saw an explosion in how fast we're able to solve these problems. The Bitcoin algorithm made them much harder to compensate, and GPU mining became unprofitable overnight. Because of this, GPUs have not really been seeing at Bitcoin mining for almost 10 years. That's not to go say that GPUs were scrapped overnight. They found other use cases. They're flexible. They went to mine Ethereum. They went to go do AI. They want to play games but not Bitcoin. Once you have an ASIC for something, it is just not commercially viable to run on GPUs anymore.

I think the same exact thing is going to happen for transformers. Previously, in like the 2010 era of Bitcoin, there wasn't really enough volume to justify spending $100 million to build a custom chip. But once Bitcoin became popular, there totally was. And then the chips came out, and everything else becomes no longer feasible. For transformers, even 18 months ago, there wasn't enough demand to justify spending $100 million to burn that algorithm into the silicon. But now with ChatGPT, GitHub Copilot with Character, with all these models from Google and Anthropic, it does make sense. The first transformer ASICs are coming.

Building the Physical Infrastructure Powering AI

Patrick: [00:27:01] **Can we talk about this progression? So I'm just going to use -- I'll stop saying ChatGPT just to abstract away. But let's say we're at model Level 4 right now, and we're going to go to 5 and 6 and 7. You referred to the need to probably build things that don't yet exist, not just chips, which is obviously what you're working on, not just training the models, not just gathering data but actually the infrastructure, the physical infrastructure to do this.**

What do you think happens to data centers and power sources and things like bandwidth? It just seems so interesting that we're going to have to build new stuff to enable each successive progression of new model. So describe how you think that iterative thing might play out.

Gavin: [00:27:41] I think there's a great analogy here actually, and that is a semiconductor fabrication, the buildings that make microchips. These incredibly advance facilities have incredibly complex supply chains. Many have heard of TSMC and the machines they buy from ASML for $300 million and that mirrors ASML buys from Carl Zeiss. Well, that's just the for the lithography machines. A huge array of complexity goes into this stuff.

And that means that building a new fab costs $10 billion, $20 billion. And I think we're going to see a similar thing for models of level 5 and 6. This whole chain will have to be redone. If you want to go and scale from 4 to 5, have the same multiple of compute over a 3 to 4 or 2 to 3, you have to have 10, 100x more chips, and that means 10, 100x more power.

This is not going to be a 20-megawatt data center as a two-gigawatt data center. It's a data center that consumes the entire energy output of a big power plant. And I think this is also going to be very centralized. If you compare say, the bandwidth of an entire undersea submarine cable to then between a couple of GPU racks next to each other, the GPU racks have double the bandwidth of a literal undersea cable. So there is so much incentives to centralized data centers. So I think we're going to see a few very, very large facilities that do all this.

And the same is true of semiconductors. For reference, TSMC's fabs are all very large, very self-contained buildings. But so much else has to happen, too. You need way more power. You also need way more redundancy. If you have 10 GPUs, then, sure, none of them are going to fail. If you have 10,000 GPUs, then, okay, maybe one fails sometimes, and then you reset to a previous checkpoint and you try again. You have 10 million GPUs or 10 million transformer ASICs. Then you're going to have one failing every couple of hours.

You're going to have to figure out how do I deal with one of these things crapping out without having to stall the whole thing. No one's solved that problem yet. Or heat, we've been able to go cool data centers in the 200-, 300-megawatt range. But what happens when the two gigawatts that go into that building are all turned into heat? How does that get pumped out? Is that just dumped into the atmosphere? What happens next? I really think whole set of industry's going to evolve around building these massive, massive AI models. We've only seen the beginning of it.

Patrick: [00:30:18] **All that sounds so remarkable, that heat dissipation, the power, the bandwidth, number of chips. All of that, I think, if I understand correctly, is especially needed in training. Each marginal inference is less of a centralized crazy power-intensive thing, although collectively, obviously, inference will get quite big too.**

We haven't really talked about training. And I know you're working on chips for inference to allow you to do orders of magnitude faster and cheaper inference, which will power the apps. But what about training? What is actually going on in training of one of these models for those that don't understand that process?

Gavin: [00:30:56] Well, so first, I want to go talk about inference for a second, and then we'll build into training because I believe that inference is going to be run at a similarly large power-hungry data centers, too. And here's the reason why. While we run a transformer model, we have this huge number of parameters. And each one of these parameters is a number. And to use that number, we take in a number from our input. We multiply them together, and we add them to a running total.

So every one of those parameters, in the case of GPT-3, 175 billion, is loaded from memory once and then used in the math operation once. It turns out that loading a thing from memory is way more expensive than doing the math. So how do we solve this problem? Well, we say that these weights are the same of across one user or two users or four users or eight users. So you batch together a huge number of queries. Then we load in that weight once, and we use it 16, 32, 64 times.

And that's one of the really interesting things that a transformer ASIC can do. You can have a much, much larger batch, not 64 but 2,500. So we're able to go load that weight in once, pay the expensive price and then amortize that expensive price over a huge number of users, making inference much, much cheaper. And now while this sounds good in theory, this does mean that you have to run that model in a place where you can have a huge number of users all kind of grouped together. So I think that means inference will be centralized in much the same way as training.

To go to training specifically, why are these machines so power hungry? Well, in inference, you have to run the model forwards. But with training, you have to compute how much each one of those little weights contributed to the final error. To do that you have to run the model backwards, which takes about double the compute.

Additionally, running forwards and backwards requires different kind of network primitives, different kinds of connectivity than just running forwards does. So training is a more challenging problem. But the economics that make a transformer-specific inference chip makes sense, I think, will also apply to training.

If you're spending, say, $100 million on an AI model and it costs $100 million to build a new custom ASIC, then no matter how good the custom ASIC is, you're not going to make your money back. But if it costs $1 billion to train your model, a custom ASIC costs $100 million, meaning if you're 20% better, then, yes, it does make economic sense to build that custom chip. If you're spending not $1 billion but $10 billion, it is a no-brainer. You have to do it.

Now building that custom training chip is not an easy thing to do. Right now, going from conception to a mass production of the microchip can take four or five years, and we can't wait that long for training our new models. So I believe that to support this demand, support these custom ASICs for training that we will eventually need, there has to be a whole redesign of the way semiconductors are designed.

Chip Cycles and Innovation

Patrick: [00:34:00] **I'd love to understand a little bit about how the cycle time for new chip ideation, design and then delivery has happened in the past because, from everything you're saying, it sounds like the iteration speed of chips themselves will need to go a lot faster than it has in the past in order for us to have all this great stuff that we've been talking about and that, that is a vector of innovation that's as critical but certainly like extremely critical alongside everything else we've talked about.**

So how has that worked in the past. If I'm NVIDIA or something, I'm whatever company designing some new chip, talk us through the major stages of chip design, maybe even starting with the thing trying to be solved, how you arrive at the problem to be solved in the first place. So that's step one. And step X is the chips are in a data center somewhere being used. And how long does each of those steps take? Like I'm sort of a novice on semis. And so I'd love to learn how it's done in the past, and then we can talk about how we might innovate to change that in the future.

Gavin: [00:35:01] Well, there are a few steps involved in bringing a new chip to market. No matter what year you're building in, first, have to say, okay, what's the problem we're trying to solve. And then from there, you'll devise an architecture and a micro architecture to sort of a more detailed very early architecture, of how your chip works at a high level to solve this problem.

And as part of this architecture work, you're going to get a number of blocks. You'll say, okay, in order for me to do this giant matrix multiplication, I'll need a matrix multiplication block. And that matrix multiplication block will be made up of many sub blocks that each do a multiply, add operation. It'll be made up of a control block that sends a signal to the full matrix block and so on.

Then you'll take these blocks. Number of blocks a chip has varies widely, but it can be as high as hundreds. And they'll give it to an RTL engineer, who will write code and a hardware description language kind of like a programming language that tells the block how to behave. Then this block is verified. Because it costs so much to manufacture semiconductors, you have to make sure there is not a mistake in the RTL block before you go pull the trigger and tell a TSMC or whoever it is that they're going to be cranking out wafers off their line.

So a verification engineer will very carefully write a test bench to check that, that block behaves the way that it should. And they'll also check that their test bench actually runs every single line in that Verilog file. They call that line coverage. Then they'll go a step further. That Verilog, that code will be compiled to a netlist. This is how each group of transistors talks to each other. And then you'll run toggle coverage on this. You need to make sure every one of those transistors will flip to either state.

So that's what they call the front end of the chip. Now you have a netlist that comes out of that instead of all the transistors on how they connect to each other. But now you have to put them on actual silicon wafer and lay them out. That's a hard problem. So you'll give it to a back-end engineer who will make sure that, while both beyond the logic being correct, there's no issue where, hey, this block takes too long. And it's not going to be done by the time this other block has to take its results.

They call that timing closure, and they'll make sure that, on the chip, especially a large chip, process issues, manufacturing problems in the TSMC line can make it so that one part of the chip behaves differently than the other part of the chip, even things moving at different speeds, to all make sure that it works no matter what kind of wafer is made, whether it's fast or slow, what temperature it is. They call this getting all the corners closed and a number of other checks too.

There's not going to be some weird radiofrequency interference thing. There's not going to be some pool, some vacuum in the chip or other photoresistor flow in. They check a huge number of problems. And eventually, after this back-end process is all finished, they'll give it to a mask house, mask shop who will take these designs of where your transistor goes, and they'll say, all right, to make these transistors, we need to go and have silicon in these parts of the chip but not these parts of the chip.

If you ever heard of screen printing, the process kind of works like that. From this file, this GDSII file, which says where there should be and where there shouldn't be a transistor, they make a photomask that will shine light on the parts where there are transistor and not on the parts where there aren't. And then those photomasks are sent to a fab that actually makes the chip who then uses them to protect an image onto those silicon wafers and precoating them with a chemical called photoresist that hardens when exposed to light.

So that means that, after all this is said and done, you have parts of the wafer that are hardened and just the right pattern for you to put your transistors down on them. So this is a very long process, real production samples coming off an assembly line for modern semiconductors. And if we want to build the next generation of transformer-specific ASICs for training and for inference, this has to shrink.

Now there's been a huge amount of progress to make this faster, especially on auto nodes. And Google, in

process because, really, it's not that hard. It's just checking rules, and it should be done by a computer. Then if you're building on a node like OSRM, if you're building for old technology, old semiconductor tech from 2005, you can have the back end done automatically in a day instead of nine months by using one of these automated tools.

And we haven't gotten to this point with enormous modern scale chips yet. It's possible in theory, and I'm very hopeful that more smart folks will begin working on this problem. And the same thing for actually writing the chips, doing this RTL coating. Now today, it's all done by hand, but for example, C and Python, we have seen language models themselves begin to help out with it.

GPT-4 is very good at both those languages as is GitHub Copilot because there's so much code in C and Python available online. So having access to these AI tools can make a developer much faster. For a Verilog in these hardware languages, that's not true yet. While software loves open source, hardware really doesn't. So it is very, very hard to get all of this code into your model at training data because it's all proprietary.

And I'm very optimistic that somebody will be able to go to Synopsys and go to all the old chip vendors that say, hey, your code for CPUs from only 2005. That's probably not very useful to you anymore. Can I use that to train my AI model? And if enough companies are in this to say, yes, really they'll open source it but probably not, maybe just to go train that model, then you can get a language model that was good at Verilog -- as good at Verilog as a C and Python, and you use that to speed up writing some of this RTL developments.

Patrick: [00:41:08] **What do you think the outside limits are on how fast this could happen where you said five -- four, five, six years might be the start to finish time now? Where do you hope it is 10 years from now? What do you think the upper limit is of how fast this iteration time could be?**

Gavin: [00:41:27] I really want to go call out a lot of the Google-led silicon ecosystem. And I want to call out GlobalFoundries, with SkyWater foundry for outsourcing their PDKs to help make this process fast. Now if you're a hobbyist, who wants to build a chip on one of these older technologies, you can do it from conception to putting it together a few transistors, taping it out weeks, months, even if you're starting from absolute scratch. That is so cool.

They're able to do it pretty cheap because they take all the hobbyist projects. They put them together on one mask set. You make one photomask that has all the chips on it. And I'm really optimistic that this can be scaled up. Now it's definitely not easy. These are hobbyist chips on old nodes, having hundreds of thousands of transistors on them.

And the amount of effort it takes a computer to decide where to put these things is super linear in the number of transistors, going from 100,000 to 80th billion like you have on the NVIDIA GPUs is more than 80 billion, over 100,000 times harder. So instead of being able to do it on one DP server, is suddenly a huge, huge data center working on this optimization problem to get it working.

So I am optimistic there's a future where just like for a hobbyist building a chip on an old node today, we'll be able to get companies to build chips on new nodes by spinning up huge data centers to do this back-end process and training these language models to help them write the front end. But I think the upper limit really is a couple of months instead of the 4, 5 years of today, a lot of work is going to have to happen to bring that down.

And it's also a problem where the first chips that are made with some faster process probably aren't going to work. People don't like rocking the boat because of how expensive it is to build the chip only for it to come back dead. And once a few people do it, that will be the obvious solution, but nobody wants to be first.

Patrick: [00:43:31] **Why do you want to try? I mean why...**

Gavin: [00:43:33] Because it has to be done.

Patrick: [00:43:35] **Someone's got to do it.**

Gavin: [00:43:36] Someone's got to do it. That's the only way that we get to our $10 billion training runs. That's the only way that we get to GPT6. That's the only way that we get these AI-generated books and movies and TV shows that will make the artist of today be impressed. Someone has to do it, and it might as well be me.

Market Structure and Security Implications

Patrick: [00:43:56] **It's interesting to think about in this future world now, the existing landscape of companies and stakeholders and how they'll start to interact with this technology as it becomes available. So when I went looked up the top buyers of H100 or something from NVIDIA, you see kind of exact ones you would expect. You see, I think, Meta and Microsoft are the 2 top purchasers, Amazon is a big purchaser.**

So these hyper scaling, hyper large technology companies that are either providing cloud services in the case of Azure or AWS or their Meta, training their own models, open source models, buying all these H100s, who do you think are the customers in this different world where data centers are different, ASICs are their predominant chip? Fortune 500 companies are all trying to find their version of applied generative AI. Do you think that the market map and market structure looks a lot different in terms of like is it the same buyers buying these chips as buying the H100s?

Gavin: [00:44:53] Again, there is a great analogy in the world of chipmaking. The world of chipmaking used to be pretty cheap. Anyone designing their chips probably have their own little chip fab because it costs a couple of million dollars. And as these things became more and more expensive, all but the biggest players became priced out. Now only 2 or 3 firms, TSMC, Samsung and maybe Intel are able to make the world's state-of-the-art chips. I think the same thing will happen for a generative AI and these mega-scale data centers.

Today, many companies are trying to explore probing the waters because it is so cheap to do so. But as having a state-of-the-art model becomes $100 million, $1 billion, $10 billion endeavor, it will be much more common to outsource that to one of the few firms who will have one of these mega-scale data centers. And don't get me wrong, you'll still be able to fine-tune on top of these models of these companies build, but they'll be much more like TSMC, the pure-play AI-generating house, and they will beat some full stack integrator.

I just think the world cannot support a 100 different $10 billion models, but it can support 3. So I see a world where you have these few enormous leading-edge state-of-the-art development houses, possibly at the current hyperscalers, possibly at some new company that specializes on just this kind of thing. And there'll be a tail, much like there are for semiconductors. Not every application needs leading-edge chips.

For example, the U.S. government and U.S. military care much more about chips being produced in a reliable American factory and the chips being state-of-the-art. So fabs like GlobalFoundries have accepted, they're not going to be as good as TSMC, but they're still a market for them just by being American. I think we'll see a similar thing for these AIs. You'll have a few smaller data centers that don't try to compete on being as purely smart as the big megaliths, but they'll find some niche.

Patrick: [00:46:56] **Obviously, TSMC is one of the most interesting and intriguing geopolitical assets that we talk a lot about in the news. It's even a political issue. Do you think that these data centers and the control over the specific models becomes like the central political issue like there's a presidential campaign where the candidates on the stage are mostly just talking about who has the best models and where are they located and what's the security around them and nationalistic type issues? This all seems almost like the start of a sci-fi novel or something.**

Gavin: [00:47:29] If I said that we were going to have threats of war over a chip fab 10 years ago, that sounded like a sci-fi novel, too, but this has to be the conclusions here. Some countries are going to have these hyperscale data centers. And if you don't have one, it's going to be really hard to catch up. China is famously trying to catch up on semiconductor development.

They realize how critical that is for AI, but they did not get on the EUV train when it was leaving the station. They're now on this older immersion technology, and it's really holding them back. And while they have produced some great "7-nanometer chips" I do think they'll going to need better technology.

China is not state-of-the-art for semiconductors today, and it does not seem like it's on a path to be either. And I think for data centers as well. The United States is very lucky to have all the big hyperscalers, very lucky to have the big AI companies too, but we cannot be passive about this. We're in a good state now, but that could change. So I think the export bans for state-of-the-art AI chips are really in the national interest here.

Lessons and Limits

Patrick: [00:48:38] **Can you explain the essay, The Bitter Lesson and then talk about the implications of the upper limit of how good these things could get based just on the available data in the world. These things have been trained on insane amounts of data, but there is a finite amount of data in the world. There's more being created every day. But talk about The Bitter Lesson and the importance of data and the sort of upper limits of how good these things could get.**

Gavin: [00:49:02] So The Bitter Lesson is an essay put out by Rich Sutton, great pioneer of AI in 2019 before the whole transformer craze. Rich looked at the field and saw what would happen all the time if somebody would take a model that worked, and they'd hard-code some knowledge into it. For example, somebody would say, "hey, here's a great image recognition model, but it gets this one weird case wrong on occasion, I'm going to hard code a hacky fix on it." And inevitably, doing this whack-a-mole fix strategy works in the short term. You get a little bit better performance. You can go publish your paper. All is well. But it never scales.

The only techniques that have routinely worked are those that levered the fact that compute is getting cheaper. For example, I mentioned chess engines before. In the very old days, people were rating chess engines by saying, well, a rook is worth five points, a bishop is worth three. I'm going to add a rule to my machine that says, "hey, if you can go trade a bishop for a rook, that's a good trade to make. So you should go and make that move." And again, that works a little bit, but it doesn't really generalize.

So the strategy actually worked for chess that enabled Deep Blue to become the World Chess Champion was not encoding human knowledge, not hard coding rules, but by just leveraging a massive amount of competition, Deep Blue did search, much like this tree search we were talking about earlier. It would test hundreds of thousands of positions and the one that it liked best and then play moves to get there. And the same thing is happening for artificial intelligence.

In the very old days, the image recognition, people will say, "hey, there's a block of white pixels in this kind of shape, it might be a bird, it might be light bulb." They had these big trees of fixes for it. And that didn't really work, didn't scale up. But the thing that did scale up is making neural networks very big and feeding them huge amounts of data. So the lesson of The Bitter Lesson is that the only techniques that will really advance the field of artificial intelligence are those that are able to leverage the cost to compute getting cheaper that are able to leverage these enormous data centers that must be built.

And you mentioned data as well. A key part of the learning algorithms is that data. And there's only so much data available in the world. Some people have raised the alarm that we're about to run out. The models aren't going to be able to go and get the next trillion tokens. It doesn't exist. And yes, I do think that will be a problem eventually, but we haven't even begun to tap into the most valuable source of data, the video.

I don't know about you, Patrick, but when I learned to walk, when I learned to pick up objects, I didn't do it from a book. I did it by looking with my own eyes and seeing, hey, does this behave the way that I think it does. And we've begun to kind of dip our toes to do it. People have built vision transformer models. And usually what they do is they train the model on text. And at the very end, they add in the image part that's just as a little side feature. And I think that's wrong.

I think transformers tomorrow will be trained from day 1 with enormous amounts of video data. So that solves the problem in the medium term. But eventually, these models will get 100x bigger, they will require 100x more data. They will exhaust all that is available right now. But I think that self-play and generating their own training data is the final frontier here.

One good analogy is the machines that play Go. They try training AIs on human games to make them better than humans at the game of Go. And it worked. We ran out of those games very fast. So what Google did with AlphaGo is they made the model play against itself and learn from those games.

And the same trick has begun to show some promise in natural language processing, where you have an AI that writes out some output. Then you ask you to analyze that output and say, was this good or was this bad? As a human, you can do that. You can get better. That's why I revise the things that I write. And I think that technique of the models, generating their own training data will allow us to scale even once we run out of all the images and video available on the web.

Patrick: [00:53:23] **When we get to that stage, though, what's the objective function in Go and chess? We know what winning is, we know what good and bad is. So we can define it. I suppose we could define the rules of physics or something in an open-ended play yourself game that these crazy models might do it themselves. These are goal-seeking things. How do we define good objective functions in that case?**

Gavin: [00:53:44] Well, one of the beauties of language models, I think, is that they learn these concepts

themselves. You can test this. You can go ask GPT-4, here's a scenario. Is the human behaving ethically in this scenario? So the AI knows what is good, what is ethical. And it got the understanding by reading all the texts we've generated as a species. So when we ask it to go say, "hey, pick the response that is more ethical, pick the response that's better." It is using that concept that develops from reading all there is to read.

The Most Important Players in the AI Stack

Patrick: [00:54:17] **Can you walk me through the most important big companies of today and just riff on each a little bit? Like there's the obvious ones. There's the hyperscalers, there's OpenAI, there's Anthropic, there's NVIDIA, there's all of these different key players. I want you to pick the ones that you think are most important, really just through the lens of your own strategic thinking. Like you're building a business, there's been enormous scale benefits in this last cycle of technology companies.**

It's staggering how big the big technology companies are. How do you think strategically about the role that they'll play, how as a young upstart business, you might position yourself strategically or counter position yourself against these incumbents? Why won't those same five companies just win everything? Like this whole strategy game is really, really interesting to me. And I'm curious how you think about it or if you have certain lenses to approach it.

Gavin: [00:55:07] I know that it looks from afar like Microsoft and Google and Amazon are dominating everything and they do dominate. But that one layer of this enormously complex supply chain. Let's think about it from the top. At the very top here, we have the state-of-the AI models being trained by folks like OpenAI and Anthropic. These folks have enormously good talent. I think that's the big moat, but they don't really run the compute themselves. They get it from one of the hyperscalers. So we have the models at the top.

Below this, we have the hyperscale tech giants. Those are the ones people are scared of Google, Microsoft and Amazon are the big 3 in this domain, buying those chips, racking and stacking them, supplying data center with power and then selling that to these companies on top. Where do those companies get their chips from? Right now, it is almost entirely NVIDIA, but I think that's going to change.

AMD is coming up with their new MI300X next year. I think it's going to be a viable competitor to the H100. And companies like us are trying to replace this idea of general-purpose AI chips entirely and build specialized ones just for the transformers. So I think that us, along with some of the general-purpose guys become this layer underneath it.

But us, NVIDIA and AMD cannot make the chips ourselves. We have to go to a fab house. TSMC is the one of choice. But the chip is just one piece here. On a chip like the MI300 or on H100, the memory actually costs more than the silicon itself by more than a factor of two. Now these memory modules are made by only two or three companies as well, Samsung, SK Hynix and maybe Micron, two of those are in South Korea and Micron's American. So these people are other really, really critical folks in that supply chain.

And folks talk all the time about NVIDIA, TSMC, ASML relationship. They don't talk enough about NVIDIA to Samsung, NVIDIA to SK Hynix relationships because these memory modules are so critical for building the state-of-the-art GPUs. But then these companies themselves, where do they get their machines from. And now we get to a little bit of diminishing returns. ASML is a much smaller company than TSMC.

And there are so many others that help make these chips as well. But I think that this supply chain of model provider on top, then hyperscale data center below that, then chip designer below that and chip fab below this. This is the fundamental stack of the future. And now what I did not mention here is the end application.

What does the users see? I don't think this will be built by OpenAI and by Anthropic and by these model companies. There will be folks building on top here, this fifth level of a stack. And some of those folks will be incumbents, big established firms that want to build some new AI products.

A lot of it will be start-ups who say, hey, chatbots are boring. I have this new interface, speech to speech. I have this new interface, enormous amounts of document injection, query. I have this new interface, AI power search with the whole Internet. So I think there's amount of opportunity.

I think there's opportunity at all these levels, building new AI products, innovating on AI models, building new kinds of data centers that are designed to operate these crazy high scales, building specialized chips and building them fast, running these models and fabricating those chips in a way that is much faster than today's processing use.

Patrick: [00:58:52] **Given that the application developers at the top of the stack have kind of an open greenfield, which is really cool. I'm sure we'll see amazing things. The way they build competitive advantage and succeed seems kind of clear, just like it has been in the past, you're operating at a much deeper part of the stack. How do you think about building a better product, but also doing so in a way that 3 years from now, NVIDIA doesn't just get one of them and look at it and say, okay, well, we're the scale player here. We sort of have the right to win in ASICs, just like we did in GPUs. How do you think about that strategically?**

And I know the world needs all these things, and it's a big world, and there's a big and growing pie. I don't mean to suggest anything different. But how do you think about competing at a deeper part of the stack where there's more supply chain, more interdependency, huge giants that move fast and so on?

Gavin: [00:59:38] Let's be clear, NVIDIA is a fantastic company. And I believe they will eventually build specialized transformer chips mostly like what we're doing. It just makes sense. But right now they are stuck in the innovators dilemma. The H100 is selling like hot cakes and has an enormous margin on it. And if they were to build, not even a trend with a specific ASIC, but say H100 for inference or H100 with a little bit of a graphic stuff ripped out, that would reduce the margin they have on the H100 and would -- it will be bad for them entirely. They can't innovate or the advantage goes away. And this is especially true for specialized chips.

One of the biggest moats for NVIDIA is CUDA. Their software stack that is very, very flexible. It's able to run graphics, able to run AI crypto. That is a great moat in the world where general-purpose chips, general-purpose AI chips are dominant, but specialized chips like ours are much easier to program. In a world where you burn the transformer architecture into the silicon, you don't need CUDA in the same way. If the world goes to specialized chips, NVIDIA kind of loses that moat.

So while I do believe they will compete, they're not going to be first. The only people who can really innovate are startups like ourselves. So let's see a world where we get to market first and NVIDIA gets to market a year later or so.

Now what happens in that year. I think, again, going to the crypto analogy makes a lot of sense. A lot of companies, 30, I think, tried to build bitcoin mining ASICs. They're much simpler than GPUs. You can take them out much faster. But the companies that got to the space first ended up winning. There is now this duopoly between MicroBT and Bitmain. Bitmain, they're testing the waters for IPO a couple of years back was valued at more than $50 billion.

So because when a specialized chip move into a space, general-purpose devices become no longer competitive. The person who gets there first is almost guaranteed to win. And I think that to go back to us. This means we have to move as fast as possible. Back when we began this company last year, before transformers were big, before this was obvious, then yes, we could afford to be a little more relaxed. But today, when people are saying, "why is there no transformer ASIC?" We must move as fast as possible, ship on time and ship silicon that works for us to win.

Patrick: [01:02:06] **You mentioned the importance of CUDA for NVIDIA. We haven't talked about the software that you have to build at the same time as you're building hardware. What role does that software play for a company like yours? And the way that the chip itself can be programmed or tapped or accessed by other developers, say a bit about those two parts of the story.**

Gavin: [01:02:27] In terms of the question of why did other AI companies fail? People have tried to build general-purpose AI chips in the past and have generally lost out to GPUs, Graphcore, SambaNova, Cerebras and Groq have had some modest success, but have not taken over the world, the way that folks thought they would. And I think this is two reasons. First, the software. The second the chips generally weren't better.

We'll do software first. It is extremely hard to do what NVIDIA does. CUDA has been developed over -- I think it came out in 2007, more than 15 years. And because it is so old, it has become stable, reliable, the workhorse people can depend upon. A start-up trying to get into the space won't have that. We'll be putting out something a little bit more raw that's much harder to use.

And if you look at like say, the ease of compiling code for NVIDIA versus even say AMD, it's no contest. CUDA is great because it is production-grade. And the only way to solve this problem besides being old, I think, is to be less ambitious. If you're doing a simpler thing, you can get to real production great quality much faster and luckily for us and for specialized chips because there is so much less programmability on the thing because you're only taking transformers to transform or compiled code, the software stack is much, much easier.

Now that's not to say I want to be complacent about it. We already have a prototype of this running internally, running with our chip in simulation. And because of this, we can begin iterating on that software, getting it stable, reliable production grade so that when the chips come back, the software works the way you expect it to.

But the second piece of why these companies didn't succeed is that they weren't better. If you look at, say, the amount of memory bandwidth that they have versus the GPUs they compute it against, it's the same, if not worse. If you look at the amount of flux compute they have. Again, it's the same, if not worse. And that, I think, is a much bigger problem than just the software itself.

Patrick: [01:04:37] **If you think about the performance improvement that you want to drive, you just said, "but these things didn't work because they weren't better." How much better do you want to be? Is it 5x, 10x?**

Gavin: [01:04:49] It's got to be more than 10x. I don't want to share exact numbers right now. We've got to have some secret sauce. But because we're able to specialize, you can get a huge amount more compute on these chips. And let me just give a concrete example here. It takes about 10,000 transistors to build a feasible to play add unit. That's the building block of any metric multiplier.

And NVIDIA has not that many on their chip. Only about 4% of that chip is the metric multipliers because to feed them requires so much more circuitry. That's the cost being so flexible. Or us, we will have more than an order of magnitude, more of these multipliers in the order of magnitude more raw flops. And of course, turning that into a useful throughput, that is still not trivial. You need a very good memory bandwidth there as well.

But the reason I can confidently say, we will not fall into this trap and that we will be better is I already know the metrics on our chip. I already have PPA figures. We will just have so much more raw compute, substantially more than an order of magnitude. That's not just for compute, but also for latency. People talk a big game about, hey, if the cost of compute goes down, you can do similar more stuff.

And I do think that's true. But it's not what gets me excited. With chips that have 20x better latencies than the state-of-the-art GPUs because they're able to run a huge prompt in total parallel, that lets you build products you could not otherwise build. Let's you build this real speech to-speech machine, let's you build a machine that ingests the whole Internet in seconds instead of hours, and that is what gets me excited.

The Role of Strong Strategic Leadership

Patrick: [01:06:36] **One of the things that's so clear to me is the role of strategic leadership in making things like this happen. And by that, I mean, usually in a person, a single person. Obviously, like on the software side, Sam Altman seems to be a or the person that's really driven a lot of this forward. He clearly early recognized an opportunity and galvanized talent and a team around it and has managed them to build something really spectacular.**

I sort of think of you like the hardware person potentially that could do this, how do you think about strategic leadership? How do you lead a technical staff? I don't know how technical Sam is, but my sense is that he is more of a traditional leader not in the deep, deep weeds, how would you characterize yourself? And what do you think are the key components of a good strategic leader in the hardware side of this equation given that we're already familiar with some of the leaders in the software side.

Gavin: [01:07:31] Let me tell you a bit of the story of how we got here. And then I'll bring that into what I think a good leader has to do. When I was on my gap year working for OctoML, working as a digital nomad which is often as dorm room, we had this idea for, hey, let's build a transformer specialized ASIC and that we get a huge amount of performance improvement. We went to a big industry vet who told us that nope, not going to work. But also, he said that despite being a veteran in industry that he wasn't the guy that know for sure. But he knew just the guy to tell us why our chip would not work.

Mark Ross, his resident doubter. Mark Ross is a seriously legit guy. Kind of personally see a little bit scared. He was a CTO of Cypress Semiconductor for a time, which was sold for $9 billion in 2019. Then Mark has shipped, I think, 5 chips that have done more than $1 billion in revenue.

So we go to Mark. We show them our tech. And he says, well, it looks like this isn't going to work, but it's too early to tell. You guys should go to build a functional simulation. You guys should go write a white paper. And then he come back to me. And we can talk about, hey, here's the problem that you're going to hit. Here's why this is going to work and you guys are going to learn a lot by doing it. And then maybe from that, we can figure out how to pivot and build the chip that is viable. So over a lot of long nights, Chris and I do just that.

We write our technical white paper. We do the original architecture for the chip. And we build our functional simulation, and we go back to Mark. And Mark says, "S***, this works. But you guys are college students." He says, these are semiconductors. Just to get started, you're going to need $2 million to $3 million and a hell of a lot more after that. This isn't data caps. You guys have never built a chip before.

And then we raised $5.5 million. We went back to Mark and asked this anyone to be the chief architect. And after a series of conversations, he used to be a chief architect, and we talked him of retirement. So Mark Ross is now our Chief Hardware Architect. And from there, kind of the rest of the dominoes fell, we got Ajat, our VP of Engineering, fantastic chip guy, Intel VP before this, who was known for building these very large chips on advanced nodes with teams, it's like a dozen people.

That is not how that normally happens. One of the few good men there who can still build the chip on a budget. Then we got Saptadeep, the Co-Founder of Auradine. We got Renald, the first chip guy at Cruise and dozens of other vets from Google, Microsoft, Amazon, you name it.

What does a good leader do to kind of tie this all back. It looks from afar like Sam Altman himself is pulling all the strings at OpenAI and masterminding the whole thing. But I think it's the folks under him who are doing the magic, the folks under them who are making that into a real product. The role of a good leader is to set the vision, get the right people in those chairs and not running out of money. That's it.

Patrick: [01:10:29] **I've heard stories though, about you in meetings where you go to the 1 inch or 1 centimeter level on some very technical part of what's being designed. And I'm curious how much you'll miss that if you give it up and you just do those other things. And/or whether or not there might be a world where you being in the technical details, I think about like an Elon or something that probably can walk the SpaceX floor, the factory floor and know every freaking detail somehow. What do you think is the prospect of being a more technical hands-on leader for you?**

Gavin: [01:11:01] I think there's a lot of value in a leader being technical, understanding your team's problems, understanding of why they're doing the things that they're doing is a requirement for bringing in the right people. There aren't that many levers you can push at the top, it's very course things, to figure out the right guy to bring in, you have to understand that problem and that's what being technical gets you.

This is why Greg Brockman, OpenAI, despite being the CTO, you still see that guy coding. You can't understand how far along the model is, unless he himself has seen the internals. Even though I love the folks you work here, and I trust them completely, I can't understand how far along is design verification. How far along is this block on our chip because I'm talking to that guy, and I understand why he's doing what he's doing. And also, to be quite frank, it's fun.

Closing Thoughts on the Future

Patrick: [01:11:54] **Any closing thoughts about things in this world that would be most surprising to**

investors to end customers out there to CEOs of other businesses that aren't technology or aren't AI-related? You've gotten so deep in the weeds on what's going on and trying to drive a certain future. Is there anything that would surprise people most that we haven't talked about yet?

Gavin: [01:12:18] Just take it back to this idea of transformer is becoming the way neural networks are where it's not going to be replaced, it's going to be a building block. They're going to build, say, retrieval augmented generation on top of that to make the model smarter. If you read Microsoft paper, where they have agents living in a simulation, you're going to build memories on top of these transformers to let them remember things. So a real key storage cache.

I think that folks have talked and talked and talked about how the transformer is going away, much more productive line of reasoning is how do I build on top of this? And when I look out at on what's happening in this landscape, it is the folks who are building on top of it, we're building the most successful, coolest products, AI games based on having this memory. Pure conversational things speech to speech with the high latency than I would like today or primitive AI lore of the future. So again, the transformer is not just the next fad. This is a building block.

Patrick: [01:13:21] **What makes you most confident in that? When I asked around, I heard lots of people that agreed but why that architecture, which obviously has led to all the stuff we're talking about, so obviously, it's incredibly powerful. But why doesn't it tap out in 2 years, 3 years, 4 years, 5 years? Why might there not be a different architecture that just totally takes over?**

Gavin: [01:13:41] If the world was fair, there could be. But the world is not fair. The transformer by virtue of being the dominant architecture is suddenly we think that has way, way more humans thinking about trying to improve, trying to optimize not just people, hardware, too. NVIDIA is building a little bit of hardware support for the transformers to out master certain kinds of memory loads into their next GPUs. That is going to preferentially favor the transformer over any replacement that comes out.

And software, NVIDIA has three very highly automatic transformer libraries. That means if you want to run a transformer model, it will be super easy. If you want to run, say, some other kind of say Stable Diffusion, you have to make the model so much better, but it outweighs the fact that two, three performance improvement you get by optimizing it so heavily.

So for a new architecture to go and beat the transformer, it has to be so incredibly good that it will outweigh the hardware disadvantage it will have because transformers are favored. It will outweigh a software advantage it has because transformer libraries are so common, it will outweigh the end-user application because end users are used to working with these transformer models that it's such a heavy lift. And again, to go back to this neural networks analogy, random forests or SVNs, could have been the dominant AI paradigm, I think. But now to go cache that up to the state of neural networks, just totally infeasible.

Patrick: [01:15:11] **One thing we haven't talked about is in the world where there's a GPT-X and maybe a Llama Y and Anthropic Z. What about building models that are specialists that rely on access to a very proprietary data store. So there's companies out there or people out there that have closed access to really valuable data that could make these models better. What happens there? Do they train their own model? Do they just somehow add their data to one of the big existing models? What do you think about a proliferation of models beyond the top three, four of them?**

Gavin: [01:15:51] I think this data is very, very valuable, but it can be learned later. And the reason I'm so

confident that it's okay to learn this stuff later because humans do it all the time. If you go to work for, say, a Bloomberg or a company that has valuable financial data, you come in there having a understanding of the world around you and not having the Bloomberg secrets, and you're able to get caught up pretty fast.

I think the same thing will happen for these AI models. OpenAI will train. GPT-X, it will cost $10 billion. And then on top of that, Bloomberg will say, hey, I'm going to fine tune that with my Bloomberg proprietary data. I'm going to pick OpenAI, 10 million for this, and I'm going to use that fine-tune version. Other companies say a semiconductor company that wants to build a semiconductor-based AI for coding chips or some of the investment bank, you name it. They will fine-tune on top.

And the second piece of why I believe this is going to be the future is that there's not a whole lot of advantages training your own thing from scratch. Remember how we talked earlier about the pretraining phase when you do things that are tangentially related but easy to grade, then you do the RLHF on top of that to really make those capabilities come out.

Pretraining phase is way more expensive than the RLHF stuff on top. And the pretraining is going to look the same, regardless of whether you're an AI for finance, an AI for law, AI for therapy. There's no sense in a large company, doing their own thing from scratch, doing the expensive thing again, when it's going to look the same, you just fine tune.

Patrick: [01:17:26] **Is there anything else to talk about in hardware, edge computing or other things in the hardware world that we haven't talked about that you think are going to be important beyond just the chips and training an inference?**

Gavin: [01:17:36] Is there a market for smaller models? I think absolutely. I don't think there's a lot for medium-sized models, though. I think there's a lot of demand for a thing that's super, super low latency. Going into the server and then going back is so slow that causes problems. Much like, for example, speech recognition today. It used to be the case that when you said, "hello, Google," and then ask a question, it will send the audio to the data center. It would transcribe it and they would come back. And the latency there was annoying.

So Google built a speech recognition model on the device for better latency. Now that model is very small. And I do think we're going to see other very small transformers, for tasks like setting up the next auto complete word, doing a little bit of drafting to help with your text. These things are not going to be the crazy smart agents we're going to have in data centers. However, I don't think it's a future for medium-sized models, things in 100 billion parameter range. That's just too big to run on a device.

And if you're going to run into data center, might as well use a smarter model. And you might say, well, 100 billion is going to be cheaper. But I think there are enormous economies of scale here because the cost of loading in those weights is so much higher than the cost of doing marginal extra computation, it is really easy to kind of get added to that batch. So I think we're going to see promising future for edge and a promising future for the biggest of the big and these hyperscale data centers and nothing in between.

Patrick: [01:19:04] **Gavin, I look forward to, I don't know, hopefully, decades of learning from you about all the stuff that's happening in this world. It does seem to just be the most -- certainly for me, the most interesting, entertaining, impactful, incredible explosion of technology that I've come across. And it's so cool that you're going to be driving the hardware side forward as folks like Sam and others drive the software side forward. I always ask the same traditional closing question at the end of my**

interviews. What is the kindest thing that anyone's ever done for you?

Gavin: [01:19:30] The kindest thing anyone's ever done for me, this guy, Gunnar Mein. When I was a kid in the middle of high school, I was very interested in engineering, but young, it's hard to go turn that into impactful results. Gunnar dropped out of college in Germany, went to work from Microsoft for a couple of decades. And then decided that instead of working from Microsoft, he wanted to go teach high school, which is quite a pivot. He went and taught programming classes at my high school, which I took and they were good. But really, he stayed very, very late to run the robotics program to run the senior stuff.

Gunnar thought that a bunch of high schoolers release could build a nuclear fusion. They go turn a deuterium into a helium, real nuclear fusions. They wouldn't generate power. These things haven't built before. But they thought that a bunch of high schoolers could do it was a little bit crazy. Gunnar Mein cited that my school needed an electron microscope.

And we didn't have the money to buy one, but he saw a $1,000 broken joule 87 machine that was a user for parts being sold in Everett. He and some of my friends and myself, drove over in a U-Haul, picked up the machine, it weighed several thousand pounds drove it back to the school and spent the next 1.5 years fixing it.

We got it working actually. It was a very old microscope. It had a polaroid camera. He cannot believe that. It would flash picture one at a time as it scanned across the image because the pixel will be exposed, it really to be turned into the full image. And we ripped that out, replaced it with an Arduino. We tried to go in to rip out the high-voltage box, replaced it with a newer one because it kind of broke sometimes. We got that working. We never figured out how to keep this from blinking, but that's what we had. The whole microscope would take about 30 minutes to get running.

You have to go to the room, turn the machine on and wait for the vacuum to be drawn and turn the fusion pump for that to be pumped out. It had to be done totally manually because the automated box had broken. Gunnar would go there and supervise us while we messed around with that thing until like 9:00 p.m. at night, almost every night. And that got me interested in engineering, got me to Harvard in some ways and got me to where I am right now.

Patrick: [01:21:48] **What an awesome story. So incredibly unique. What a cool guy, so many stories like this of someone taking an interest in teaching and sharing the world with someone so young. Amazing closing story. Gavin, thank you so much for your time.**

Gavin: [01:22:01] My pleasure.