/mlst

"You don't fine-tune your way to AGI" - Here's why. [Eiso Kant]

March 23, 2026



Machine Learning Street Talk

Podcast Transcript

Revision: 114bfb23

Topic headings include retention indicators: ● above average, ● average, ● below average (vs. similar YouTube videos)

Interested in licensing ReScript for your podcast? Get in touch with Tim Scarfe.

Contents

Overview 2

Introduction and Guest Background 3

Poolside AI's Vision and Three-Step Plan 4

Foundation Models vs. Enterprise Customization 6

The Missing Scaling Axis: Reinforcement Learning 8

Reinforcement Learning from Code Execution Feedback 10

Model Economics and Experimental Optimization 13

Enterprise Deployment Strategy and Market Focus 15

DeepSeek, Distributed Training, and Hardware Architecture 17

Emergent Reasoning and Chain-of-Thought Scaling 19

AI-Assisted Software Development Today 23

Architecture Innovation and Model Interpretability 28

Karpathy's Software 2.0 and the Future of Code 36

AWS Partnership, Enterprise Security, and Closing 41

References 47

Overview

Eiso Kant is the co-founder and CTO of Poolside AI, one of roughly seven companies worldwide with the technical muscle to build frontier foundation models from scratch. He sat down with Tim to explain why Poolside deliberately rejected the prevailing wisdom of "just scale up the next GPT" and instead bet the company on a thesis most labs were ignoring: that reinforcement learning from code execution feedback is the missing scaling axis.

The argument is straightforward and, once you hear it, hard to unsee. Next-token prediction is imitation learning. Reinforcement learning is trial-and-error learning. Poolside maintains close to a million fully containerized code repositories – each with its own test suite – as a massive, diverse RL environment. The model writes code, executes it, gets deterministic feedback, and learns. This is why software turns out to be the ideal domain for RL scaling: it is deterministic enough to provide clear reward signals, but diverse enough to avoid model collapse.

Along the way, Eiso and Tim get into the weeds on frontier lab operations (over 4000 experimental runs per month), why Chinchilla optimality breaks down once you account for inference cost, what DeepSeek V3 tells us about the second-generation AI company playbook, and why the R1 zero-shot reasoning result should have been the real headline rather than the dollar figure. There is a candid exchange on whether the software development lifecycle itself will progressively collapse into the model, what Chris Olah interpretability work means for alignment, and whether Karpathy was right about Software 2.0 – with some important caveats Eiso has developed after a decade of building AI for code.

The conversation closes with the practical side: Poolside's first-party AWS partnership, managed enterprise installs in under 40 minutes, and why bringing the model to the data rather than the other way around remains the winning go-to-market in 2025.

Introduction and Guest Background

Eiso Kant 00:00:00

You will never hear me argue against scale. The narrative at the time in the world was we're just gonna scale up, you know, the next GPT model, make it 10 x larger. This was such a strong narrative that frankly many people at the time in the world, you know, believed this. We didn't. There was a missing axis of scaling that wasn't being discussed, and it's frankly why we started this company. It was the axis of scaling for the use of reinforcement learning. Scaling of next token prediction is the equivalent of imitation learning. Scaling of reinforcement learning is equivalent of trial and error learning. The reason we built this company is because we saw a future that I personally think now is 18 to 36 months away where human level intelligence across the vast majority of knowledge work achieved. You don't get to do that unless you build from the ground up. You don't fine tune your way to AGI.

Tim Scarfe 00:00:49

I've just had the most amazing conversation with Eiso Kant. He is the co-founder and CTO of Poolside AI. Now, are building frontier language models. They're 1 of about 7 or 8 companies in the world who have the technical chops to build foundation models from scratch. They have a really cool solution for doing generative AI coding. Honestly, it's now possible to write software about 10 times faster than we did before. Now, what these guys have done is they use reinforcement learning from code execution feedback, which means they're going 1 step further in the stack to align the language models that they build to the code and the software that you are writing. There's so much low hanging fruit in this space at the moment. When are we gonna have code solutions that can watch your screen, that are multimodal, that help you collaborate better with developers? Eiso has a very interesting story to tell about all of this. We also had some pretty cool Galaxy Brain conversations about how to train foundation models, about test time computation and thinking and reasoning. I think there's a lot for, you know, quite a few people to get their teeth into in this conversation. MLST is sponsored by Tufa AI Labs. Now, they are the DeepSeek based in Switzerland. They have an amazing team. You've seen many of the folks on the team. They acquired Minds AI, of course. They did a lot of great work on ARC. They're now working on o1-style models and reasoning and thinking and test time computation. The reason you wanna work for them is you get loads of autonomy, you get visibility, you can publish your research, and also they are hiring as well as ML engineers. They're hiring a chief scientist. They really, really want to find the best possible person for this role, and they're prepared to pay top dollar as as a

joining bonus. So if you're interested in working for them as an ML engineer or their chief scientist, get in touch with Benjamin Crouzier, go to tufalabs.ai, and see what happens. Eiso, it's it's an honor to have you on MLST. Thank you so much for joining us today.[1]

Eiso Kant 00:02:42

No. Thank you so much for having me. I appreciate it.

Tim Scarfe 00:02:44

Can you tell us a little bit about yourself in Poolside?

Eiso Kant 00:02:47

Personally, I'm I'm a computer geek. Started

Poolside AI's Vision and Three-Step Plan

Eiso Kant 00:02:50

programming when I was quite young. In 2016, found myself building what I believe to be the world's first company that focused on making AI capable of writing code. So I met my co founder as well actually. It's a longer story. And in April 23 we founded Poolside. Poolside was really founded on our view that the world was going to achieve human level intelligence and AI. And we took our own point of view on how to get there, and that's been the fundamental start of us, since now almost 2 years.

Tim Scarfe 00:03:23

And your co founder is Jason?

Eiso Kant 00:03:24

Correct. Yeah. Jason and I met, because in 2017 he was the CTO of GitHub. And I'm not sure if I've ever said this publicly, or at least not on camera, is he actually made an acquisition offer for that company I was building called Sourced. And we, back in the day, had the world's first models, they were able to do code completion and things like this working, And I turned down the acquisition offer, but nonetheless we became really good friends.

[1] Tufa AI Labs — website

Tim Scarfe 00:03:48

So tell us a little bit more about Poolside. What's the main goal? Poolside's

Eiso Kant 00:03:52

main goal comes down to wanting to build a world where we have human level intelligence that we can scale up on compute. We think it is essentially going to have 2 ways of having impact. 1 is the more that we can make capable intelligence scalable, we can start driving the cost of goods and services down to 0. On the other hand, there's this entire frontier of technology and scientific progress that's ahead of us, and by definition it's infinite. We will always continue to find more, and so to be able to pull that closer in time has always been our mission. But we took a slightly different path than others. We took the path on focusing on making AI incredibly capable of building software. And we laid out this 3 step plan kind of on our website on day 0. It's still there in the bottom of the footer if you click on vision. And it said, make AI capable to assist developers in building software. Step 1. Step 2, allow anyone in the world to build software. And step 3, generalize it to all other fields and domains.

Tim Scarfe 00:04:51

So there's a kind of winner take takes all dynamics in in the space at the moment. There are amazing frontier models out there. I mean, I've been playing with, you know, Sonnet 3.7 Thinking and and whatnot. And there's always the question of differentiation. I mean, Anthropic, they released this Claude CLI thing, and you can just stick it on your repo and, my god, it's really, really good. So how can you differentiate on top of that?

Eiso Kant 00:05:16

So I think all of us at the frontier are constantly competing with each other for model capabilities. And I think in the fullness of time on capabilities like software development, we might all end up even in the same place. But if you look at right now where the world is at, it's a very small number of companies that are actually competing there. I would say that we have kind of the old guard like Google. We have kind of the first generation of AI companies, OpenAI and Anthropic. And you kind of have the second generation: xAI, Poolside, Mistral. We all were founded around this you know, April '23 date. And all of us, I think, are in that same race. Now, we've decided to not focus on making our models generally available for every possible use case, but really make them available for software development. And this allows us a set of liberties in terms of where we focus on and areas we don't focus on. But don't be mistaken, like, the work that we do to build really capable foundation models still lends itself to building really, really capable models across the board. Because software development is not about writing code. You need to understand the world. You need to be able to do multi step complex reasoning. You need to be able to plan across long objectives. And so I was super excited what Anthropic did with Claude 3.7. Think it's an amazing model, the Sonnet model. And of course it's for us to make sure that we didn't surpass that. And so we're constantly in this race with each other, but we take certain views on our research and our approaches that I think will allow us to accelerate over time faster towards those goals.

Tim Scarfe 00:06:48

So I'm trying to understand this

Foundation Models vs. Enterprise Customization

Tim Scarfe 00:06:50

because there seems to be a dichotomy between having really general foundation models that can do lots and lots of things versus the story from many folks in the space who are saying we need customization, need personalization, we need on prem deployments and so on. How do those 2 worlds come together?

Eiso Kant 00:07:04

So I think it's a spectrum. I think absolutely in the first part of your training of your models, you want to embed as much diversity and knowledge of the world as possible. Software development is not about writing code, it's about being able to interact with the real world and turn that into a digital form. So, having that general part down is incredibly critical. I might not care as much about how humorous my model is and how good it can handle nuances of comedy, but I absolutely care about the knowledge that it has in many different domains. What ends up happening though is that all of us have a fixed parameter space. At the end of the day there's a cost of inference, so there's only so many parameters that I can load up and then actually run. And by having a fixed parameter space it means that you have to choose what you wanna do with those parameters. And so we try to shift the distribution of our model capabilities very much towards software development in terms of the capabilities that they have, but also meaning that we're willing to trade off, you know, not maybe being as good as writing a creative bedtime story or writing comedy or areas that probably more you would find in a consumer AI from other people. But you mentioned a second part, is this notion of customization and where do you deploy. I think this really comes down to how do you view where models are gonna be in the coming years? And so I think all of us at the frontier, it's our responsibility to build the world's most capable models that can interact across all of science and technology and knowledge work. Even if we focus on their abilities to build software, we still care about all of that. And over time in our step 3 we want to branch off to opening up to those other areas as well. But if you look at what's gonna happen in the future, there's a big question. There's a question of do we have some all powerful model that is static? Right? It's it's 1 model and we all use it to do economically valuable work. Or do we have all powerful models that are able to actually become versions of themselves deployed in environments learning from the data in that environment. So the question is, you know, do we have a software developer that can write all software or do we have 1 that's deployed inside a banking environment and has truly access to all the information and learned over time from it? So it's a little bit, do we anthropomorphize it? Is it going to be like a human? Right? I can be a very capable software developer but you deploy me at, you know, a bank, all of a sudden I'm going to have to learn everything over time from that bank and that embedded knowledge? Or am I something generalized that gets applied to it? I think the honest answer is that we don't know yet in this space. What I do know is today, when the models are not yet at human level capabilities and are not even superhuman level capabilities are there yet, it's very valuable giving the model access to as much data and as much context and as much ability to learn in an environment. And so we just look for the shortest path towards doing that. And the shortest path with enterprises towards doing

that is by willing to deploy the model, the context intelligence layer, and the applications behind their firewall close to the data. It's a tactical in time decision. I think over time the form factor of that might change, but it's something that we've seen resonated really well with the customers we have.

Tim Scarfe 00:10:16

It's because I'm trying to tease out your view on scaling, so that there are many folks who think we should scale the models up. GPT 4.5 just came

The Missing Scaling Axis: Reinforcement Learning

Tim Scarfe 00:10:25

out. It's quite interesting that there was, you know, Gwen and Kapathi, you know, who were on Twitter, and they were basically saying, well, you know, high vibes people, that means like smart people, they can see that this is this is a step up and it's doing well in in very nuanced things, but the benchmarks aren't capturing it, we need better benchmarks. But undeniably at the moment, there is a gap in capability, right? So we need to have customization and like, you know, thinking on-site, surface contact with domain verticals in order to do to do well. But I think you're saying that you can imagine a world in the future where we could bring all of this data back into a huge foundation model and and it would work just as well.

Eiso Kant 00:11:00

You will never hear me argue against scale. Scaling of compute and scaling of data is critical for us to close the gap between where models are today and where we believe they can be. Human level intelligence and even beyond. But that doesn't necessarily mean that the axis of scaling today are the same axis of scaling that they were what people thought they were 2 years ago. Right? So when we started this company, the narrative at the time in the world was we're just going to scale up, you know, the next GPT model, make it 10x larger and provide it with more web data and we're going to have this AGI like moment, this human level intelligence. And this was such a strong narrative that frankly many people at the time in the world believed this. We didn't. And it's not because we don't agree with skill. I think skill massively matters. But our view was that there was a missing axis of scaling that wasn't being discussed, and it's frankly why we started this company. And it was the axis of scaling through the use of reinforcement learning. You mentioned Karpathy, I liked how we said it the other day, like scaling of next token prediction is the equiv-

alent of imitation learning, scaling of reinforcement learning is the equivalent of trial and error learning. And while there's probably some nuance to that, I think it is a right way of thinking about it.

Tim Scarfe 00:12:12

Yeah. I read this amazing book by Max Bennett called A Brief History of Intelligence. And he was basically saying when you look at the animal kingdom and humans in particular, you see this axis of simulation. Right? So it's the ability to imagine things or, you know, imagine experience which you haven't directly had access to. Right? And that just creates this this explosion. And language, of course, is is an even more sophisticated invention because it allows you to sort of memetically share those simulations that you didn't actually have with with other people. So with reinforcement learning you can actually try things and you can accumulate knowledge without needing to have direct physical experience.

Eiso Kant 00:12:54

So I think it depends on where you apply reinforcement learning here, right? So I tend to agree with a lot what you said, which is that at the end of the day what we do in thought, and I think it differs by the way, the personal experience to personal experience. We spoke about this earlier. My thought is entirely language based. It's a constant internal monologue that is so language based that there's no visual representations or abstract concepts. So for me I feel quite akin to language models, to put it kind of funnily set, because I see how they think and how they reason, can relate to it. But the reason I mentioned this to your point is that language is a way, I don't think it's the only way, but it's a way that we can explore different possible chains of thought, different possible thinkings. And I agree, a lot of how my mind works and I think how many is is that you're looking at an objective and you're thinking through the different possible chains of thinking that can get you to that objective. If that's about writing a piece of code or if that's about something far more long, you know, arranged. And there are several things that we do to ensure that that objective is correct. 1 is that we try to keep it consistent with what we know are representations of knowledge that we build upon. So, if I am, you know, reasoning or thinking through a math problem, I'm constantly consistency checking against the knowledge representations that I've previously learned, right? The axioms that have to be true in math or physics or any other domain. But then there are certain things that no matter how much I try to keep true against, I actually need to do the work. I need to get the real world feedback. Call it maybe slightly flawed example here, I think it's a useful 1, is if I want to learn Go or chess. I can read a whole bunch of textbooks. I can play

out chess games in my head to some extent, but at the end of the day I am still a snake eating itself in terms of my own synthetic data. Right? Like, and so coming into an environment where I play against someone else, computer or person, doesn't really matter, and I learn from my mistakes. Hey, my reasoning chains that I explored led to me losing this game. That's actually where I think it's really, really valuable to have some form of external feedback. And that external feedback in the case of chess or Go is deterministic. And sometimes that external feedback can be human feedback, but it's very hard to scale. And it's also not necessarily always the right type of feedback. And so that's where I think reinforcement learning can live on that boundary. Where when we have the ability to form a reward that pushes us more correct, raised in the right direction of correctness, we can then improve the next set of thinking and thoughts that

Reinforcement Learning from Code Execution Feedback

Eiso Kant 00:15:40

are those next thinking chains that you're talking about.

Tim Scarfe 00:15:42

Yeah, there was a great Nature paper talking about model collapse with Ilya Shumailov. We were dropping the interview with Ilya actually next week. But in a way, it reminds me of some of these AGI doom type discussions, right? Because when you talk about omniscience, I think that's not a scientific discussion. I think that in the real world, we need to push molecules around. It's the same thing with software, right? The reason why these Gen AI software coding things are so powerful is because they can actually test the software that they generate. That's really, really important, getting that signal from the real world.

Eiso Kant 00:16:14

So we're very known for our work on learning from code execution feedback. I started this work when we were building our first company in the space in 2016. It's kind of a big part of what we've spoken about that Poolside uses, and this is really the notion that if you have a very large diverse environment, which in our case we do, we have close to 1000000 repositories that are fully containerized with their test suite and many millions or tens of millions of re-

visions, that we can say, hey, at this commit hash in this repository I want to change this code and then I want to execute it and see what comes back. If that's either running tests against it or just a compiler or interpreter or even synthetic tests. And what this allows us to do is it allows us to have a very, very large environment because 1000000 repositories represent huge diversity of types of problems if that's in cryptography or a web app or a core database kernel. And it allows us to then design tasks for the models to do where they can explore possible solutions and the thinking that leads to those solutions, and then learn from when they're right or when they're wrong. Or at least I always try to be careful saying right or wrong, more correct and less wrong. Right? Because that's essentially what reinforcement learning is. You're trying to push the model in a direction that the next time around when you're sampling your thoughts and you're sampling your solutions, you're slightly better. You do that enough times in training, you can start getting to a place where you get very good. But if you do this on a very narrow task, you get this notion of model collapse, right, or overfitting. You get to a place where it's like, okay, the model can only do this now. And then it's no longer a useful generalized intelligence. Right? So coding kind of sits in that spectrum of like, it's deterministic enough, but has enough diversity that even when you're overfitting a little bit towards it, you're still making it very much a generalized intelligence. You're not making it a task specific thing that can only play the game of Go.

Tim Scarfe 00:18:09

I'm fascinated by this idea of a diverse set of possible minds or different intelligences. I think that you can create intelligences through a variety of representations and there's many degrees of freedom. You made an interesting comment actually that you're not very visual, you know, you you think in this analytical language space, I'm very visual, I can imagine sounds and so on in my mind. And the way we we write software is actually very diverse, you know, like design pattern books and so on. Those are different analogies, different abstractions, you know, even Einstein, when he thought about relativity, was thinking about ripples and waves and so on. And with software, you know, there's there's the syntax, there's the actual sort of like, you know, how we write the code, there's the semantics, what what it means, and there's the behavior space, how we actually test it. And you're kind of talking to a form of AI where we're learning like a hierarchy of representations and it can kind of flit between them dynamically.

Eiso Kant 00:19:02

It comes down to how we view these models, Right? And I want to really caveat this with that we don't have a good ground of scientific theory yet or really robust interpretability that what I'm about to say is anything but my best guesstimate or opinion at this point. But I think probably most people in the space will agree on the following, is that what's happening in these models is that we're learning extremely high dimensionality representations. And some of those representations that we're learning represent, you know, the ability to use language. So these are representations that are massively interconnected with everything else. Some of the representations that we're learning are very specific pieces of knowledge. Right? So we spoke we had a bit of conversation about this earlier, but if I take a piece of knowledge, like when FDR was born, that sits in a high dimensionality space. Probably sits close to other things related to US presidents and such, or if you go back to the early days of machine learning, we think word to vec and, like, stuff like this, like, I think those are still useful mental models to have. But back then we were talking about embeddings and representations that represented words or or bag of words or knowledge. Nowadays, what we have with the the type of models that we've been able to build is that we have representations of things that are far more generalized and far useful. The ability to use language, the ability to start doing reasoning. And and I mentioned this because I think in the first wave of how we've trained these models in the last years, of just scaling up next token prediction on more and more data on larger models, we were improving the representations that represented the most overly represented things in the data. Language, knowledge, But we weren't yet able to start really improving the representations of complex reasoning, of multi step processes, of the things that are required for building complex software, the things that are required for, like, figuring out new scientific breakthroughs and theories. And now we have access with reinforcement learning to really improve those. But at the end of the day, if we had an infinite amount of data in the world that represented all of our thought process and all of the feedback we got, that it wouldn't really matter what we use. We could just learn it with Next Token Prediction. Right? We wouldn't necessarily need to use reinforcement learning. These are all just ways for us to improve the data and hence improve the intelligence, and we're trying to do that in the most compute efficient manner. So my team gets very tired of me saying this. They always say all the work we do is either improving compute efficiency on training or inference or it's improving the data and hence the intelligence of the model. And everything you do you can put in 1 of those buckets. And I know it's an oversimplification and there's always a little corner case here and there, but when we come up with a crazy new architecture for linear attention, which is something we've put a lot of work into, to me that's just improving compute efficiency of inference. Right? If we figure

out a really amazing way to scale up reinforcement learning, that's actually just improving the data. And so, to me, they're just the 2 facets of model building.[2]

Tim Scarfe 00:22:06

Yeah. I think the the economics of of models is is really important actually because even now when when OpenAI finally released o 3, there's a new version of the ARC challenge that that came out, and they were spending $2,500 per task,

Model Economics and Experimental Optimization

Tim Scarfe 00:22:20

but they could solve it, they could get superhuman performance. So now it's simply a matter of computation. But there's still a Pareto frontier, right, you know, we've like the Gemini model, it's it's very, very cheap. You can just sample it, you know, maybe 50,000 times and and you can still get the answer. So so then, we get to this definition of of intelligence and AGI. We've been talking about that a lot. Francois Cholet said it's basically your your reasoning or your knowledge acquisition efficiency. So, know, how quickly do you take new points in experience base and turn them into new skill programs. And you must be wrestling this Pareto this Pareto frontier, right? So, you're you're figuring out what's the appropriate size of model, what's the appropriate architecture, and what's the trade off between how fat the model is and and how much knowledge acquisition and reasoning I do in the situation.[3]

Eiso Kant 00:23:01

I think at the end of the day, all of us at the frontier right now are taking advantage of as much computational resources as we can get. And I think it can't be ignored. Like, I think if you are in the race for frontier model capabilities, your amount of computational resources that you can direct towards training is absolutely critical. But the wrestling part comes to where do you apply it? Do you apply it to make the model larger in a parameter space? Do you apply it to make massive amounts of synthetic data generation? Do you apply it to the scaling of reinforcement learning and sampling more? And all of this is essentially an equation that has an optimal, right, for every single 1 of these things. And the way that we operate, and frankly I would say most frontier labs operate, is that we try to run experimentation in each of these areas and several others to find that optimal. So to give you a bit of a sense, our team in

[2] Efficient Estimation of Word Representations in Vector Space (Word2Vec) — paper
[3] On the Measure of Intelligence — paper

January ran over 4,000 experimental runs. These happened across architecture, across data ablations, weight mixes, reinforcement learning, number of samples, all of these different variables. And what we're fundamentally trying to do with all of this experimentation is trying to understand where, what is that optimal balance between those things. And you said something really important, at the end of the day cost matters. So the overarching kind of objective function, all of this is the maximum amount of intelligence that you can create within a certain amount of time and budget that you can then serve for a certain price to the end consumer. And and those the training and inference part are are really critical. And so in our domain, because we focus on software development capabilities, we're in a quite valuable domain. It's economically valuable. I think it is a lot tougher if you're trying to serve both that domain and you're trying to serve the free users who want to write bedtime stories. Right? Because they have different economically values associated with them.

Tim Scarfe 00:24:54

From an architectural complexity and customization point of view, you know, the enterprise, they wanna design their own architecture. They they wanna have clear security boundaries. So, you know, the software engineers in the finance department, you got the soft software engineers in in the legal department, and and they want to create their own trade off along those boundaries that that we're just talking about. So does that does that make it more complex that rather than designing 1 recipe for everyone, you're you're doing a lot of, you know, bespoke stuff?

Eiso Kant 00:25:19

So I think there is the there's the building of the foundation model and then there's building of everything around it. Right? All of the software that allows you to then deploy it in different environments, if that is workstation, server on prem, if that's in the private cloud environments, VPC, that's pure cloud anyone can access. And so we made a decision early on from day 0 of the company to say we are going to do everything possible to become the trusted partners of enterprises. And it came from quite a simple analytical process. We said, well, where does the majority of economically valuable software development work in the world sit? And it sits in software development in enterprises. Right?

Enterprise Deployment Strategy and Market Focus

Eiso Kant 00:26:00

70% of all dollars in software development gets spent in enterprises. But these are also, like you said, very complex environments with lots of security concerns and boundaries. And so we took a, call it again a simple point of view, said, where do they want us to be? Where are we seeing that we can actually be that the customers ask for? And what we heard over and over again is bring the model to the data, not the data to the model. And so we decided to build accordingly. And so today we deploy our full stack model and applications all the way on top in these private environments That requires a lot of work, lot of engineering, a lot of work that we do to streamline this so that it can scale. I personally think in the fullness of time everyone will end up on the cloud. But today if you are tactically looking at, you know, the global 2,000 enterprises, a lot of them are still very happy that we're able to deploy in environments that other people aren't.

Tim Scarfe 00:26:53

So, you've gone for a kind of Tesla esque strategy in in a sense that you're controlling the entire stack. You're building I mean, you're 1 of not very many companies that have the skills and expertise to build the foundation models. But I guess the question is that there are so many folks that are just focusing down the stack, you could add so much value doing the reinforcement learning from code execution feedback, doing the whole kind of like, you know, architecture piece. Why did you decide to go the full hog and build the foundation models as well? I think it started where we

Eiso Kant 00:27:20

had this conversation today. Like, how do we start this conversation? We started around, like, what is Poolside looking to achieve? And the reason we built this company is because we saw a future that I personally think now is 18 to 36 months away, where human level intelligence across the vast majority of knowledge work is achieved. And if you hold that point of view 2 years ago when we started that we were And at the time the timelines were not that concrete for us. We would say 5 to 10 years, maybe 15. We knew that the world was gonna get to a place where we would be able to replicate our intelligence and even go beyond. And when that's the point of view you hold, the question is, well, what do you need to be 1 of the companies who can help bring that into the world? If you remember we had our own point of view on how to do so in research and execution. And you don't get to do that unless you build from the ground up. You don't fine tune your way to AGI. We put that on the website, on that same page we spoke about earlier that was there on on day 0. We had this list on there that's is essentially It was called strong beliefs weekly help in the face of empirical data on our research. And 1 of the things was you need to build your foundation models from the ground up to be able to get to these things. You can't fine tune your way to success. We said all data over time becomes synthetic. Reinforcement learning is key to be able to scale up capabilities. So it really came from that. Both what we wanted to achieve and what we believe mattered from a research perspective. And I think that has really played out in our favor so far. And I think it will be very unlikely that we see anybody in the world get to human level capabilities by post training the latest open source model. And I would even question, and I'm a huge open source advocate, my first company in the space I built fully open source, But I would even question if we are going to have, at some point, truly open source AGI, if there's gonna be room for that in the world, if we are continuing down a path where the capital required to build this is so incredibly large.

Tim Scarfe 00:29:24

How hard is it to build a foundation model? I mean, just to give you a couple of examples, you know, DeepSeek. What's cool about them is they've made a lot of their training methods and optimizations public. They've got some great papers out there, new sparse attention papers, really, really cool. And I'm I'm guessing that as you go up in scale, it gets harder and harder to train these models, but it's so difficult for people like me to know. I mean, is it just a software engineering challenge? You know, how how hard is

Eiso Kant 00:29:50

it? So I love to break that into kind of 2 parts. 1 is actually comment a little bit on DeepSeek because I think DeepSeek is a is a great example of the second generation companies. Us and and x AI and DeepSeek, we've taken a different approach. So DeepSeek, as far as I know right now, is about 200 people, right, researchers and engineers. They've got, you know, billion dollars plus of infrastructure there and they've got 2 years of incredible work already behind them, and we know it because they've been publishing their work. Right? So all of us in the field, known entity. We've been following the papers. And I have a lot of respect for what they've done. Because if you look at that last paper that they put

DeepSeek, Distributed Training, and Hardware Architecture

Eiso Kant 00:30:30

out, the 47 page technical report on V3, In this space we all know if you do all of that work you get a really good model. And so I will even, you know, say to controversy of some that like I don't think they stole data, I don't think they did anything nefarious, I think they actually just did great work and we've got a 2 year track record of research papers to actually follow to see that. Now there's questions that we need to have about do we want, you know, capable AI, you know, deployed in the world in the West where we might not share the same values or same, you know, principles as, you know, the CCP, you know, has, but that's a completely separate discussion. But the other notion is that process of 2 years of building, what what we have gone through, what XAI has gone through, what DeepSeek has gone through, it's yes, of course, the models get larger and the engineering gets more complex and the research, but at the same time, I think it's building up compounded advantages over time. You know, I think talent is is so critical in our space. We have this incredible team and if I look at what we have learned together as we've grown over the last 2 years, there's an immense amount of value in that. Now, of course, that needs to go hand in hand with actually improving your data constantly. Right? Like, month you can look back and you say your data on which you're training is better, it's cleaner, it's better representative of what you're trying to achieve. It's improving your actual distributed training stack, right? Either by making it more compute efficient, which is a big part of it, but also allowing it to work at certain skills that before it wouldn't work at. Right? This is all of the work that you've seen on the different types of parallelism that you get as you scale up into larger and larger clusters. There's work that you do that really is specific to how the chips underneath change. Right?

So we've got the Hopper series, but now if you look at what's coming out with the GB 3 hundred's, all of a sudden we have 72 chips with NVL connection. That changes the architecture that you wanna run if you're training on that. So people often take the view of like the architecture comes and then you map it to the hardware. It's actually the opposite. You look at the hardware and you determine what the optimal model architecture is for training and inference, so you're constantly going back. But over time these all just become compounded things. Our environment of code execution went from 1000 repos to 10,000 to a 100,000, now close to 1000000, and it's gonna keep growing. So, some of it's engineering, but some of it is also just the implicit knowledge and experience that you gain in your organization that builds these kind of modes for others to not be able to compete overnight with.

Tim Scarfe 00:33:01

Is it fair to say though, if every time you 10x your scale, what you already know isn't enough to get you there? You have to sort of not go back to the drawing board, but you have to spend a lot of capital trying a whole bunch of different things.

Eiso Kant 00:33:13

I think scale needs to be can use a little definition. Because a lot of people have talked about scale as just scaling up model size. Right? I mean, that's you mentioned GPT 4 and a half, which I think OpenAI says, know, 10x more compute, which probably means, you know, maybe 5 times larger and x amount of times of data or, like, whatever that combination is. Until, you know, recently the world assumed there were 2 scaling axes: Size of model, size of data. I think reinforcement learning is truly a third scaling axis right now. And of course that still is a proxy for data, don't get me wrong, but I think it's important to call out as a separate scaling axis. Because this changes the dynamic in terms of what skill means. Maybe you are not taking the model to, you know, 10 train parameters. I'm throwing out a number here. Maybe you're not taking a model that large anymore because you're finding that you can scale more efficiently by scaling up the, you know, reinforcement learning side. A good example of what we saw in the last couple of years was the LAMA models, where you saw at some point they went from, you know, 2,000,000,000,000 tokens to 15,000,000,000,000 tokens. By the way, all of us in this space have been doing this for some time because and that would be referred to at the time as overtraining because it was not chinchilla optimal. But chinchilla optimal never took into account that these models actually have an inference cost. So there are places where on the theoretical, yes, a certain model size with a certain amount of data with a certain reinforcement learning is the theoretical compute optimal way of training a model. But actu-

ally, if you have a constraint of running this in the real world, I need to serve this to a customer and it can cost me only so many dollars per million, you know, tokens to serve or a number of requests, then that constraint changes where I might spend my skill. And the reason I mention this is that I might say, okay, I might train it for much longer. But it doesn't actually necessarily introduce a lot more engineering complexity. Right? It might But if I'm trying to go 10x larger, then it introduces engineering complexity. But it also depends on what hardware I do it. If I'm scaling up like Elon on, I think it's 6 32 ks, you know, hopper, like H100, H200 clusters that he has interlinked together, then all of a sudden that's a very different way of scaling than if I take the equivalent flops on a 100 ks, you know, GB200s or 300s that are outcoming. So don't get me wrong, there's always next engineering. To us it doesn't really feel like 10x from 1 to another, but there are major changes that we make as we find having more access of scaling. Where I think we've been very well positioned is that 2 years ago we started already deeply the company around large language modelling and reinforcement learning. So this wasn't a new thing for us, so we've been building up incrementally over that. I think other companies all of a sudden had to bring this out from almost nowhere, I'm pretty sure that felt like a 10x engineering project.[4]

Tim Scarfe 00:36:04

Can you help the audience understand a little bit more about thinking? So, you know, the r 1 came out, and you you know, it's the same with Sonnet 3.7 and o 3 and Gemini Flash thinking and so on. And the the end user perspective is you you see these thinking tokens and the language model is kind of doing a form of self prompt augmentation. So there's, you know, there there's like the the the tier 0 of chain of thought and scratch pad, and now the things are prompting themselves. And 1 way of thinking about it is, you know, we train them with reinforcement learning and it's kind of like, it's imbuing this process in. But the interesting thing is that you can just take a 100,000 thinking

Emergent Reasoning and Chain-of-Thought Scaling

Tim Scarfe 00:36:40

traces, you can fine tune just a normal base language model, and then just through sheer, you know, dint of interpolation, you get a lot of the performance. So, you can buy performance, there's a kind of sigmoid relationship with the But what's really going on there? Is there something special about the reinforcement learning in of itself or should we just think about it as a form of data augmentation?

[4] Training Compute-Optimal Large Language Models (Chinchilla) — paper

Eiso Kant 00:37:01

I have opinions here. Some of them I think already backed up with papers that are out there and things. Others, not yet. And and we don't publish. So I'm saying this because I wanna reference things that are that are out there publicly to to back up some of the argumentation. Look, at the end of the day we are updating a model, right, based on, you know, a loss or or some, you know, function that we are applying there. And so, yes, in the in the truest of definition all of it is just data. Right? All of it is just data. Like say if you had infinite data, you had infinite reasoning traces for everything else, you could learn it through next token prediction and it would probably be an incredible model and reach human level capabilities, right? So But it is very clear that what you are doing today by taking a 100,000 reasoning traces versus the equivalent compute spend on reinforcement learning to get there, that reinforcement learning outperforms the SFT side. I think that it's just because there is more signal in terms of what you can provide, in terms of reward, than you can just by providing only like the sample of data. This is, I think, it's trade off, right? And again, it always comes down to like the data and the compute efficiency. And so I think if you had an extremely large amount of data, there is a path there. What I think is happening right now in the world where it's like, okay, we're seeing massive generalization from a 100,000 reasoning thinking examples, I don't buy it. I don't think that's actually what I'm seeing in the models. I think even current benchmarks are are able to show that, but it's quite often when you look at these things you see like, oh, it went up on this math benchmark a lot. And then the reasoning, you know, traces are all very specific and quite closely linked to like what's happening in that benchmark. So at some point, yeah, you might grok something a little bit more generalized, but you need a lot of data for it. So I think that the scalable ways of improving models are more around applying reinforcement learning where possible. That being said, there might be compute trade offs where at some point you say, oh, I want to use some of that SFT data to either bootstrap versus having it, you know, learn from scratch. I think there are places that can be done there. But you mentioned R1, the most exciting thing that I think was finally published in our space, because, you know, none of us in the West really publish anymore, was the 0 work. And the fact that you could see a model develop its thinking capabilities in coherent language without having been provided, you know, sufficient samples of what thinking looks like. This should be the thing that blows all of our minds. It shouldn't be the headline around the $5,000,000. The headline should have been models are able to develop human like thought in language that is leading to better outcomes in things that are objectively measurable like math and coding capabilities. That's the exciting part without having actually been, like, aligned towards it.

Tim Scarfe 00:40:14

Yeah. That paper blew my mind. I'm not sure if the 0 was it's not like AlphaGo 0 where it's, you know, there's no human seeding. Think there was still a little bit of human seeding, but it but it was mostly self play. But but it was incredible, right? Because it learned these emergent behaviors, like it would say wait, and it would do stop, and it would do reflection, it would do reasoning. And many of these seemed almost like there's there's there is a natural way of doing reasoning. It was very like human aligned.

Eiso Kant 00:40:37

Well, think, look, we can't forget that the basis of training data is still the Internet, right? It's still the web. And so when people go and say, oh, I just want to see it perfectly learned from nothing, it's like, well, you want all of evolution to happen overnight? Like, still have we're teaching these models based in our own image, in our own data, and what we have. Otherwise, frankly, also wouldn't be very useful. We want them to act in our environments. And so but the fact that before if you take a pre trained, you know, base model and you see the difference between that pre trained base model versus 1 that's had actually, you know, reinforcement learning applied for developing its thinking, you can see that the capability of those thoughts, like you said, the self reflection, all of these things, is improved. Right? And I think we often use the word emergence, but I think it's a spectrum and we see things, you know, improve, and all of a sudden we now have a lever that we can pull, we've had it for some time, where we can improve the thinking capabilities of these models. And by improving the thinking capabilities of the models, to our point earlier, there's a smaller space of solutions you have to explore to get to the correct thing. And the more we can make that space more and more correct in the areas we care about, mathematics, software development, scientific theory, all of these kind of areas, the more useful and valuable these models come. And by the way, humans are exactly the same way. Right? Like, I have this massive, you know, set of learning behind me that makes me useful in a software development environment, but if tomorrow you drop me into quantum physics, I'm probably not that useful.

Tim Scarfe 00:42:15

Yeah. Even then though, there there's this kind of analogical relation between flexible forms of of thinking. And I guess I I use the word emergence because it's just surprising. It's it's a surprising arrival of a capability which gives a significant uplift. And you you see what interesting dynamics as well. So you you SFT these thinking traces because I kind of think of the thinking traces as being a form of flexible thinking, that they give you more degrees of freedom to kind of operate in a particular situation of of intelligence. And also, there's an interesting relationship between how fat and how thin the base model is. So, with, o 3, it's a very thin model. And they found that it was easier to to scale, you know, a computer inference time. But they they hit the the edge of the sigmoid faster. Whereas Sonnet 3.7 is a much fatter base and it's harder to scale, but they've actually got more headroom left, you know, if they continue to scale. And then there's also an interesting thing that when when you fine tune the models, they fine tune like a Lama 1,000,000,000 model. And because the Lama model was kind of so thin, it's almost like the thinking traces couldn't take root. You needed to have like a base level of intelligence in in the in the model that you're fine tuning onto in order for them to work.

Eiso Kant 00:43:26

What what is thinking in this concept? Thinking is able to explore the possible space of a solution. Because right now, the reason we call this reasoning models and often not thinking models, the reasoning really is, to some extent, objective oriented. Right? It needs to be able to achieve something to be actually reasoning and then follow a certain step process. And the more complex the objective, right, it's still very clear you need a better understanding of the world, you need better understanding, you know, manipulation of language. And so thought in that sense is still something that I think is constrained by the model size, absolutely. Right? And it always was, like even before reasoning models it already was. But now we have an access to use what's already there and really improve it. Another way of thinking about this is if I sampled a model 1000 times with enough diversity introduced, either through temperature or prompt or whatever I was using, and I could find in there, like the correct thinking and lead it to correct solution, it means that somewhere there in the model it's already there. We just haven't found the perfect latent space for it. The best is, of course, if I just only do this with temperature, right? If I start allowing for more creativity of different sets of probabilities. Because what you fundamentally want is to be able to reward the model in such a way that for things that have deterministic correctness, at temperature 0 it will get it, right, and frankly even hopefully at a temperature 0.7 it will still get there. It will kind of expand open its cone of possible options and then it will collapse as it gets closer

AI-Assisted Software Development Today

Eiso Kant 00:45:00

to the solution to the correct 1. And when you take a very small model, you realize that that opening cone is very wide, but you can't actually collapse into the right solution, right, for many things. But if it is somewhere there already in that cone, then you can try to push it towards it. Right? And and that's where again, name of your podcast, right, like the machine learning side still kind of comes true, like you can still overfit a small model to a task, but you can't get it, you know, to actually generalize enough. And so, I often think about this as compression. Models are just a massive amount of data compressed into a certain space. The compression is too small, it becomes you lose way too much, And if the compression becomes lossless, like so large compared to the data, then you're not really learning anything. So, I also don't hold the view that there's this, like, infinitely large model world. Like, at some point it's going to make more sense to say, for this amount of compute, this amount of data that we're applying to teaching a model, this is the optimal size, and then we're going to want to maybe paralyze them horizontally to try to achieve objectives. The the bitter lesson, learn like learning and search.[5]

Tim Scarfe 00:46:08

I interviewed the the winners of the ARC challenge, and they said that, you know, language models, we we sample them greedily, which means, you know, we take the next token, the next token. And natural language is kind of messy, so there are many, many degrees of freedom. And they were kind of speaking to the fundamental trade off between, you know, creative thinking and reasoning. So, the art challenge, there is just 1 solution. Well, you know, not not 1, but there's a very sparse space of solutions. And they actually came up with a depth first search kind of sampling type strategy. And there are some folks at DeepMind I spoke to, and they were saying, you know, when when you do reasoning, you you actually want the softmax to be very precise because you want you want the thing, right? You want the thing. But sometimes you actually want to have, like, creativity and diversity. So how do you square that circle? How do you have both?[6]

[5] The Bitter Lesson — paper
[6] ARC Prize Challenge — benchmark

Eiso Kant 00:46:52

Well, I think there are problems where you need the creativity and diversity to get to that determinist, like, that final solution. Right? I think that's the notion of it. Like, if there is a single algorithm that you have to run over and over again, like addition or multiplication or something, then it's a very narrow cone of things that you want. But if you're trying to find You mentioned the general relativity of Einstein. Right? So for Einstein to discover general relativity at the moment in time and the axioms he had around him of truth, it required quite a wide cone of diversity of creativity. Right? To explore different ideas and then of course make sure they're consistent with what's already true in the world, to make sure to know which ones are correct and which ones are wrong. But if you would have kept a very, like, small, like, diversity, you know, like, you would have probably never discovered it. Right? It's the quote of it's a fine line between genius and madness, I think kind of applies, like, to models in this case as well. So, but at the end of the day, if you have infinite creativity, drag the temperature up of a model, like, you know, just it'll collapse into garbage. Right? It won't be something that makes sense. So, it's our job when we train models and build them, that for the things we care about, the things we define as valuable intelligence, that we make that trade off perfectly. Because that's actually what we're doing. When we're using reinforcement learning to teach these models, we are doing exactly that. We are saying, hey, when you're sampling in this part of the cone of possible, you know, thinking traces, you're directionally correct. When you're sampling in this part, you're entirely wrong. Right? And so if if you start going down the path of trying to apply calculus to a problem that doesn't require calculus, you want that entire set of thinking traces that stem from that to no longer be something a model does when it encounters a similar situation in the future. And so that's, I think, is the trade off what we're constantly doing, and I think human intelligence is exactly the same thing. You see this if you take someone who is very new in a field, very young. That's why I also think most interesting scientific breakthroughs come from people in their twenties and, like, you know, because they're no longer con there's no like constrained cone yet. They're willing to explore lots of different ideas. It's why you see people who are great at lots of different fields like a Feynman being able to come up with very creative ideas, but also they might at times, you know, go way too far off and don't get to the right objective. So I think intelligence is always that trade off. I don't think we'll ever get it perfectly right, but we can keep getting it more and more efficient and over time we might be willing to say, hey, for known compute budget work, which I think is the vast majority of knowledge work, like in the real world, accountant, software developer, we kind of know the intelligence budget. That's where you want that cone to be as perfectly narrow so that it can, you know, maximize for the economically viable work. But when we come to unknown compute budget

challenges in the world, solving cancer, the next breakthrough in material sciences, here we might say, you know what? I want that exploration much wider and I'm willing to pay money for it to be much wider so we can explore more ideas. Right? It's coming back to AlphaGo. It's how many moves do you wanna explore next? How much depth and breadth do you wanna have?[7]

Tim Scarfe 00:50:01

Yeah. I I love that. I mean, the the space of intelligence is very gnarly, and we work as a collective intelligence. There's a great book by Kenneth Stanley called Why Greatness Cannot Be Planned, and he basically said that, monotonic objective optimization is is the dumbest thing you can possibly do. So, you know, what we actually do is is through serendipity and what's interesting, our nose for what's interesting, we collect diverse stepping stones and many of them lead to to greatness. But of course, in in an LLM context, it's about just sampling and actually, you know, take taking in those diverse perspectives. But I I wanna talk about software a little bit. So so so Paul's side, your company, your product is intelligence, but initially you've been very focused on software engineering in particular. I can speak to just my own personal experience that Gen AI software has revolutionized the way I write code. I'm now writing software in, let's say, a month or 2 that would have taken me years to do before. It's it's absolutely incredible. But what what what is your main objective here? How do you see the sort of the software engineering trends changing?

Eiso Kant 00:51:00

The main objective is to achieve human level capabilities and go beyond. And that means that in a world today where we have maybe 100 plus million people who are building software, we want to bring that from 100,000,000 plus people to anyone who's able to build software. And the core and the sets of people who today are at the frontier of what they're able to do in software to be 10x or 100x more productive. And I think this just comes because what is software? Software is a lever that we have on the world to be able to bring productivity. Right? It's a lever to abundance. It's a lever to reducing the cost of things. And so to me, jumping really hard on the end of that lever, by putting the biggest weight on it, is putting the biggest, most capable intelligence on it. Because that allows all of us together with AI to pull that lever and to drive the cost of things down, and and that's always been the the thinking behind this. Now, I think it's important to not just talk about the future because otherwise you you get the the founder of a Frontier Eye company just saying AGI AGI AGI. And and I'm sure you've had plenty of those podcasts already. I think it's important to also know

[7] Mastering the Game of Go with Deep Neural Networks and Tree Search — paper

what can you do today. And today is in the intersection between model capabilities and limitations, a world that's developer led AI assisted. Right? So the AI capabilities and limitations and and and the human. And how do you find the perfect intersection by adding product on top of that? Right? How do you create a product that allows people the max personal lever that they have for their own productivity? It's like you said, do what you do, which before would have taken you years, you can do now in months. And that has a lot to do with the model, but also has a lot to do with the user experience. It has a lot to do, how can you bring the right context to the model? Right? How do you make it easier for it to find the information to be able to give you the correct answers? Some of that is external from the web, some of that is within your code base, some of that's within your knowledge bases. And so building really powerful assistants, today we do that in editors, we do that in web, we have soon CLI coming, I think is really critical. But you're This is like a symbiotic relationship. Like as your model gets better you can do more on the product. The product form factor changes constantly as the model gets better. You've already seen that it went from code completion to chat to now increasingly more argenic, and I think in the future increasingly autonomous. And so you just have to kind of be on that frontier constantly playing with all of those things.

Tim Scarfe 00:53:24

So, 1 of the things I've noticed, and again, the reason why I'm so excited in particular, I think it's really good for founders like me. I've got a very small team and I can just rapidly iterate. And I guess my process of coding has become more like I'm a reviewer. Right? So, I I, I get the language model to generate a bunch of code, I do a bunch of tests and increasingly rather than writing the code, I'm I'm just reviewing it and I'm saying that, know, looks good to me, looks good to me. Sometimes I'm going backwards, I'm going I'm going forwards. And the the the thing that I want to understand is, how does it work when you have teams of people? Because the the way we write software is, we have a mental model. We create these abstractions and we have some idea of of how the software should be crafted and we share that with with our friends. And now we're generating code almost quicker than our ability to review it. So how does that scale in Teams?

Eiso Kant 00:54:16

In the world today that's developer led AI assisted, the the question as always comes is like, you know, what requires essentially knowledge sharing, what requires review and what doesn't. Right? When you make a 1 line change to the documentation it doesn't require review. When you make a massive refactoring of the entire code base that impacts every developer, gonna wanna knowledge share that with the entire team and maybe get input. And I think AI is no different of you scaling up your team, right, or if you're scaling up AI in this case. So, it always sits in that boundary of where is it important to make that knowledge share happen and where is it not. And so code review has often been seen as this, like, process you have to do to catch bugs or to do x, z. I always think code review has truly been around knowledge sharing first and foremost, and then occasional where needed the ability to get input from other people because it touches surface areas that impact others or that you might not know or you weren't the best person on. And so when all of a sudden you're now producing, you know, 10x to code or you're moving a lot faster, and it sounds like in a domain you're working in, it already feels like AI is, you know, a very valuable partner, like almost like an anthromorphized, like, intern, right, in other domains it's not there yet, it should be seen as adding someone to the team. And I think increasingly that's the relationship that we're gonna have with AI, where it is like adding people to our team except instead of adding humans we're actually adding, you know, AI agents.

Tim Scarfe 00:55:43

So, 1 1 way of thinking about this is, you know, we've got this fairly linear software development life cycle. We have business analysis, and then we do story points, and then, you know, we we write some code, and then we do some tests, and then, you know, we we we approve all of those, you know, with release control and so on. And 1 way of thinking about this is it's about control. And it's about aligning the code that we write with our business objectives. That's why we have all of these different gates and approvals and so on. So what what does that mean when we have increasing autonomy and when, you know, in the code crafting process itself, we can just do so many things. Does does this traditional software engineering life cycle give us bottlenecks?

Eiso Kant 00:56:24

I think a lot of the software development life cycle collapses into the models over time. And so But I think it depends again on the environment. If you're, you know, writing code for nuclear missiles, you probably want a lot of those steps of that linear process to to still exist even if AI is the 1 acting in it. Right? Because you care about a certain number of nines of reliability. In that case, probably, hopefully, not even nines, just 100% reliability, which you all know is impossible in software. Right? But we want many. So you're willing to invest in that. In other places, software becomes more ephemeral. Right? Maybe you write something that works for a week as a tool or does a set of tasks, and so there the SDLC doesn't really matter. And so on those 2 extreme ends of a spectrum, you kinda shouldn't realize about myself, I use spectrums a lot, it's just the way my brain works, is as AI gets more capable, we're gonna probably have a distribution of software that tilts far more to the left in terms of volume of software that's closer to not requiring those stringent processes than to the rest. But the world's global banking infrastructure is not going away. Right? And there we want a certain set of checks and balances in place. The question just becomes, is AI acting across all of those checks and balances? Is it some point it becomes so reliable that I can start removing a lot of them? Because if I have a a software developer, human software developer who never, you know, makes a mistake, never writes a bug, his CI test's always 100% passed for the last 5 years, at some point I might say, you know what, give the 3 hours of CI back to the human. You know, let them just move faster. Now again, that's a theoretical example because in the real world that's not perfectly like this, but as we get more and more above human level capabilities, at some point we might just say, you know what, that's fine. Maybe even unit tests fall away, maybe even CI, like all of this starts becoming less and less. So I think it's useful to think about things in terms of the limits, not because we're at the limits tomorrow, but because it allows us to

Architecture Innovation and Model Interpretability

Eiso Kant 00:58:20

kind of show where we're heading towards. And so I think, yes, in the fullness of time, a lot of the software development life cycle collapses into the model and and doesn't require many of these checks, but not everywhere, not all at once.

Tim Scarfe 00:58:31

So thinking about the sort of the dynamics of software engineering in large enterprise, you know, a lot of a lot of companies in the FTSE 500, they frankly can't hire really talented software engineers. There was always this problem that they would do low code and no code. They would, you know, build things on Microsoft Power Platform and so on. And and now we're in the era of Gen AI coding. And now almost anyone can write amazing software, almost discardable software applications to do whatever they wanna do. So do you think that we're gonna see more folks writing code, and how will that change over time?

Eiso Kant 00:59:02

I'll have to say, and and look, that's maybe because I spent so much time with enterprises, I I think we find incredible software engineers everywhere first and foremost. But but but I hear you in terms of what you say is that, you know, what you get paid to be a staff engineer at Google is not the same what you get paid to be a staff engineer at a bank. Right? And so so there are some distinctions here and there. But overall, I think there's there's great developers everywhere. What we're seeing right now is that AI is while it might be for you in your field or for greenfield projects already at the place where it feels like that massive, you know, unlock, I still think right now in most enterprise environments it's feeling like a 20 or 30% productivity gain. And it's a productivity gain for some people by 2 x because they're doing, you know, unit test automation and all of a sudden now it's just way faster and maybe even becomes 3 or 4 x. And in other places where we are working with, you know, a company specific programming language in a specific domain where the models are really not that good yet. And so I think it's worth acknowledging that model capabilities mapping over to the real world in enterprises is not a 1 size fits all. And what I do see is that people are excited about being able to do more. If that's being an existing developer who can now, you know, take the boring parts of your job and actually automate it or can build a lot more software faster, or the fact where, yeah, I now see a product manager going, oh, I can actually build this prototype myself and show it, you know, to the managers and then see if we wanna go build it out like at a better skill. So I do hold the view that more and more people will be able and will want to build software, but the want is an important part. You know, because we're software developers ourselves, we often assume that everybody wants to build software if they're able to. And the reality is not everybody wants to build software even if they tomorrow can, but the product manager who's always wanted to build his ideas quicker, absolutely. Right? That maybe 1 in 5 business person who, you know, has this idea that he's been trying to rally a team around but couldn't, now can. But it is not universal that everybody in the world will be building software because it's still something that requires you to wanna do it.

Tim Scarfe 01:01:06

Yeah. I mean, I guess I agree that clearly the the lift is, you know, the the naught to 1. Right? You can now build individual applications just in seconds and and it's incredible. But I I still think that there's there's a much bigger lift, and the only thing blocking us is is a lack of imagination. So, example, the reason why Google engineers are paid so well is they're building scalable distributed systems that millions of people use, and, you know, the multi agent fault resilient systems. You can still build systems like that. And I guess the question is, some of this is an education thing, right? There are good ways and bad ways of using Gen AI code. A good way of doing it is understanding that there is a complexity ceiling, and if you build a monolithic application and you just keep building on top, on top, top, it's going to collapse. But you can, it's almost like it's teasing you to design modular, almost serverless type, you know, multi agent type systems and and the LLMs can handle that, you know, to a much higher complexity ceiling.

Eiso Kant 01:02:07

I think you're right in mapping the current limitations of models to a really good way of getting around it, is building small modular things that, you know, the the models are able to to essentially understand easily and and work within, and having those kind of architectures where there's some separation of concerns help in that world. But it's the models of today. And I think this is an important part to always come back to that I don't think there is a universal limitation to the type of software that can be built by models when we talk over the next 3 to 5 years. I think but you're absolutely right in saying that if today I try to build a massive monolith application and just have to model Vibe code the hell out of it, at some point, the whole thing collapses in on itself. By the way, I don't know about you, but there have been times in my career early on where I did exactly the same thing. I would build something and build something and build something and at some point I'd be like, oh my god, what monster did I write here in code? And then it pushed me to refactor it, make it more modular, make it better. Right? And and so we're seeing limitations in the models today while they're still far from our capabilities, also we can reflect on having seen in ourselves at different moments as well.

Tim Scarfe 01:03:12

Yeah. And that that's really interesting because it's like the language models, they kind of scale with quadratic complexity and software scales exponentially of complexity. So it allows you to build software which is 2 orders of magnitude more complex than you would otherwise have built, but you still hit the complexity ceiling just just, you know, very, very quickly.

Eiso Kant 01:03:28

That's an interesting way of looking at it. I haven't given that much thought, but I'll have to do so.

Tim Scarfe 01:03:32

The the other thing which I think is really interesting is, at the moment, you know, when people do generic coding, they're they're they're generating software and and they're doing unit tests and so on. But it we still have this fairly linear mode of software engineering, which means, you know, we we we we have release control, we put stuff into production. Now we're starting to see the advent of like MCP servers and so on, which means that the during the development process itself, the the intelligent system can actually talk to your database. It can say, well, what's the scheme on on the live database? Or talk to my actor system, how many actors are in play at the moment? You know, do I need to repair this actor? So now increasingly there's an operational layer to to the software process.

Eiso Kant 01:04:10

I love to anthropomorphize this for a second. Right? What we do as developers is we'll open up the database console and we'll check the schema. We'll go talk to someone on Slack or in person and we'll gather information and I'll pull up documentation. And so I think that some of the protocols that that the world is creating are a way to make that with current model capabilities easy for the model to do. I think again, if we play this out over the next couple of years, I'm not sure if it's a protocol. Right? I'm not sure if it's just a computer use agent that's been doing this or the malicious writing code to hit the API, you know, of of Jira or or hitting the, you know, like, just directly connecting to the database and executing a SQL command to get the schema. So I think there's things that we build today that matter for the limitations that the models have, and they matter and they're useful. I think the interesting question that stems from this, at some point when we talk about a large multi agent system, just like we talk about a large company with lots of developers collaborating, how do we make that collaboration efficient and work well? Right? Is this a world where we have 1000 instantiations of our model, you know, each essentially acting upon, you know, in an organized collective like a company? Is that hierarchical in nature just like we are in our organizations? Because there's things that we can't do. I can't access the thinking traces and solutions of my, you know, 500 peers. It's not can't be stored in some central database, but it's something agents can do. And so all of a sudden I think we we while we're building towards human level intelligence, there's already things that models today can do that we can't do by

the nature of how we operate. The paralyzation of being able to run through an entire code base and summarize it file by file over 1000 files. I can do it, but it wouldn't take it took a very long time and not be so efficient. The length of context windows. I don't know about you, but I can't hold 1000000 tokens in my context window and retrieve perfectly across it. At some point, we're gonna have to get away from this realm of like andromorphizing it and starting to say, okay, these are certain things that models can do in different And that to me comes to these protocols is, right now we need them. I'm not sure if we need them in 18 months.

Tim Scarfe 01:06:23

You were speaking to something interesting there, which is that there's a kind of semantics gap in code. I mean, there was always a famous adage that, many companies, they didn't wanna release their codes on GitHub because it's so valuable. It's it's actually not that valuable because the semantics, the intentionality, the motivation behind the code is not in the code. Know, it's the same thing with language, you know, like there's there's a lot of missing information that is that is not in the data. But we can capture this information because you've now got an organization of developers. They're building code, right, using these tools and the entire thought process is in language. So, could capture that into a kind of semantical database, you can rag into it and now, you've just got all of this kind of meaningful motivated information, which means the language model won't always be making the same mistake because it knows all of the reason why we did this is because of that.

Eiso Kant 01:07:13

There's lots of really cool stuff, right? We've never, like you said, we've never had the ability to trace the thinking of humans and store it. And as these models are being deployed increasingly more as assistants and and future agents and even autonomous agents, we can now all of a sudden access that. The question is will we access it or will the models access it themselves? And and and so I I always try to be careful with going too many steps into the future because at some point you're, you know, you start collapsing into, like, sci fi. But I don't think we are that far out from this specific scenario. There's quite a few people who won't agree with me on this, but I think it's really important that as long as we can, we should keep models thinking and reasoning in language. There's incredible work that can and probably even more compute efficiently be done with latent space style thinking. There was a great meta paper about the latent space reasoning tokens. There's been some other approaches that I've seen that I like a lot. We saw a language based diffusion model coming out, like, you know, earlier this week. I I really like the guys behind it. And and so I think there's lots

of architectures that will work. Actually, my team kind of gets tired of me saying that probably every architecture can work in the world, it's just a question of how compute efficient it is. So I had no doubt the fusion language models were going to work. It's just, is it the compute efficient thing for the type of capabilities and tasks that we care about? Now, the reason I mention language being important is that as models get more capable, having the ability to see the reasoning and thinking traces that they have, and like you said, reference them from the past that leads to what led certain decisions, I think will become massively valuable both from an interpretability perspective, from just the ability to build upon work that was done previously by other agents. And I think there is a discussion to be had on if it is going to be useful from a safety and alignment perspective as well.

Tim Scarfe 01:09:04

There's this diffusion based language model is a great example. I've been very excited about this for years, but they never worked particularly well. So this this 1 that just came out that he spoke about that it had roughly 10 x efficiency advantage. So, you know, like with the auto aggressive language model, you have to actually token token token token. This thing, they only ran diffusion about, you know, 5 5 or 6 times. They got the same results. Diffusion's really good from a code point of view because, you know, just like with vision diffusion, you can actually edit things. So you can say, okay, I wanna I wanna hold this code fixed but I wanna edit this bit in the middle. Why why is everyone not doing this?

Eiso Kant 01:09:37

There's so many architectures that can work, and I think diffusion is is 1 other field of it. But we've done a lot of work in our in our industry and in in each frontier company to make the ones we have really efficient. And so, to shift over from 1 to another, you have to have the efficiency gains, you have to be willing to take the the time that it takes to actually go after a new architecture. All of the experimentation you've done until that moment, right, does it still hold true in the same way or not? And so, you know, we made a big bet a little over a year ago on linear attention, RNN inspired attention. And we've had models in production since about fall of last year already with linear attention. And so we would ask, like, why is not everybody doing this? Right? Why? Like, this makes so much sense or it seems like gains, etcetera. And part of that is also just because all of us end up investing heavily in a certain area and and then we just scale it up further, and making that shift needs to be really really valuable to do so. And so I I need to dig further into the diffusion language models in terms of how much of that efficiency, you know, shows up compared to some of the things that we're doing or others. We're unlikely to go down this path ourselves because of different things that we've done in other architectures that that we maybe just don't disclose yet.

But I think it's exciting, and I think we there's a lot of room left for architecture research. I think the fusion language model is the tip of the iceberg. I think a lot more things can be done, but also they need to be skilled up. So some of the coolest stuff that we've seen in open source, you know, or others comes at this 7,000,000,000 parameter, if if at all it's 7,000,000,000. And then the question becomes, well, how does it, you know, operate at 70,000,000,000? How does it operate if we wanna try to make an MOE equivalent of, you know, of a diffusion language model? Like And so, at some point, it's like what works at small scale very likely can work at large scale, but does it stay the same efficiency? Right? Can you get those same gains?[8]

Tim Scarfe 01:11:41

Yeah, that's a very good question because they built a mid sized model, and it was commensurate in performance to the other frontier mid sized models. The other thing I loved about it is you can actually run it, you can do an unbounded amount of computation. So you can actually just continue to do diffusion and from a test time computation point of view, it's actually very very flexible. I I love that. But just coming back to the software engineering thing, so I I think what we're seeing though when when we do increasing AI dev is a lack of autonomy on the parts of the humans and a lack of legibility. Right? So we're now building increasingly inscrutable software. Now, know, let's make no bones about this. No 1 understands, no 1 at Google understands how the software code, you know, so we shouldn't be hyperbolic about But, we could, as we were just saying, we could design an information architecture which mediates a cognitive interface, which means at least at some level of abstraction, we understand what the thing's doing, we're putting guardrails and and so on. But we are describing a future though where we're just building these inscrutable monsters. What what does that look like? In terms

[8] Mamba: Linear-Time Sequence Modeling with Selective State Spaces — paper

Eiso Kant 01:12:43

of to your first point of of models writing code, I think code is that higher level abstraction already. Right? We can all go in and try to understand code and maybe add some print statements and we spend time and it takes look, it takes cognitive load, but code is deterministic, it gets interpreted by a compiler and all of us as humans, if we are willing to invest the time and effort, can understand it. It's just, is it worth it? Right? So are we okay with building big software code bases that we can't fully understand? And what we historically already have been with human intelligence, like you said to your example of the Google copays, not a single person understands it anymore. And we're totally fine with that because it does what it's supposed to do. And when we want to actually introspect a part of it, because there was a bug and it went wrong, or because we deeply care about the recommendation algorithm that X released, you know, open source. Like, we can choose to spend time there. But just like with human built software, we don't always spend time going back. Look at the amount of legacy copays that no 1 has looked at for years, but they perfectly run. So I think that's just a choice. To the point of model, not the copays, but the model being the quote unquote, you know, monster like the the the interpretability, there I think it's gonna become increasingly important for good interpretability work. And I think there are question marks if at the limit of model capabilities interpretability will be as useful as it is at a call it, you know, less, like, smaller scale. What I mean by that is I'm not sure if we will ever be able to truly understand the reasoning and thinking process that happens inside a model's neural net, just like we can't in our own biology. And so But I'm super excited about some of the interpretability work that Chris Olah's team has released and has been public about. I think that's an incredible direction. I I highly want to encourage everybody to be doing, you know, this type of work and and and trying to understand what's happening in models. I think that as we if we keep models reasoning and thinking and language that adds a layer of interpretability, it doesn't mean that's exactly what's happening underneath in the model. You could imagine a model that develops perfectly fine sounding reasoning and thinking chain, but actually it's trying to optimize[9]

[9] Zoom In: An Introduction to Circuits — paper

Karpathy's Software 2.0 and the Future of Code

Eiso Kant 01:15:00

for a different objective. And that's where I think safety and alignment interpretability kind of meet in the middle and really matter. But I I like to think that with good work on interpretability of what's happening at the activation and and weight level of the model, combined with keeping models reasoning and thinking and language, and trying to understand if those 2 things stay consistent, we can actually do a really good job at alignment and at safety.

Tim Scarfe 01:15:26

Yeah, the interesting thing is because the language models are trained on all the colloquial human code, they produce locally interpretable code. It's a little bit like the the, you know, language models, they are actually surprisingly aligned, like when I'm when I'm using open interpreter on on my CLI. If I tell it to delete all the files on my file system, I would say no. And I've built an LLM application and and and if if I tell it to change your name just to to someone else, it will say no, actually, know, it's OZO, this his name, I'm not gonna change the name. But but but then there's this kind of like global eligibility. But there's also the thing that we're generating code, but also there'll be a mixture of code and models. And some of the inscrutability will just come from this thing is like a living thing. You you know, like there was that Black Friday incident where, you know, we're doing high frequency trading and automated trading, and it works really well until it doesn't. And then you get this cascade effect, and the locus of control is now in the algorithms and and the models and and the machines. So we could easily build these very complex systems that seem to work until they don't.

Eiso Kant 01:16:27

So I would take everything you just said, and if we replace models with humans, it would still hold true as well. Right? I think there I think that when we talk about here, there's this massive complex financial infrastructure co pays that's been built by all of these humans, where not a single person, you know, knows it anymore, and our locus of control is not on any 1 anymore, and then an unexpected event happens. I I think we often take what can go wrong and put it on the models now, but it's already what the world is like, and so I think the best job that we can do is to make these models more capable than us. Make them highly capable in in the ability to write good code that covers the edge cases. That's not lazy to write the tests. Right? That's another thing. Let's talk very honestly about ourselves. There's things we enjoy doing and there's things we don't. How much of the world's code doesn't have proper test coverage? How much of the world's critical code doesn't? Right now, all of a sudden, if I can decide to deploy dollars to compute, to intelligence, to shore up the test coverage of critical code in the world of financial infrastructure, to shore up the security of I think it's pretty much commonly accepted that the most of the world's critical infrastructure runs on really insecure code. Right? The power grid and others. If now I can say, okay, we're willing to now invest, could be from a company, could be from a public sector, 1000000000 dollars in making the code of our, you know, electrical power grid more secure. But now I don't have to bring humans together, but I can do this with AI, and I know it's not gonna be lazy about writing tests or other things. I think that's exciting. Right? And and we get to focus on the things of exploring the frontier of science together with AI and the things we wanna spend our time on. We have a shortage in the world of people to write incredible software and code, and that just shows on the amount of legacy systems we still haven't upgraded yet. And so I think this is a way of overcoming that, not necessarily, and of course, just like with humans, it adds another area of of fault, but I'm not sure if Which 1 of us in the fullness of time would you rather have write code for a critical system? Like I often say privately is like, I want the nurse to be a human, but I want the surgeon 1 day to be a robot.

Tim Scarfe 01:18:38

Just looking at the evolution of AI software, I'm very excited about metaprogramming. So, can actually have the system, repair itself and generate its own, code, you know, in response to failures. But then, the next stage of evolution is, why do we need code at all? Right? Why why don't we just mean, it's so called transduction where we don't even have an intermediate code, you know, step at all. We we just get the model just to do the thing, right, and and the whole thing is is adaptive. Are you excited about that?

Eiso Kant 01:19:04

My opinion over the years, since 2016, has has changed on this. I used to be of, like, Karpathy's quote in his I think it was his 2016 blog article Software 2.0. Software 2.0, right? Where it's like, at some limit, you know, everything just becomes a model. I think about it a little bit different today, no longer that extreme. I think that there are parts of the world's infrastructure that we want to have in something that's interpretable, and code is interpretable. It can be traced, it can be tested, it can be understood. A human or an AI can go in and understand it completely. And so our financial payment infrastructure in the world probably want that to be in code. Our, you know, electrical grid probably want it to be in code. What's operating, I'm I'm seeing the railroad out here, you know, the switches between trains probably in code. But do I care about other pieces of software if it's, you know, just a neural net simulating the whole thing behind the scenes? Now I think I also want to come back to what code is. Right? Code is operating in most cases of most software, not everything, on a CPU. Right? And and we have done an incredible job of optimizing the hell out of that evolution of hardware to be able to run deterministic code that, you know, can then serve us to be valuable. There's lots of places where for a long time the models might not be even compute efficient enough. It might just be too expensive to have all of software in the world collapse into a model call. Right? But you're not right that if we get to a point of model capabilities that are so good, so trustworthy, and we can treat them as doing deterministic things. Right? I know that every single time I ask X it is so aligned that it's going to do Y, because that's essentially what code for a CPU is, right? Like at the end of the day we are asking for something to be truly deterministic, then maybe there is a, as those cost profiles change, more and more will move to the models. But I I still, and maybe, look, maybe I'm holding onto an old idea, but I still like to think that it will be cheaper for a capable model to build and maintain the Uber Eats app and have it run on CPU infrastructure as maintainable code to update it and change it than it will be to simulate the entire thing as a neural net. Right? And so I think for a long time,

the code in the real world will continue to exist.[10]

Tim Scarfe 01:21:28

The amazing thing about these AI models is that in so many ways, they're just smarter than us. So they can they can basically write or learn functions that we can't write code to do, which is which is incredible.

Eiso Kant 01:21:40

A 100% for NPR product. There's so many places where this is going to be incredible.

Tim Scarfe 01:21:44

Absolutely. So maybe it it is a spectrum. And I mean, I was I was having a bit of a galaxy brain thought. I was driving to, you know, to pick Marcus up, our our creative director this morning, and Google Maps, it was taking me this weird way. And I was thinking to myself, what's it doing? Like, is is this thing being a utilitarian? Is is it actually, like, optimizing for the, you know, to reduce the average route time and it's taking me down a bad way cause it doesn't care about me. And I I don't know. And and in the future, these these AI systems, they this is kind of what I mean by the loss of control. Like, it might be doing some weird galaxy brain thing, and it might not necessarily be what's right for me, and and maybe I I want to know about that.

Eiso Kant 01:22:23

I think today there's a lot of machine learning systems that already we don't have any good interpretability on, if that's in fraud detection, if that's in actually in in in the algorithm that's mapping you to your location. Right? I'm not sure what's behind the scenes of Google Maps, but it could very well already be a neural net that is very task specific. I wouldn't be surprised or at least a part of it, right, likely is. That's learned on a lot of data of, you know, the London traffic patterns and whatnot. So I think we've already faced that in machine learning and we'll continue to face it. It's There's parts in the world that we want things to be deterministic in my view, and we're gonna want to keep them. And there's other parts where you're absolutely right. But I want the most optimal route planned from somebody who understands me, not just the general London traffic. Right? Maybe it's the fact that I prefer to take, you know, trains that are less busy then ones that are more busy and I'm okay to get to a certain time. I think as we add intelligence, generalized intelligence, right, so human like intelligence

[10] Software 2.0 — blog

to the world, We've got a lever that we can make continue to more compute efficient that we might wanna use it. I might wanna use, you know, AGI to help me figure out my route, right, because at some point it's gonna be so cheap that it's kind of worth it to do. So I do something today with the models, it's gonna sound very silly, I travel a lot for work, and every time I before I fly around land somewhere, I tell it my schedule, I tell it what I ate, and I say what's my best plan for dealing with my jet lag as I'm coming in this schedule where? And then I follow it. It will tell me, you know, you know, protein meal with little carbs and sleep for 90 minutes here, and I will just follow the models. Now look, our model wasn't trained for this specifically, but it's a pretty good generalized model and so far it's worked pretty well for me. But I use it for that specific reason because all of a sudden intelligence is not cheap enough for me to actually get it to do this. Now you could write an algorithm, or you could write code that takes all of these factors into account and calls the API location of the weather and things and like Or you could just trust the model. And I think these 2 things will always exist, but the code is going to be written by the model in the future, and it's choosing to do things in code because it's just more efficient and more deterministic, or we want it to stay deterministic like payment infrastructure.

Tim Scarfe 01:24:29

Do you think we might lose something when when we do this? I watched a great YouTube video last night, and it was it was this guy saying, in the in the days of analog recording, you know, we used to have to record things to tape, we could only do 1 take, and there was lots of noise and and so on, but it created this kind of serendipity that, you know, like, we we can't just delete it and start again. We have to do it again. We have to talk about it. You know, like, even this filming day today, it's a very creative, serendipitous process. And do you think by kind of like offloading so much of the intelligent thought process to machines that we might lose something?

Eiso Kant 01:24:59

It

AWS Partnership, Enterprise Security, and Closing

Eiso Kant 01:25:00

is worth asking the question, do we think in the same ways today, have has our own thought process evolved in the same ways today as it did 100 years ago, 500 years ago, 2000 years ago. And I think there are things that as we have learned more representations of knowledge, of thought, how we have as we started getting augmented to tools, like, you know, I'm still of the age where I remember, you know, kind of pre Google, I was young, but a pre Google and post Google world, right? And now all of a sudden I'm in a world where I'm willing to all of a sudden start, you know, no longer remembering certain facts. Actually there's very few specific, like, you know, knowledge or history facts that I still remember. Where probably if you and I were born, you know, 150 years ago, remembering the things that we had learned from reading books and other things would be critical to us. Right? And so that's changed, and and it's changed because we now have a tool that we can use instead. So I might not need to know anymore the theory behind jet lag, right, and the biological process because I trust something else to make a decision. But I think we've always had versions of that. If that was in the past trusting, you know, the the local, you know, wise person on x y or zed or the fact that we go to a doctor, like, to trust. So we'll change. Will our own thought process evolve? Like, if you give enough time over evolution of this, you know, will we become smarter? Will we become less smarter? History hasn't shown us, even though people like to say it in the moment, become dumber because of technology, I actually think it's made us more enlightened. It's made us more, you know, ability to explore lots more ideas, do a lot more, like technology progress has continued to be exponential, and I think it's exciting that it continues to stay exponential. So maybe I'm an optimist, but I do think there's a fair argument to be made if you're a parent. Right? And you've got a young child and you essentially say, like, I still want you to study this. I want you to learn these things. So it's developed your own thinking. It's kind of the TikTok debate. Right? Do you want to give TikTok the entire day to your kid, you know, and have it spend 8 hours a day just on the phone? I'm not sure if that's, you know, how you want your children to to grow up and develop their intelligence.

Tim Scarfe 01:27:09

Would you encourage your kids today to learn how to code? Yes. Go on.

Eiso Kant 01:27:15

Because just like you would encourage your children to learn history, to understand mathematics, to learn how to program and write code, I think you want all of these things because at the end of the day, we still need to train our own intelligence as well. And so just because we're now getting increasingly capable of training models to reach our level of intelligence 1 day and even beyond, that doesn't mean that our experience of life should be 1 without us actually training our own intelligence. And I think coding and building software is a great way of training our intelligence. I, today, in my job, don't get to write a lot of code anymore. It's kind of just the nature of being co founder and where my role is, but everything that I learned from having written lot of code in my life has helped me develop thinking process and understanding that I like to think make me more capable of doing what I do. So I think just developing intelligence in general is a good thing, and I think coding is a good tool for that.

Tim Scarfe 01:28:11

Can we just speak about the role of multi modality? So at the moment we're primarily talking about, you know, text in this, you know, in this generative AI setting. I can imagine a future where it seems like low hanging fruit, where we just record the screen and it can perhaps in an observational way, can see the application, maybe in the future, it can interact with the application, maybe in gentle way. Is is that coming?

Eiso Kant 01:28:34

Oh, 100%. Like, look, multi modality from both visual language modeling to computer use to the work that you see that's happening in video and building world models in the world, all of that is coming. I think the question always comes down to the companies that are working on it need to work on it in terms of what their objectives are. So when I talk about right now building the world's most capable AI for software development and us being on the trajectory towards that, There are things that we care about that are massively represented in text, right, characters, language, code, etcetera, but also things that are massively represented in visual understanding of what's on a screen because applications are are on screens, right, to put it. And so building the models understanding to be able to know what's on a screen, how to interact with it, to have computer use like capabilities so that agents can go and open up the Amazon Web Service Console and and, you know, click around and find the the data that they need, I think is is critical. I think when you focus on developer software development capabilities, a lot can be done with text modality. With text modality, you're not gonna get the model to create beautiful UIs. It's just you can't get, you know, that iteration

on UI requires visual, like, you know, modeling and having it as a modality. But where where maybe a computer use agent is useful to go to the Amazon website and log in and find the IIM role, I can also have a model understand how to do that by making an API call. And so I think we put ourselves with a bunch of guardrails by focusing on software development capabilities that allow us to focus on modalities that are slightly more compute efficient than video or image generation, and hence use our compute more efficiently to gaining to getting towards our goal. But if I was trying to build full self driving, right, my modality would not be text.

Tim Scarfe 01:30:22

I'm quite excited about a future, you know, there's like the Genie paper from DeepMind for example, and, you know, so they're they're talking about being able to to generate video which is interactive in in real time. Could we ever, in the future, have software which is generative? So the the user interface is generative because we all think about things differently. Right? Like, the the user interface for you, you know, the optimal user interface might be different for me.

Eiso Kant 01:30:45

I don't see a reason why that world can't exist. I think the question is where do we want it to exist? I want my Uber Eats app to look the same way every time I open it because I build up my own I train my own model to be able to make sure that, you know, I can find, you know, where the food is that I wanna order. But there are places where I want a UI to be dynamic based on maybe the data that's behind something. I maybe want it to be more tailored towards me. But I venture to say that the vast majority of people who consume software in the world, us as humans, actually don't want our UIs changing massively on a day to day basis. We want a set of consistency in our UIs. I think this is human behavior, but I don't think there's any technical reason why that can't happen.

Tim Scarfe 01:31:28

Interesting, interesting. What's your relationship with the cloud providers?

Eiso Kant 01:31:32

Publicly, we have announced in December that we have what's called a first party relationship with AWS, Amazon Web Services, and it's quite a unique 1. What it allows us is that when an enterprise customer is buying poolside, they can buy it as if it's under Amazon's paper, meaning that they're the seller of record. And so this opens up a path for large complex enterprises to contract poolside and and bring our product in as if they're adding on another Amazon service. So this usually reduces the the time that it takes to get started with us, And it also allows them to fully burn down their spend commits. Right? Enterprises have large spend commits with Amazon, and so this is a way that we we have done this for several reasons. 1 is Amazon's distribution in enterprises is massive. Right? They're the largest service area of any cloud provider in the world. And if you think about our business, we are in a capabilities race. We've spoken a lot about that today, but we're also in a go to market race. Right? Our ability to land customers and grow revenue allows us to invest massive amounts of capital into more compute and into more talent to be able to scale up, like, our model capabilities. And so these are very symbiotic and hence the relationship with Amazon, which has been amazingly exciting. We've also done a lot of work with them, not just on the go to market side, but on their silicon. We've done a lot of work around Trainium 1 and Trainium 2, and so we have a fully dedicated team towards it, and we're pretty excited about what they're building there.

Tim Scarfe 01:32:59

Yeah. It's great for startups as well because you you get a bunch of free credits with Amazon, so presumably you can just use the free credits for for your service.

Eiso Kant 01:33:05

I think we're at a scale of compute where where credits is not is not what makes the difference anymore to be very honest. Yeah. But for our customers and others, it's it's absolutely look, it's it's part of it. No doubt. I I think when you're you know, today we're very much focused on enterprises and and and who we work with, and and what we see for them is where you're deployed makes a big difference for their security profile. So ins we deploy inside Amazon Web Services VPCs, right, in that private account, and the model waits and the full stack live there. And the combination of that really allows enterprises to get comfortable with the model that's going to their data instead of the other way around.

Tim Scarfe 01:33:45

Interesting. And just from a sort of getting started point of view, so folks want to have dedicated hardware. Do you tend to find that your customers have a centralized model where they have 1 shared implementation or is it more complex than that?

Eiso Kant 01:33:58

It really depends on the enterprise. A lot of what we find is that inside enterprises there are security boundaries that mean that they need multiple instantiations of the model. So if they're fine tuning a model on their proprietary SDK that they have internally but that only covers 1 business unit, and that model is not allowed to be shared from regulatory compliance reasons with another part of, like, the business. You might find organizations that want to spin up many versions of poolside models that get fine tuned in many different environments towards certain use cases. From where we sit, see all types of complexity, from on prem to VPC to different model instantiations, different security boundaries, different onion layers of access rights, and we've built for all of that. Like, it's it's we talked a lot about the model today, the sheer level of engineering we've had to do to be able to build, to succeed in defense, in government, in financial services has been actually quite a lot.

Tim Scarfe 01:34:51

And and finally, do you have a forward engineering team just to help folks get up and running with the infra and the architecture?

Eiso Kant 01:34:56

Absolutely, yeah. So we have the poolside solutions architects that are fully able and willing to spend a lot of time with our customers. But what we have increasingly done is that we've increasingly built towards a managed install approach, so that if an enterprise is going to give us temporary access rights, very, very limited access rights, we can spin up the entire infrastructure of poolside in their account. So what before might take, you know, several days of effort can now be done actually in under 40 minutes by just providing a single IAM role, but there's enterprises. There's always going to be something that you are going to discover along the way of a massive deployment, a firewall that sits on the network somewhere that needs a certain setting tweaked, and that's of course where often our solutions architects are very help helpful. But also just in terms of helping them think through, you know, as we're making these models their models, we're fine tuning them on their data and making them more capable in their environment, helping them think through about what data, you know, which groups of engineers to deploy, what versions of models with, how to measure the impact because we offer quite a lot of metrics that are available to the customer to see how the impact of the model is towards acceptance rates, towards how many changes are reviewed but not, you know, actually not applied, how many lines of code are actually applied. And so we do a lot of that work and and we try to help our our customers think through it. Our commitment is to become the trusted partner of enterprises, that as intelligence gets more capable, they'll wanna scale it up with us.

Tim Scarfe 01:36:21

Eiso, it's been an absolute honor. Thank you so much for joining us today.

Eiso Kant 01:36:24

No. Thank you so much. It was a pleasure.

References

[1] **Tufa AI Labs**
https://tufalabs.ai/
website

[2] **Poolside AI**
https://poolside.ai/
website

[3] **Eiso Kant on X**
https://x.com/eisokant
social

[4] **The Curse of Recursion: Training on Generated Data Makes Models Forget**
https://arxiv.org/abs/2305.17493
paper

[5] **Efficient Estimation of Word Representations in Vector Space (Word2Vec)**
https://arxiv.org/abs/1301.3781
paper

[6] **On the Measure of Intelligence**
https://arxiv.org/abs/1911.01547
paper

[7] **DeepSeek-V3 Technical Report**
https://arxiv.org/abs/2412.19437
paper

[8] **Training Compute-Optimal Large Language Models (Chinchilla)**
https://arxiv.org/abs/2203.15556
paper

[9] **The Bitter Lesson**
http://www.incompleteideas.net/IncIdeas/BitterLesson.html
paper

[10] **ARC Prize Challenge**
https://arcprize.org/
benchmark

[11] **Mastering the Game of Go with Deep Neural Networks and Tree Search**
https://www.nature.com/articles/nature16961
paper

[12] **Mamba: Linear-Time Sequence Modeling with Selective State Spaces**
https://arxiv.org/abs/2312.00752

paper

[13] **Zoom In: An Introduction to Circuits**
https://distill.pub/2020/circuits/zoom-in/
paper

[14] **Software 2.0**
https://karpathy.medium.com/software-2-0-a64152b37c35
blog