Welcome to FYI, the four-year innovation podcast. This show offers an intellectual discussion on technologically enabled disruption because investing in innovation starts with understanding it. To learn more, visit arc-invest.com.
Arc Invest is a registered investment advisor focused on investing in disruptive innovation. This podcast is for informational purposes only and should not be relied upon as a basis for investment decisions. It does not constitute either explicitly or implicitly any provision of services or products by arc. All statements may regarding companies or securities are strictly beliefs and points of view held by arc or podcast guests and are not endorsements or recommendations by arc to buy, sell, or hold any security. Clients of arc investment management may maintain positions in the securities discussed in this podcast.
Hello and welcome to FYI, Arc's four-year innovation podcast. Today, we're super excited to have James Wang back on the podcast. James covered AI at Arc. He's an Arc alumni. He spent the past couple of years working in crypto and now he's gone back to his roots into the AI space. I can't think of a more exciting time to do that. So, James, great to have you back.
Thank you, Frank. It is so cool to be on Arc's FY podcast. It is like gone full circle.
Yeah, this should be a fun one. So, why don't you tell us about the company you're working with now, Cerebrus? Yes, it's funny. I discovered Cerebrus while working at Arc. I think it was maybe 2017 in Europe. It was called Nips of the Time. It's in Long Beach. I remember a reporter from the information asked me about Cerebrus and if you know what they're doing and I'm like, I haven't heard of them, but I'm not sure what they're doing. But that's when I first kind of got this company on the radar. And of course, very soon they launched their wafer scale AI chip. And it was the most, I mean, it is no exaggeration to say it is the largest leap in semiconductor manufacturing in history because we went from a chip that's the size of a postage stamp to a chip the size of an iPad. That has never happened before. And it didn't come from an industry giant, as you might have expected, like Intel and Nvidia or whatever. It came from a company that was stealth and backed by Benchmark. So, it was just mind blowing as a moment. And I was an AI analyst at Arc at the time. I covered their hardware extensively, both the first generation and second generation. And I've always kept it in mind as the debate over can AI hardware companies
take share from Nvidia has happened. They've done some amazing work since. And when GPT happened
in December last year and mid-journey happened, I just knew I had to get back into AI. My goal was to always get back to the industry side of things. And I just needed the right way. And I also wanted to move back to the Bay Area. And Cerebrus was like literally both things. In one, it was perfect. And that's why I joined. It's like my second month here.
That's awesome. Yeah. Convergence of a lot of stars aligning, I would say it sounds like.
So, you mentioned wafer scale computing. What does that really mean and talk a little bit more about how that differentiates from, like, let's say what Nvidia is doing?
Yes. If we step back and look at the AI landscape, what happened? And AI took off circa 2011, 2012 after the Google unsupervised cat recognition system happened and some Microsoft voice stuff happened. And the business has been just growing nonstop year after year. Around 2015, VCs and other, I guess, founders took notice and they're like, look, Nvidia makes a thing called a GPU, the G stands for graphics. There's no reason why this thing should be

ideal for AI. The software has made it usable for AI, but we can do a better job. So companies like Cerebrus, Graphcore, GROC, a bunch of companies got funded in this 2015, 2016 era with the idea of taking a piece of this AI pie. And I've observed this like while at ARC extensively. I'm like, it's so curious. It's very hard to know if they were going to succeed. In theory, they should succeed because Nvidia's chip is optimized for graphics to this day, even the AI chip has dedicated silicon to render pixels, to convert video, MPEG, like all kinds of stuff that you don't need. And the first principle analysis was a chip built specifically for AI should do well. Among the cohort, most are trying to do, I would say, Nvidia, like a chip, but better. So they would make a chip also at the reticle limit of what an ASML machine can do around 800 square millimeters. But they would do all the silicon to graphics, not a single transistor, sorry, all the silicon to AI, not a single transistors for graphics or anything else. Companies like that are like Habana, which was acquired by Intel, AWS systems like Tranium, Google systems like TPU, Graphcore, all of them are like chips this big, and they try to optimize for AI. Cerebrus is unique because it is the only one that doesn't look like that at all. Cerebrus did not try to make a better chip that is for AI. It completely reimagined the limits of what the chip could be. All computer chips are cut out of, roughly speaking, 300 nanometer circular wafers, circles, and you have tiny little rectangles in the circles, and you cut them out and each is a chip. Cerebrus is like, forget about cutting out little chips, we will just cut out the corners of the circle into one big square, and that is our chip, an entire wafer scale computer. And for AI, this is incredibly helpful because especially in this day and age of large language models, this advantage wasn't quite as clear a few years ago. A few years ago, like with image recognition, voice recognition, those kind of networks, they tend to be small, and GPUs have a decent amount of memory, 40 to 80 gigabytes. You can fit the models in the memory, and if you can fit stuff in memory, it runs fast. If it doesn't fit in memory, it runs slow. We all have a very intuitive understanding of this because there was a time when we could feel our computers fetching from hard drive versus when it's stored in the DRAM. The second it fetches from hard drive, it's just so much slower. In the past, when AI models were reasonably sized, 50 million, 100 million parameters, they fit in a GPU, and it ran pretty well. But large language models completely changed the paradigm. A 1.3 billion parameter GPT model does not fit in 40 gigabytes of GPU memory. There's a lot of redundant tensors and states you have to store, and very soon you blow the budget, and it doesn't fit, and now it's very hard. Cerebrous's approach of making the chip gigantic means that it is ideally suited to large language models, and it can train these very fast. The architecture was kind of like in theory better for the first generation of AI, but had to be proven. When LLMs came out, the advantage became just undeniable, and we're squarely at that moment
today. Do you think they were too early, or it actually takes a lot of iteration and time to get to the point where you can run LLMs on the current generation of Cerebrous? Because I assume there's a learning process of going to this new architecture paradigm that's obviously not easy to do. Yes, they are early in a sense, but if you didn't start then, I don't think you'll be ready today. Like Cerebrous was found in 2016, where on the second generation of the hardware, you needed
to learn from the first generation. You needed time to build out the whole software stack. This

stuff just doesn't happen overnight. Google's on the fourth version of their TPU. In hardware, I think you kind of need to be ready to build three generations of hardware or don't even bother at all. It's kind of like the Falcon rocket. The Falcon 1, the first three blew up and had one last shot before they were going to run out of money. You do need a few generations to iterate, and I think finally we're at the point where we have a level of hardware software all the way down to model maturity that we have something very compelling to show to the world, which is exactly what we
did last month when we released the Cerebrous GPT family of large language models. This was kind of our coming out party from a software perspective. We've had many coming out parties from a hardware
perspective. We've held up the chip. We've won awards. We're in the San Jose Museum of semiconductor
history. It's incredible, but it is so different to have a chip you can hold
versus have models that are large and have trained and have converged. Each of those steps are... I appreciated them theoretically when I was covering the space as an analyst, but on the inside, it's so visceral that you can't help but really feel it. You can just tell the difference because if you go on Hugging Face, which is like the GitHub for AI models, look around and see what large language models companies have posted. Companies like the likes of Habana, which was acquired for over a billion dollars, they have models that are 300 million parameters on there. We have a 13 billion parameter model. It is discontinuous like the level of work needed to go from small and to go from large. Yeah, it's an interesting trend where a lot of the hardware companies are now pushing higher up into the software stack, both to prove the use case, but also to force you to get the drivers and the full stack fully integrated to be able to train these models. You mentioned this. You've released Cerebrous GPT, which is a series of, I think, seven models from small millions of parameters up to 13 billion. What was the reason for that and talk more about the nature of the release that you did? Yeah, I think it's worth to go over maybe just the history leading up to this point for LLM development. We first got the inkling that large language models were a useful category with GPT-1, where if you just train transformers on unsupervised text, just text completion, it kind of starts to pick up the structure of language and starts to be a language understanding system, which was not even the explicit intent of the system. This is a very serendipitous outcome. With that initial insight, OpenAI just knew that if you scale it up, it would perform better at the baseline tasks, and at some level, emergent properties would appear. It's not just gets better at A, suddenly B, C, D abilities appear, like emergence. Very, very exciting stuff. It's exciting because normally in AI or any software development, to build a new feature, you have to explicitly go do that thing. It doesn't just magically appear. If you do more of A, B doesn't magically appear. So every new feature required extra work. What's amazing about LLMs is if you just keep doing the same basic stuff at greater scale, new functionality appears. What started as like, I can detect sentiment in an Amazon review, can now do text summarization. And then at another level, if you keep doing it, it can translate from German to French. There was no intention for it to translate from German to French. So this is a very, very exciting development because it meant software productivity was going up, and it could scale on its own without humans being in the loop. And the whole deep learning and experience from the start has been ways of removing humans from

the loops of the machine can learn on its own. And that was doing it at a meta level. So after GPT-2, open AI was like, this technique works. We'll just keep doing it. And the reception like from the AI community was you can't, it's like the meme. It's like the angry meme. No, you can't just scale up models. You have to have insight and you have to be responsible and you have to understand semantics. You can't just merely be scaling models, the crying face meme. But open AI isn't like, no, we can just do this. And in fact, we'll do a set of empirical research that gives us a sense of how the scaling works. Like if we scale up model size by X, what kind of performance can we get? And does this continue? Is this a continuous phenomenon? And this is the famous scaling laws study from open AI, Kaplan 2020. And they basically ran the different models of different sizes, different parameters, and see if the model performance continues to scale as you increase the parameter count of the models. And they found that it worked across, I think, seven orders of magnitude, a log scale. And there was no diminishing returns. It was just like on a log log chart straight line, which was incredible. It was just completely unexpected. In my opinion, this result, a scaling law, now referred to as scaling laws broadly, is kind of the most significant natural law that we discovered in this century. It's comparable to all the natural laws we discovered in the realm of physics in the last century, like the laws of gravity, quantum physics, all those things describe natural phenomenon in the world. And those laws were highly influential for all the reasons downstream. But this law, what's important is all those laws are empirical. They are not like laws of theory, or they started as theory, but they were all confirmed through experiments. So if you run the experiment, you plot the results, it agrees with the laws. It has no opinion. It's not like I like it or don't like it. This is just how the world works. And this one is exactly the same. It's empirical. It's not like Moore's law is empirical. But it's basically a measure of what happens in large language models with different results of different amounts of data versus parameters. And it basically gives us a roadmap. ML researchers a roadmap to say, if I inject this much compute into the system, what kind of performance can I get? And it will keep going. So you can keep doing this for 10 years. And the industry now can have a roadmap for progress for the next 10 years. It's kind of like Moore's law, where it's just like we've been observing transistor densities improving, doubling every two years. And if you keep going, eventually you get to this level of performance, all of Kurzweil's projections are based on this style of argument. And now we have that in the realm of AI, which is unprecedented because AI was a very lumpy industry or research area before. You would have occasional breakthroughs, occasional papers will come out, but then you have a dead decade, like a decade where no one worked on neural networks. Now we have a predictable roadmap of how to improve performance for seemingly many orders of magnitude, enough easily to occupy us for the next decade. So it was just such a turning point for the industry. And immediately following that, everyone started working in concert with this will become a self-fulfilling prophecy, just like Moore's law. Everyone trained larger models. They were like Microsoft NVIDIA models that went to 500 billion parameters. It became a chase of model size for the next like 18 months, until the group at DeepMind, you know this, came out with a kind of counter paper called the Chinchilla paper, which said you don't actually want to just increase the size of your models continuously. There's an optimum ratio for the amount of data versus the size of your models, and that ratio is 20 tokens per parameter.

Hugely influential paper, what, March 2022. So it's like just a year old for Chinchilla, but completely like the industry was on this race to one vector. And then DeepMind comes out with this paper that thoroughly like, I guess debunks that some of the core directions taken in the open eye paper. And now we're like going a complete different direction, almost maybe the opposite direction. So instead of having more and more parameters, it's like having more and more tokens. How does Sarip's GPT fit into this? These two papers are studies done by, I guess, even though despite their names, essentially closed teams, the data sets are scraped from the internet. So in a sense, they're not proprietary data sets from the source, but they were created in a proprietary way and they're not available. You can't just download the open AI data set, you can't download the Chinchilla data set. And while the papers describe the recipes, the recipes are not immediately recipes as in how you train the models, they're not immediately replicable. Our goal with Sarip's GPT was to take this state-of-the-art research and actually reproduce it and then to give the open source version of the techniques, of the data sets, and even the hardware, so that anyone can do this work. So that it's not just this piece of theory that was released by elite AI research organizations in the likes of San Francisco and London, but it's actually something that the rest of the world can feel and reproduce. Like all great science is reproducible and AI is the great empirical science of our decade, so we wanted to make sure it's reproducible. And Sarip's GPT is based on the Chinchilla recipe, so we use 20 tokens per parameter. One key differentiation is we train it on a purely open data set, the largest open English data set with a variety called the PILE created by the great folks at the Luther AI. And all seven models are optimal at this 20 parameter, 20 token per parameter ratio, so that basically it guarantees compute efficiency. If you train for longer, you would have done better with a larger model. If you train with more tokens, you advise first, right? So it's optimal both ways. And from doing this work, we also derived our law, so we are contributing to this body of literature on scaling laws. And the Sarip's GPT law basically gives you a recipe that, given a certain amount of compute, what kind of like test loss you can expect from that. And we published that as well. So the paper is open, the data set is open, the methods are fully open in our paper. We use some like novel optimizations that like help the results converge faster. And the models are a hugging face, so anyone can use them for inference or fine tuning. So it's like the, I think the most comprehensive set of models you can use in a purely open source environment. And that is a demonstration of compute efficiency training first established by DeepMind.

And I think, you know, you mentioned this, but one of the things that's great about your release is that it's counter-trend, where open AI started very open and is getting more and more closed. So we know a lot less about GPT-4 than we knew about GPT-3.5 and GPT-3. And I think that's happening as these companies get closer and closer to commercialization. Everything becomes a

much more valuable trade secret that they don't want out in the public. And we don't know the data that it's trained on. So I think that's great. The other thing that you included in the paper is how you measure performance against the training compute. Because we've, you know, we've spent a lot of time at ARC looking over the different papers that are out there, the different measures of performance. And you end up with a very academic kind of loss function in these scaling papers, where here's how the loss goes down being kind of the measure for performance

as you increase compute. And what Cerebrus did was plot not just the loss against compute, but performance on more common sense benchmarks that are actually related to or closer to real world performance, whether it's like the Winogrand benchmark, which is human fluency. And that's, you know, very helpful to demonstrate, you know, these aren't just theoretical, but it actually passes, you know, more human tests as you increase compute as well.
Yeah, Frank, I really appreciate you noticing this point. So is actually most papers measure the scaling law on the pre-trained loss. But the pre-training is not a human task, like the downstream tasks, like reading comprehension, translation, those are the things that we actually use models for. And we're in a way, this is the first paper that shows comprehensively that scaling laws apply not just to pre-training loss, but apply to the accuracy of downstream tasks comprehensively, smoothly with predictability. So this is like, I think us, that our contribution here is really affirming that the transferability of scaling laws to task level of tasks. And what's awesome about that is you'll eventually see on those benchmarks where human level capability is. And, you know, many people talk about, you know, when is it going to be, you know, as good as a human at x or at y? And AI doesn't necessarily have to stop at human level capability. That's what we saw with kind of the previous generation of predictive AI of, is this a cat or a dog? You didn't just get to the accuracy as a human, you got to much more accurate than a human, and you can do it, you know, 100,000 times in a minute. And so that's where I
think it would be really exciting to see, you know, now with a scaling law that you can plot against an actual, you know, human grade benchmark, you can see how much compute you need, how much
money do I need to spend and amount of access to hardware do I need to actually be much superior to a human across a variety of tasks, which is pretty exciting. Yeah, having these laws for AI development is, once again, I compared to physical laws. If NASA didn't have, you know, the laws, the Kepler laws of mechanics, like planetary bodies, you couldn't launch a rocket to the moon. You need to know ahead of time what you're going to get, right? If you can't plan the whole thing out ahead of time, nothing is going to work. AI training takes weeks, months, sometimes like even longer. It costs millions of dollars, tons of CO2. If you don't know what you're going to get ahead of time, it's like launching a rocket with no plan. And now we have laws to actually guide us through the whole path of, we know, like OpenASGBT4 paper doesn't say much. But one thing it does say is we have basically laws that will tell us at what size of model we can predict what kind of performance. So we knew what we're doing ahead of time. The scaling laws are what gives AI researchers the ability to actually do productive work like that.
And the other thing that I think is really interesting that you also started to hit on is that it's helped the industry right size their models, where the trend was after the OpenAI paper to increase the parameter count. So when we say size of models, we're talking about the number of parameters. So $13 billion or $175 billion was the largest size for GPT3.
And what the Chinchilla paper showed was that if you want to use the same amount of compute, so a fixed budget, they were actually massively undersupplying the model with data and over supplying it with parameters. And you can actually achieve better performance for the same cost or equivalent performance for a lower cost if you train it with more data and a smaller size.
And that also has implications for the cost to operate the model in its life because

the size of the model, the number of parameters directly correlates to the inference cost of actually running those models in production. And so as we get closer to production use cases, ChatGBT probably being the one that everybody knows now, the actual size of the model becomes a constraint in terms of costs. And so what the industry is realizing is they can supply these models with more training data and actually make it smaller and achieve the same performance, which is pretty incredible to see that kind of learning take place in really just a couple of years. For sure. And now the pender limit has swung so far that people are now going way beyond the Chinchilla parameter ratio. Chinchilla is like 20 is optimal. And then Facebook and others are like, we can go beyond that and it becomes even more optimal from an inference perspective. So like Llama right now is all the rage, right? Llama and all its derivatives. And I think those are very exciting developments, especially for deployment on local systems. Like inference in the cloud has a cost, but inference on your computer basically has no cost. And it won't be long before we have like mini GPTs embedded in our iPhones that work offline, which is incredible. We will be able to compress all of human knowledge, all of Wikipedia and all that jazz into this mysterious representation in like a gigabyte or two or three that is on your phone locally, that even if you're like hiking somewhere in Scandinavia, you can ask it, what is the president of the United States or something like that? And it would know. And it's like a full like talking knowledge base in like matrix representation. It's incredible. Yeah, you increase the capacity of an edge device so dramatically without even having an internet connection. Because we've been relying on this kind of constant connection to the cloud to get access to those like the human knowledge question of I need to Google search this, but now I don't need to Google search. I'm just going to ask a question of the model that's directly embedded in my phone. And we're not there yet, but the costs are coming down so dramatically that we're going to be there pretty soon. We are practically there. Like we are there, we can now get it to a MacBook. And between a MacBook and an iPhone is like I think the highest in iPhone has as much memory as a like median MacBook. So it will be tokens will come out maybe a little slower. But the key thing of the model is it has to fit and the model will fit. So it's incredible. We're like living an amazing time. Yeah. And I mean, the way Apple is moving, you know, you get the same chip and an iPad and a Mac now anyways. So the the aha moment that I had for this was, I think from a from a Ben Thompson Stratecary article, and I hadn't learned this beforehand, but a diffusion model. So what you see from stable diffusion and mid journey, that type of model where you're creating images from text can now fit on an iPhone and Apple's actually embedded it into their kind of AI developer kit. And so when you had an application launch like Lenza, which did cloud inference, I believe, when it was first launched, this is the upload a few photos and it will create a library of portraits for you in a variety of styles that went viral last year. That was a big inference cost for that company to launch that doing all that inference in the cloud. But if all that inference is able to be run locally, it's not a cost to your company to launch this model. It's actually a cost on the consumer in terms of their, you know, their battery life and their phones going to get a little bit hot. So it is really this kind of aha moment when you can fit a model on an end device that's not hosted by you. Yes, for sure. Like people have been talking about edge AI for a long time and edge AI has not had a killer use case. I think this would definitely like materialize and turn into a killer use case. Yeah. So it's not likely edge AI, but what is cerebrus's role in inference?

Do the CS2 chips work as well for inference as training? I've been wondering that question.
Yeah. The CS2 can do inference, but I wouldn't say it's like target use case. The main point of the CS2 is to solve the training problem in a way that far exceeds what a GPU can do for training. And I think that's what makes it truly compelling. And that's what takes advantage of the wafer scale architecture. For inference, you kind of want like a small batch size as possible, so you get like short latency. You want to quantize down to like instead of floating point, you want to like if you eight bit integer, four bit integer, like as low as possible, it's a complete different set of optimizations. And I think you can make different hardware tradeoffs to achieve that. For training, it's all about like how to coordinate large scale computing and make a model run as a single unit. So let's imagine you're training a large language model. Let's start small. Let's say one billion, right? Because one billion fits in a GPU.
One billion, if LLMs ended at one billion, I would say GPUs are perfectly fine. There's no need to go extra exotic. Why? Because a billion parameter model fits within a 40 gigabyte GPU. And if you buy DGXBox with eight of them, you just copy the model eight times, identical model eight times across the eight GPUs, and you feed them different data, and then you average the results out as they train from that data. This is called data parallel using mini batches.
This is very, very standard practice. This is how you train like image nets and things like that. And this architecture that is pretty standard works completely fine for that. Efficiency is quite good. The problem is the L in LLMs stands for large and there is no size limitation. It doesn't stop at one. It doesn't stop at 10. This is about 100. It's no limit. The second you go over 1.3 billion, by default, it doesn't fit on GPU. So imagine you're a ML practitioner, you're not, remember, you're a person trained in machine learning. You're not a person trained in like memory partitions and high performance computing and server node load balancing, none of that. You just use PyTorch and NumPy. When you go from 1.3 billion model, you press train, it works. You go to 3 billion, you press train, bang, it just stops working. It doesn't fit out of memory. Like this is the fundamental problem that ML practitioners experience. The second they go slightly larger, it falls outside of GPU memory. And now you have to do manual work to make it work again. It is not to say it doesn't work. Obviously it does. Microsoft in conjunction with NVIDIA has trained 500 billion parameter models on GPUs. But notice it's Microsoft plus NVIDIA.
It is not average Joe in the basement, right? Because the work needed to break a large model into tiny pieces that fit individually on GPUs and then have them coordinate and glue back together, the results is incredibly complex work. It's basically a discontinuous amount of work.
So from 0 to 1 billion, it's easy. When you go to 3, all of a sudden, it breaks. So you have to implement tensor parallelism to split each tensor eight ways on a DGX server. That part is hard, but doable for a competent person. But your utilization goes down somewhat.
Once your model gets to, like, I don't know, 30 billion, now it doesn't fit even inside a DGX box. A huge server, fixed server with eight GPUs and no longer fits. Now you have to implement pipeline parallelism on top of tensor parallelism. So now you have three things going on at once. And that will get you to 30 billion. And as you scale the amount of DGX boxes, that might work up to 64 boxes. Beyond that, something else breaks, literally at every level, it breaks because quite frankly, it's not supposed to happen. The thing is, the AI model, the software you're building is orders of magnitude larger than the hardware you're using. Of course, it doesn't fit. Of course,

it doesn't work. Of course, you have to hire Ninja programmers to glue it all back together, break it up, glue it back together. That's the game, the state of large language model training on conventional hardware. That's most of the work has been done this way. And that's why only few teams, few elite teams have done this work because you have to be elite to do this kind of work. OpenAI has done it. Anthropic has done it. You know, Maryland shot down the street has not, they haven't even gone to tensor parallel. This is just incredibly complicated. Contrast this with what we used to call software 1.0 developer, right? You write JavaScript. Like, you write simple JavaScript, you press enter, it runs on the CPU, it works. You make the JavaScript 100 times longer and more complicated, you press run. It's a bit slower, but it still works. Graceful scaling, right? Because it's single threaded and it's not like this thing that has to sit in memory. The fundamental advantage of Cerebrus, and it took me honestly six weeks to understand this, being on the inside even, because this is such a subtle thing, is that Cerebrus, the chip is 50 times larger than a GPU. It has 40 gigabytes of memory on board the chip as SRAM. So short-term
memory, if you will, 40 gigabytes. GPUs have 40 gigabytes of long-term memory. The equivalent of short-term memory is like 40 megabytes on RAM. So like a thousand X difference. The result of building this crazy architecture is large language models fit by default. And they fit in a special way that we call weight streaming, which is we're streaming one layer at a time.
The original architecture made the model fit entirely on chip, and that was very convenient. But the reality is large, large language models became so big, they don't even fit on our wafers. Like, we have never talked about it this way because it's like inconvenient narrative, but I'm just going to talk about this way because I think it's really important. Large language models, the second they went beyond 10 billion, they don't even fit on our mega-sized chip. That's how large they've gotten. So the amazing thing is the team at Saruba spent like a better part of a year re-architecting the whole computer, not just the chip, but the server, the entire appliance back end off this. And a few people understand what's happened.
So what we fundamentally did was we disaggregated the memory part of the processor with the compute
part of the processor. We built specialized memory appliances that are terabytes large, that are dedicated appliances in the same server as the hardware, so that we can have the amount of memory in our systems can be independent of the amount of compute in our memory. This is what is called a disaggregated architecture. You know who's a fan of disaggregated architecture?
NVIDIA. If you watch their GDC keynote, Jensen will talk about the future of the data center's disaggregated compute networking storage, except actually, ironically, they don't do it. Ironically, if you look at a GPU, the GPU is here, the HBM package is here, and here's the HBM memory, and it is completely fixed. If you buy an NVIDIA GPU, you have a, you know, whatever 300 teraflop chip, and you have a 40 gigs of memory, you can't make the memory go up without making the GPU go up. Our architecture is entirely disaggregated. We can have a two terabyte memory system, we can swap it out for a 10 terabyte, we can have a petabyte memory system if we wanted. The result of that is that we can have arbitrarily large language models without blowing up the chip. That's the key, like, insight from this. This took, like, more than a years of work to do the software and the hardware plumbing, and the result now is, whereas in a GPU, returning back to the

programmer view, the programmer view, whereas if you do one billion parameterized model, you press
enter, it works, when you go to 3 billion, it fails, and now you have to add extra code to break the model across 8 GPUs. It takes you a week to do all that, maybe it works, maybe it doesn't, but finally you have it working, you press enter on 3 gigabyte, 3 billion parameter model, it works, you're like, cool, I'm a genius, I'm going to 30 billion. Now you press enter, it doesn't work anymore. I tell, oh my god, okay, now I have to create new glue to go to the next server.
That's the GPU experience, and if you make it to the end, you can build something, and maybe it performs. The cerebrous experience is the one billion parameter model, you press enter, it works, you go to 3 billion, you press enter, it works, you go to 30 billion, you press enter, it works.
It's like, like, true elastic scaling with complete invisibility to the programmer, you don't have to add a single line of extra crud code to make the model stitch across servers, because the memory size is increased, like, is separate from the appliance, separate from the chip, and it's arbitrarily large. It's a blend of DRAM and flash memory, and this works across servers, too. So if we have four CS2 nodes, this will scale gracefully. It is data parallel only across all these modes, so that basically training on our system is infinitely easier and more efficient than on GPUs. That's pretty great. I think, you know, it's interesting comparing to the GDC presentation, you know, Jensen will say a lot of those same things, and I think he has that similar vision, but without kind of truly separating the memory from the compute, and, you know, already being ahead with a much bigger wafer, it makes, or a wafer not a chip, it makes sense that you, you know, you kind of get these efficiencies that you don't see in kind of the NVIDIA systems, say, even though that's kind of where they want to head.
Yes, we all have the same goal in mind, because it benefits all of us to make AI training easier and less friction, because it helps all our businesses. It's just different approaches.
NVIDIA approach by necessity, because their chips and hardware is built at a smaller scale, is large models, NVIDIA would train large models by dividing the large models into smaller pieces and gluing it back together with software, and this process is very complicated, and it's up to the program to do that. They take some of the heavy lifting away from you by giving you libraries to help with this work, and they built a technique called Megatron LM, which has been very influential in distributed GPU training. Microsoft built a system called DeepSpeed, and if you add, if you glue together Megatron DeepSpeed 0 and all these techniques, and you're really elite, you can get it to work eventually, but it is just so much crud work. It's like work that no one wants to do.
This is why the OpenAI paper has team for data, team for algorithms, and a special team just for training and infrastructure. These people literally don't work on the core thing. They work on the making of the thing even possible, the plumbing, the divide and glue together component. That's 30 people for GPT-4. When we trained cerebral GPT, we had one person just managing that because it
was no divide and glue. The hardware, our glue is literally the hardware itself. The architecture makes it flow naturally, so you don't have to break your models. The other thing is when you have more units that you're all having to, when a model is too big to fit on a chip, so you need more units that are all talking and settling up with each other, that introduces a network overhead, and that increases complexity and lowers performance. That's why NVIDIA bought Melanox, and that's why they're so focused on integrating the network component into the accelerator or GPU

architecture, but you can shortcut a lot of that complexity and the need for that if you have those bigger training tiles. Now that I say the word training tile, it reminds me of the other thing I wanted to ask you about, which is another team that I'm going to say is somewhere in between the cerebrous end of the spectrum and the NVIDIA end of the spectrum, which is Tesla and Dojo, where they have a D1, which is their smallest chip, and they have many of those connected very closely together into a training tile, which is, I'm going to say pizza box sized, that maybe would look to the average person similar to what cerebrous is doing. How do those compare? Exactly. I think the Dojo approach is halfway between the cerebrous approach and the
TPU approach. The pros and cons, one pro is that they can focus on the level of designing individual chips, because they just have to yield the chip. They don't have to worry about hardware redundancy as much. In a wafer, maybe they get 80% yield. In such a case, they can get 80% of the chips and throw away 20%. We have to get 100% yield. If we get the same defect, the whole wafer goes down if we don't do anything. That's why we have to build in defect redundancy into the hardware. Today, that is so built-in, the second generation, I think we yield 100% of the wafer. It's easier for them, because they can just use the traditional technique of plucking working chips off the thing, only working ones, and then reassembling all the working ones together inside of this package. That's an interesting attribute.
The fundamental electrical mechanical properties is every time you go off chip, you consume 10x the power, 10x the latency, you lose 10x the bandwidth. At the package level, it's a little bit better, but chip to chip is not like core to core, even in the Tesla architecture.
If your software is not super constrained by that kind of IO communication, this might not be a big deal. The reason why I think it is okay for Tesla to do this is they only have one customer, which is themselves, and they only have really one model, one kind of model they have to train, which is their vision model, their FSD model. In a way, the bar for Tesla to succeed is a lot lower than the bar for an independent hardware provider to succeed. Tesla has full, someone like anyone making chips and selling to another customer, you have to kind of guess, and you have a very high latency to understand what the customer actually wants. Whereas for Tesla, they have complete, it's full state of what the customer wants and what the state of the software is. They know exactly what their graph looks like, and they literally can make hardware custom to themselves.
If Tesla is a full scale company with AI chips deployed at edge devices, training for FSD, for robots, for all kinds of stuff, they're at a scale where all that makes sense.
I think it's fine. I think with the vision focus, that IO constraint may not be a big deal.
For us, it's also a reflection of the founders. The founders are hardcore, like Silicon, like VLSI people. They had a vision of solving this. They knew they could do this, and this would be a good fit. They decided to go after this, whereas Tesla didn't have to do such a thing. They're not out to break new ground in computer architecture. They just want to solve it for their FSD system, which if it works for them, it works for them.
Right. It doesn't have to be a chip for everyone. It's a chip for a very specific thing that Tesla is trying to do. The way that we look at it, even if it's just performance parity with NVIDIA, or maybe a little bit better, in housing it, they can make it 100% specific to that use case and remove that dependency, lower their cost, and be able to move at their own pace if they develop more and more in-house capability, which won't work for everybody, but for a company like

Tesla, it does. The pace is so important. I argue this when I wrote about it at ARC. It's so different when you can just turn around and say, do you need the chip to do this? And Andre can say blah, or whoever can say blah, and think about it. If your chip is coming from NVIDIA or someone else, it's like you going through a developer relations guy, and then him going to the kernel guy, and it's just so far away. But when it's in-house, you can fix it the same week.
Yeah. For everybody else, is Serebras available in the cloud? Can we go,
can an average AI engineer use Serebras hardware without buying their own training tiles?
Yes. Our cloud offering has just become available now with the release of Serebras GPT.
They're just like cloud instances. There are different layers of the abstraction you can offer.
You can offer a bare metal instance, like an AWS instance, where you just get a prompt and it's up to you what you install on top, or you could get a much higher level in the thing,
like you could have a set of platform with a pre-installed operating system, some tooling,
like a platform as a service, like a SageMaker thing, or you could have,
you know, this is a classic like SaaS stack. You can have Salesforce and Dropbox and stuff like that.
Serebras, the layer we're making our offering, which would cause Serebras AI model studio,
is a platform offering. So it's not bare metal, but it's not a GPT API thing either.
It's a environment designed so that specifically designed to help people train GPT style models.
So we have a set of model options, like all the GPT variants are supported.
You bring your own data, you know, you can bring any language, any number of sizes,
you define the sequence length, and you press train, and it will just autoscale across this at the end of the cluster, which is 16 CS2 systems. So think of it as a very
customized, optimized platform for the LLM trainer. If you are a generalist, let's say,
what's a good fit? What's not a good fit? If you're a generalist ML developer,
and you're doing all kinds of stuff, maybe one day you're working on image models,
one day you're working on recommenders, one day you're doing logistic regression,
and you just need a generalist like machine, you're better off just using your laptop,
or maybe a general like cloud based instance, because that requires like broad,
it has no specific pattern, and you want like horizontal support.
But if you are a, if your company is already, if you already have a very specific direction,
and you just want to train language models, like I need GPTJ trained in like a week,
and I can't find A100 instances or H100 instances in the cloud, I can't figure out how to like do this Megatron LLM code partitioning nonsense, and I'm time to figure that out, it's not my job.
Just train me my GPTJ model on this new dataset I have in my company, because my company has this
dataset, where's the fastest easiest way to do that? I think Syrige is actually the best way to do that. It is, it is you just supply your, you just supply your model or specify a pre-existing model, bring your data, press train, and it's zero crowd work to set up all the partitioning work,
and you get your weights back, like our first customer got a 1.3 billion model weights back within 24 hours. It's, you can't even procure A100 cloud instances in 24 hours, and some results in 24 hours, and that's for a 1.3 billion parameter model, and granted people will be like,
you know, that's not that big, that's not that impressive. The part that I am personally excited about is I know, like in my heart of hearts, that this, the same system that did this for 1.3 billion works for a 100 billion parameter model with no modifications. It's not like you have to be

a heroic engineer and like putting all this glue code, it works because it's the same architecture, and I can't wait for the day someone comes in and's like, I hear you guys are pretty good, but can you do this 100 billion parameter model, like try it out, let's see if it works, you press enter, and it's actually going to converge. That will be the day that, that like, yeah, that's the day I'm waiting for. Yeah, if I had a guess that won't be far away.
We'll see. We'll see.
Yeah. So the complexity is much lower, as you described. What about the cost? So if I was going to train a 13 billion parameter model, and my choices are Cerebrus GPT cloud, or go see what I can get in AWS running on A100s, what does the cost difference look like?
I think the cost, assuming the software, if you x out the software cost, if you assume like, somehow the larger GPT models work out of the box, I think the cost is roughly comparable, depending on how you price it out. So we might have a slight edge there. It depends on, it depends on which AWS instance, whether you do the three-year reserve or the, if you do that like on demand, I think we're way cheaper. I think we're way cheaper.
And maybe that's the comparison, because in a way, I don't know if you know, right now, there's a huge like GPU cloud shortage. You can't get capacity anywhere. It's like COVID early days, but instead of toilet paper, it's GPUs. It's incredible.
But yeah, so like, I see developers literally on Twitter begging for like, where can I find instances? Or we ran out of capacity at this server, that server.
So in a way, we are on demand. We are available right now.
Because it's a platform, right? I just go and I can use it. You don't need to worry about the underlying. And that's the benefit of platform as a service versus infrastructure as a service, is I'm not managing the individual servers and even trying to, because I've, I used to work in cloud engineering, even just connecting those multiple servers together is a total pain.
It's completely a server pain. Imagine the server's not even yours. You can't just go in there and plug in an infant a band for you. It's like, Amazon, please make sure this is working, right?
So we are on, we like, yeah, if you compare us to Nvidia, like on demand service in the cloud, we are cheaper. The turnaround is faster. It's immediately available. And the models are proven to converge, because otherwise we wouldn't have built these models and put them on a huggy face.
Exciting times. Maybe we can step back and just talk about, you know, general AI things, because I think you're an interesting mind at this base. You're very active on Twitter, especially these days. So I think consensus is starting to realize how big of a market AI is going to be. And there is still kind of question out there as to where value will flow across this ecosystem. There's kind of the open AIs of the world that are building these kind of like leading edge foundation models. And then there's a lot of kind of, let's say private companies that are going to deploy really focused like point and use cases are basically going to be building on top of and deploying the foundation models. And then there's the hardware vendors that are making all of it possible and supplying the actual compute. How do you see that landscape kind of today and end over the next, let's say, three to five years?
I think number one, the short sped is the demand for compute is going to be just continuous and practically unlimited. You know, people talk about, you know, training models are cost millions of dollars. Let's say cutting edge stuff at opening IRNVIDIA, Microsoft costs $100 million. It doesn't make sense. It costs a billion dollars because we've never built a billion dollar projects.

That would have to be compared to be like the Manhattan project or something or like a particle collider or like these government research projects. And this is not a good, this is not a correct line of thought because you're comparing research projects with commercial projects. With commercial projects like LLMs, there is no upper limit to how much it can consume in terms of cost. It may sound absurd to say you spent a billion or $10 billion training a language model. But if you spend $10 billion to train a language model and it can generate $100 billion of revenue and it's got, you know, better than 10% margin, you are fine. You're going to do it all the time and there's no limit. And all that money will flow to like compute and, you know, down to silicon providers like us, down to TSMC, down to ASML. Like that whole supply chain will absorb so much of this. The key question then becomes, can the models themselves like just generate revenue at scale? And like from your abuse of the products from my use, it seems like that there is no limit. It really is just no limit. I've never had a product just, you know, people call it the iPhone moment and all these analogies. And I feel like all these analogies are not enough because every product up to this point has only performed exactly to the specification of the product as intended. Like when the iPhone came out, it said it could do 10 things and it did those 10 things a few well, a few less well. And that's it. It's not like you picked it up one on the 11th day of owning an iPhone and it's like, all of a sudden it does this new thing you never thought it could do. That's never happened in tech because all the things we built in tech, we specify what it should do and then we engineer it and then we ship it up to this point. GPT and these LLMs is the first product that is entirely of a different category in the sense that we didn't really have clear specifications up front. We used a general method, we got to it and the product has a open-ended specification or set of things it can do such that day one, day week one, month one of release, we haven't even used 10% of the product yet. It's the only product you can wake up to the fourth week of using it and discover it had a thing that it can do that wasn't even intended. Like it's entirely open-ended. I don't even know how to convey that excitement. Yeah, it is remarkable. Like what I was going to say when you were saying, it seems bigger than the iPhone moment because you can wake up and discover new features and it's like, well, that's kind of like the App Store moment where I wake up and there's a new app I can download like Uber who thought when you bought an iPhone that it was going to be your taxi service. But what you just said at the end, which I think is the key difference, you had to wait for a developer to build Uber and an ecosystem to form around it. This is a native thing where I just have a new idea that I ask the model and I discover it can do it or it can't do it because it can't do everything. It for sure can't do everything, but it codes for me now. We're doing our monthly reporting and I'm automating it with code that I wouldn't have had the time to write if I didn't have chat GPT. Yes, exactly. The point is the feature is latent, buried inside of the thing and we don't even know what's there yet. We ship this thing, we don't even know what it can do yet. It's like only the first invention that is bigger than our intention, which has never happened before. And also, what is the difference between this versus like existing software can do A, B, and C? Most of the features of GPT style stuff weren't intended. They were side effects of training a large model. Translation is a side effect. Recommendation is a side effect. I just moved from New York to San Francisco. I like these restaurants in New York. Which restaurants do you recommend in SF? And it gave me a list and they were like, I follow the scenes. I'm like, this is the right list. Did OpenAI spend one minute on adding

recommendation and restaurant features and databases into GPT? No, they didn't spend half a second thinking about this. It comes for free. It's a side effect of general capability. So general being a key word, because I think that is the strength, right? Because the model is so generalized that you can do many different things. And the creator is not thinking about all of those different things. Is this an AGI moment? What do you think about that? And should we all be scared like Elon's been scared for a long time and very vocal about it? And there's the open letter going out that he and 20,000 other people signed. Where do you fall there? I'm optimistic by nature. So the bias needs to be up there in the foreground. I don't think there's any reason to be scared right now. The language models today have no intention. They the only intention they have is to satisfy the mild amount of reinforcement learning they experienced just before coming out. And those are just humans saying, I prefer this style of language over this style of language. So the current track, there's no reason to be scared. But I am a little bit, I'm just trying to consider that I'm trying to play out the consequences of what this means, right? One way to think of it like, are we going to hit a plateau or not? If we hit a plateau, fine, then nothing scary will happen. Like this will stop working. This will stop getting better. It will only get better asymptotically perhaps. And the world will still roughly be the same. But from the scaling laws, empirical finding, we know it's not going to plateau. So it will keep getting better non asymptotically. So the consequence of that is there will be like, it's getting a lot of scale. And also the tooling now you see on Twitter are like, the AI is capable of writing code that is helpful to help itself. It's also capable of generating data that is unique enough to train itself with global loss. So now we have feedback loops and feedback loops are very scary, right? It's the definition of the singularity. And if you play it out, and there's no plateau, and you have feedback loops, it kind of stands to reason that at some point, and read maybe fairly quickly, they will reach abilities that are quite that will make the current thing seem completely like a baby. The fact that there are no plateaus and their feedback loops just seems to suggest we're going to get incredible capability soon. It's not going to be a decade long slog. And there's no upper bound on the capability. So in theory, any task that we can that we call work should be especially in the computer realm. Sam Alton has a very practical definition of AGI that I think is helpful, which is it is an agent that can do any median human task that's on the other side of a computer. If you imagine you have a co-worker or a generalist, an executive assistant, or whatever kind of person on the other side of the world, and you just type it instructions, and you have maybe voice calls every now and then, an AGI should be a plug-in replacement for that. And I don't see why we can't get there with the current technique just off-scaling. The scaling laws is what gives the confidence for that. Without the scaling laws, you have no guarantee. You're like, maybe we'll hit a, like the Yanlan concept, like deep learning researchers, like driving a foggy highway, you could go on for a mile, you could hit a brick wall the next day. But with the scaling laws, you kind of have radar, and you're like, yeah, I see far into the distance. We're not going to hit a wall. That's what gives the optimism the rational basis for the optimism. And if that comes to be, it really, I think it's hard to convey the magnitude. Like it really is, you cannot predict anything past that point, just like if Apes could talk, and they're like, imagine if there's a smarter version of us, and it's like whatever, what would they do? Like they couldn't speculate the end result, which is, you know, we're sending spaceships

and things, and they are completely irrelevant in that picture. So I think we kind of have to take that branch of outcome seriously. But at the same time, we can't make any predictions about what will happen if that branch happens. If it doesn't happen, it would violate certain things we know today. So I find we're in this very, we're in a conundrum where it's like the future is likely to be very different, and we can't know anything about it at the same time, which is kind of scary and exciting at the same time.
It is. Yeah, it's pretty remarkable. I think, like when I think of it, and Sam's definition, and everybody has a different definition of AGI, which is what makes it tricky. And it's going to be hard to ever settle on one. But I think that bar he said, it seems closer and closer, especially when you take the capabilities of GPT-4, for example, and you give it plugins, which I think are a great idea where it's now this is extendable, and the model can write codes, maybe it can write its own plugins at some point. I think when you get to like, and I'm an optimist by nature, for sure, similar to you. So I'm not really scared. I'm 100,000 X more excited than I am scared. When you get to this like runaway AI, where it's like, we have no control over what the machines are doing. I think for me, that cutoff is right now, you can just unplug the machines. But when the machines can provision their own physical resources, and they make that jump from the model on the other side of the keyboard to the model operating in the physical world, and you know, I'm not just running a simulation, the model is actually moving physical space. That's when you can see, you know, they're they're standing up their own a 100 clusters, you don't you don't even need to make a decision.
Yes, exactly. I think that's exactly my direction as well, which is there's no risk.
So long as AI is very, very firmly in the world of electrons rather than than Adams.
If our world were such that critical manufacturing infrastructure and like, I don't know, defense, all that stuff is like literally one click away from open AI, then yes, it's very risky.
But you know, an ICBM is air gapped. Our tractors are not smart. Like the whole physical world is still very, very, very like mostly air gapped from what's in, you know, in a developer interface.
So given that, I guess the future wasn't like technology hasn't been that fully penetrated, I think I think there's reasonable. There's no reason to be super scared. I think if AI can make recursive improvement, self-improving improvement in the world of Adams, then we should 100% be scared. Like if they can, if they are, if there's a fully AI controlled factory, if they have their energy source, their own materials, and they can just experiment like using digital twins and it's actually, yeah, we're not going to compete with that. That's a different reality, but it's not at all the reality we have. So maybe the most practical, I was thinking about this, maybe the most practical security measure we can take, you know, AI safety, if we want to take practical steps, maybe the most practical thing we can do is just to air gap critical infrastructure so that AI can't get to it. Like the Eli Zierkowski, like hypothetical examples, like AI would send a packet of instructions to a DNA synthesizer that would then generate like basically killer bacteria. I think we should basically find infrastructure like that, whether it's food, energy, bio, and make sure that it is not like accessible to a chat box. Yeah. And when Tesla's humanoid robot knocks on the door, don't let it in. Oh, definitely not. EMP, home safety for everyone. So maybe a fun question to end on. I think, you know, what's so exciting is that there's new use cases just like we talked about that are coming out every day that you didn't expect. I just had a paper literally put on my

desk by another ARC analyst, Nick Groose, this morning titled Generative Agents Interactive Simularka of Human Behavior. And this paper was put out last week and researchers from Stanford and Google created like a mini game like The Sims and 25 autonomous kind of human-like agents
that were all based on an LLM and just observed them for two days and saw what happened. And they
gave like one prompt to kind of start this ecosystem off. And every agent had a different role in this little sim city, like you run the coffee shop, you're a small business owner, the small prefilled context of a world they live in. And the prompt was, you know, the coffee shop owner is going to host a Valentine's Day party. And then they see these agents living a society. They start sending out invitations. They decide who's their date going to be to the to the party. There's a one agent has feelings for another agent. And they just reflect on this. And then they also analyze it. And they determine, you know, based on comparison to real observe social behavior, that this is actually, you know, almost indistinguishable from true, you know,
the way human society would operate over a similar period. And then you can do all kinds of interesting things like that, like run psychological studies on a simulated human society or studies of all kinds. And the interesting is the inference cost for this was pretty high. It cost them, you know, a few thousand dollars to run this just for two days, because you have 25 bots that are just, you know, talking back and forth to each other. But Nick was describing this to me and it just like blew my mind. So that's what I'm thinking about today.
Like what is what is your like AI use case that has you like thinking about at night keeping you awake? Yeah, this paper is just making the making the rounds on Twitter. It's so funny.
You know, it would be more funny if the objective inside the the objective inside the simulacrum was maximize paper clips. That would be that would be a little more walls.
What's keeping me at night? I think what's keeping me at night is the is this just the structural setup of are we really in a non like non asymptotic improvement situation with no limit and we're on and it's it's it's machines doing this. Like I can't figure out what that implies. Like there are like the first the first home of safety in that walk the earth was that discontinuous moment for for apes. Like it just the world will never be the same again.
You normally don't bet the world will be very different than the past. But those bets are fails like 99.999 times. I'm just trying to figure out like, are we really is this for real?
That's kind of like, if it works, it kind of it will complete all of our other work. It really will like people working in energy space, like it will complete all that body of work better than you. First with your help, then your help is just slowing them down, right? Because
so how to take that seriously is is the meta question more practically, I guess,
I think opening eyes launch with plugins is incredible. It's like the most just agile,
on point, perfect go to market step two, since the app store, right? Because app store was like the second punch after the iPhone. And it's just literally it's kind of like
exactly a mirror of that. It's the second generation of chat GPT. And here are plugins which are like app stores and and it addresses all the core weaknesses of GPT. I've always been a huge fan of Wolfram Alpha, like math them like computational knowledge. I just love the idea.
I just love that you can, that you can actually ask Wolfram Alpha, you know, what is the position of Venus relative to Saturn? And it can compute that answer originally. It's just it sends

shivers down my spine, but it never had a perfect customer in the sense. I think it probably had certain certain industry sound found use with it, but it wasn't a mainstream like not everyone is using it from Alpha. Me and AP chemistry. That was the consumer.
And now maybe like Wolfram Alpha has the perfect like customer with chat GPT. Like this is a machine sibling that really appreciates it for what it is. The fact that the plugin architecture exists, it's just launched like a month. So we haven't even seen anything yet. We haven't had our flappy bird moment yet, you know. But yes, the world is already impossible to keep up with. I think with stuff like this, it's just every day will wake up and it will be a different world.
And if that's not the singularity, I don't know what is.
Well, I can't think of a better way to end it than on that note. It is it is pretty exciting.
I've probably said that 10 times in this podcast, but it really is. So thank you so much for coming back on the show, James. Hopefully we'll get to connect again soon. It's been great catching up, but really appreciate it. Frank, it's been awesome. Great to be on the podcast. And yeah, can't wait to see where this takes us.