(no title)

48 MIN · YOUTUBE · HTTPS://WWW.YOUTUBE.COM/WATCH?V=LRGX-GTEGVA
https://www.youtube.com/watch?v=LRGX-gTegVA

SUMMARY

Yash Bottle, founder and CEO of Applied Compute, discusses his journey from Stanford to OpenAI and the founding of his company, which focuses on building specialized AI models for enterprises. He emphasizes the importance of tailoring AI models to specific business needs, the evolution of model training techniques, and the future of AI in terms of continual learning and data efficiency.

- *Yash's journey includes working at OpenAI on the post-training team and founding Applied Compute to address gaps in applying AI in enterprises.*
- *He highlights the transition from general AI models to specialized models that cater to specific business requirements.*
- *The discussion covers the evolution of model training, including pre-training and post-training, and the significance of reinforcement learning.*
- *Yash identifies continual learning as a key future bottleneck, emphasizing the need for models to learn from sparse rewards.*
- *He discusses the importance of evaluation (eval) metrics in guiding model development and performance optimization.*
- *The conversation touches on the challenges of data scarcity and the potential for synthetic data generation to address these issues.*
- *Yash shares insights on the future of AI architecture, suggesting that scaling transformers remains a priority while acknowledging the exploration of non-transformer models.*
- *He concludes with thoughts on the importance of compute efficiency and the potential for innovation in hardware and chip design to support AI advancements.*

Today we're going to talk about this part of the stack models and uh how do you build better models for better applications. Uh our guest for today is Yash Bottle, founder and CEO of Applied Compute. I'm so excited to have Yash here, not only because he's a grad uh a recent Stanford grad. He's going to talk a lot about his his journey, but also because he uh was one of the very few undergrads who went directly to OpenAI research um uh after Stanford was a part of the post training team. uh started applied compute um after OpenAI because of an insight that he had during his work at OpenAI and has built applied compute into one of the most uh successful businesses uh applying his learnings to incredible enterprises.

Yash, thank you for doing it. Please join us. [applause] Thank you. >> Awesome. >> That's on. >> Cool. >> Thanks for having me. >> Thanks for joining us. >> Yeah. >> Yeah. Tell us a little bit about yourself. You've had an incredible journey. Um and uh you've made some some tough choices. Actually, we were talking a lot with this class right before you joined about decisions about what to study. So, walk us through your your journey.

>> Yeah. >> Um and that led you to today. >> Yeah. Yeah. So, so I'll talk a little bit about uh my my journey. It's not actually that long. Um I uh I was actually sitting in here taking finals not that long ago and I'm I'm class of 25. So, um, you know, was this hasn't been that that that long since I've been on on campus and stuff, but, um, yeah. So, so, you know, grew up in Austin, Texas, came here for school. Um, I I like to say I was a very good student in high school, was a very bad student here. Um, not grades or anything, but, you know, kind of never went to class, watched the online lectures, did that sort of stuff. Um, and sort of the the the rapid fire history is ended up building a bunch of stuff on campus. Um, got connected to Sam Alman very serendipitous serendipitously through through some mutual friends. Um, one thing about Sam that I think not many people know is he has an incredible soft spot for helping young people early on in their careers. Um, so yeah, ended up meeting Sam. We hit it off uh freshman year summer um you know a friend and I were kind of deciding hey do we want to do our summer internships? Do we want to go and work on a project? Ended up saying hey let's go work on our project done our internships and we were kind of looking for money. Um Sam uh we shot Sam a blind email. He gave us a very small check to kind of like cover food and and rent and things like that. So worked on that for the summer. Ended up shutting it down coming back to school. Uh so that that you know came back for my sophomore year. um was doing a lot of fun things here on campus like tree hacks. Shout out to tree hacks. Uh that was kind of my main thing here. Um really loved putting on that hackathon for folks. Uh but then late 2022 is when chat GBT came out and I was kind of just like playing around with it and I was like, "Holy crap, this is the coolest thing I've ever seen. I have to go work on it." Like I couldn't think about anything else. So ended up shooting Sam another email saying, "Hey, how do I come and work on this thing?"

um he you know put me in touch with what was called the OpenAI residency. I think it still exists. It's actually how a lot of folks at OpenAI went from you know academic researchers or people in different industries to full-time employees at OpenAI. Um joined OpenAI uh early 2023 on the post training team. Um worked with some people I really really looked up to in in the language model universe. um starting on evals which you know a tip for anyone here is whenever you join a company work on this the sort of like hairiest thing that no one wants to work on because people will like you for it. So I ended up working on emails for the first year and then the second year was kind of when these reasoning models uh started coming out of the woodwork. So, uh, people were training these reasoning models on primarily competitive math and it was kind of this like wow moment for everyone at the company where we're like, oh, you know, we're seeing massive performance increases in in using these models. Uh, so a friend and I were like, hey, what if we actually try applying these models to things outside of competitive encoding and math? I I wasn't a competitive coding or math kid growing up. A lot of the Frontiers Futures folks were. Um, so we ended up hacking together this agent that could kind of browse the internet, write some code, showed it to a bunch of leadership. They were really excited about it. So we started this team called Long Horizon Tasks, and I was primarily focused on leading a lot of the agentic coding research, which eventually became became CodeEx. Um, but yeah, left left uh left to start applied compute about a year ago. So our one year was actually last Saturday. So hasn't hasn't been too long. But uh yeah, we we we sort of saw this gap where you know these models were getting really really smart. But when you actually went to go and apply them inside of the enterprise, they're like they're like smart geniuses that know nothing about your business. And inside of enterprise is actually where you have most of the data in, you know, in the world, right? Like all of these companies have have tons and tons of data, proprietary data that they've built up over time. So we're actually helping companies take the same frontier technology that led to these smart reasoning models and create their own specialized models um to sort of enhance their business.

>> Amazing. What a journey. >> Yeah, it's a lot of fun. >> Yeah. >> Um I thought we'd start at the models. Yash. So I went to the past century, things have clearly escalated. And then if you zoom in to this side of it uh post Alexnet >> uh things are moving >> pretty fast. >> Yeah. >> What like could you put this in context what what is going on at the model airframe for us? Uh why is the advancement in the last four years notable and and what is driving it?

>> Yeah. So >> and I got some of your slides if you want. >> Yeah. Yeah. Yeah. So scroll through them. >> I thought we'd start with like a bit of history. Um you mentioned Alex Dent. Uh does anyone here have have you guys heard of deep learning or what that is? >> Of course. >> Okay. Nice. Nice. Um deep learning was was kind Alexnet was kind of the the I would say the pivotal moment for deep learning and it also is kind of the moment that we stopped understanding what any of these models actually do. So essentially what deep learning is, it's it's it's it's a method of it's a piece of machine learning technology that allows you to learn underlying representations from data. Um and and you basically like train train on a bunch of data uh push in a bunch of compute and you get out these really smart models that you know are are made up of you know millions or billions of parameters. You actually don't know what they do. Um but they actually are really good at doing you know tasks like prediction um language model you know ne like next token prediction is is what like looms do and the sort of before and after was before alexnet people were creating handcrafted features sort of looking at um you know pieces of underlying data training very sort of like call it call it rudimentary classifiers to detect edges things like that on a lot of vision tasks and then what Alexet did is they applied GPUs um a massive data set called image nets against neural nets which um led to this sort of breakout moment where you actually prove that hey if you scaled compute and data you were able to see these massive gains in sort of predictive accuracy of these models >> you know the model development has a lot of different uh aspects to it >> uh maybe give us an overview I know we're going to go deep into a couple of aspects of it what is model training the modern day model training look like.

>> Yeah. So, so I think this is kind of a snapshot. This is like by no means the uh most detailed timeline of events of things that have happened, but I wanted to pick out a few a few pivotal moments um in in recent years. Starting with the the transformer. So, um you know, I'm sure everyone here has heard of the transformer and sort of self attention. This was sort of the moment where uh you know researchers at at Google brain came up with a new architecture that actually allowed scaling language model training.

Um it was way more performance on existing hardware. So G like could actually run these workloads on GPUs and compared to previous method like previous neural net architectures like recurren ne neural nets or LSTMs they were able to employ this technique called attention which basically led to way better performance in ne next token prediction and could scale to these massively long sequences in language. Um sort of fast forwarding over the years uh 20 2018 to 2019 was really this era of pre-training. So people were taking these massive corpuses of text teaching models to sort of predict the next token by optimizing on loss. So basically you have a model try to predict the next token. You see what the actual next token was in the in the corpus of text and then you do back propagation on the model weights to sort of tweak the model so that it's more likely to predict what that ground truth uh next token was.

Then you sort of entered the era of scaling laws. So starting with like the OpenAI scaling laws which showed hey if you actually scale these models up and make them really really big you start to get much better performance. Um so this was like the Kaplan scaling laws uh really proved out with GPT3 which was the first kind of model that seemed to have like some level of general intelligence. So that was kind of a breakthrough moment. And then you continue to compound on this with like the chinchilla scaling laws which should hey not only do you need to make the model really really big. You should actually there's actually a compute optimal way to scale these models. You both make the parameter size much larger but you also should train it on much more data. Um and then once we started to have these models that were sort of generally useful uh it became a sort of story of how do we actually make them useful to the to the normal person? How do we create like these inner interfaces where it's like a tool that the everyday person can use? And this was kind of the era of of reinforcement learning uh with human feedback preference tuning sort of being able to steer these models because general base models are are just doing next token prediction, right? So they hallucinate a ton. They don't actually answer your question. They may say things that are unaligned or not, you know, not up to safety standards. And then GPT4 was kind of this like you know next next level step change in the quality of these models.

Um I'll kind of go through these very quickly. I think this is what people are probably most familiar with in the past couple years which are the error of reasoning models. So you know in in in 2024 um you know OpenAI came out with this model called 01 which was kind of this new axis for scaling model intelligence which was test time compute and this was kind of felt this I I I want people to to to know that chain of thought is like a completely emergent behavior. um the the model reasoning whenever it answers your question and sort of spending time thinking correcting itself no one trained it to do that basically by putting it in these constrained RL environments and then funneling a ton of compute towards it you actually got these models who had this emergent property to be able to reason and then combining that with tool use which a lot of people you know if you guys use cloud codecs deep research you actually started to get these agents that could reason work for really long periods of time and sort of become um what people are calling today AI co-workers.

>> Fascinating. >> Um do you have a slide? Okay, great. This is this is perfect. So >> yeah, you know >> we have understood there to be multiple ingredients of that go into making a great model data compute talent. >> Yeah. >> Algorithms uh maybe other things that I haven't listed. >> Yeah. >> What is the bottleneck today? What has been the bottleneck in the past? And what do you suspect will the bottleneck be in X years from now? Maybe maybe a maybe a tour of history and then a prediction for the future.

>> Yeah. So I mean kind of running through what we what we just talked about, you know, the the bottlenecks kind of went from having the compute to train these models to the correct architecture to actually being able to scale to the pre-training you know levels of data that we need like the entire training on the entire internet being able to make these models usable by preference tuning them. And then today what the bottleneck is is I'm sure people have heard of like RL environments which is actually how these very recent families of models have gotten so good at at reasoning um and sort of intelligent thinking.

>> I think what the bottleneck is for the future is >> is it's it's basically this idea of continual learning which I

know we're going to talk about a bit later. Um but so far we've kind of gotten more more and more data efficient methods of training. So pre-training not super data efficient like you have to train on the whole internet to you know get the base model no one you know people here uh you to learn something you don't need to go and like sort of like read internet uh scale data so that that that was super data efficient then we started to kind of go to these these methods in post training which were a little bit more data efficient the most data efficient today being like RL environments but the thing that is kind of the holy grail and I think what people think of when they think of ASI or AGI is like How can a model go and do something once and learn from an extremely sparse reward? So, you know, just like you know, you guys, if if you're, you know, you go and burn your hands on the stove, it's uh you just need to do that once and then uh you know, the stove is hot and not to put your hand on the stove. These models today are not really like that. Um, so I think like this this this idea of continual learning and being able to be extremely data efficient with like in real world interactions, that's the next bottleneck.

>> Gotcha. Gotcha. >> Super helpful. Well, you know, when you burn your hand, it's also very loud feedback. >> Yeah. Exactly. >> So, so hopefully continual learning is giving us loud feedback. >> Yeah. >> Um, >> and you know, the other big question, maybe maybe not on the slide, is uh why have all the labs converged at focusing on software engineering? Why have they focused on code as the first frontier? I'm sure there'll be other frontiers like life sciences or or or cyber security or others that I don't know about >> but why software engineering why has what is the unique property of code that >> yeah people find so interesting >> so so the type of RL training that these labs are doing is our like reinforcement learning with verifiable rewards so in order to actually like get the learning signal or the reward signal you need to have a deterministic way to check if what your model did was the correct thing >> code and math are really really good for this because what can you do? You can compile the code. You can run unit tests against it. You can actually check to see if the code was is doing the right thing.

>> So you know that that is one reason why co code has been super valuable. The other is like it's really easy to make a lot of synthetic data on this >> right scale of data. >> The scale of data the prior is really good. There's a ton of code tokens on the internet. Um and then I think the other thing is just like a lot of researchers uh me included think coding models are kind of like AGI complete in the sense that every task when you kind of boil it down is a coding task right like so that's why you see Claude and a lot of these other uh models writing code to do instead of doing like uh you know tool calls or or or things that are are more specific to the task. they're actually just like using code as a general language to interact with the the real world.

>> Mhm. You know, one of the questions we were discussing right before you came is how do we get good at jobs with AI that are not code or code adjacent? Give you an example. >> Um making slides >> like this one by the way completely generated with with with cloud co-work with with initial set of conversations that we had. >> Yeah. Um, what is the relationship between code and slide generation that that would make these models good at generating slides because they got good at code?

>> So, did you did you make these slides with cloud code? >> I did all of the formatting. >> Oh, great. Okay. Yeah. Yeah. So, so basically um you know I've also made slides with with with cloud code and >> basically it you know it's able to make the this table it's able to set the formatting on the on the title. It's able to put that random blue line there. But what you can do is you can combine the outputs of this model with other auxiliary rewards

to sort of tell it how good the code was that it wrote. So here this is like extremely functional, but if we wanted to really optimize for aesthetics, we could combine not only the the you know the code execution and actually being able to make the slides and they're structurally relevant, but also some sort of reward model that can look at the the the output of the slides. And it's been trained on human preferences of what aesthetically pleasing slides look like and what, you know, ugly slides look like. And you can combine those rewards and jointly optimize for both writing, you know, the functional slides and then also making them look pretty.

>> Right. Yeah. >> Right. Right. Right. Well, this is a great segue. >> Yeah. >> Into your work. Um, you know, talk tell us a little bit about pre-training, post-training. What do those words mean? >> Yeah. >> Um, and I know you are the world's best expert on on one of those. So, we we'll dig into that. >> That's that's very flattering. Um but but yeah I mean you know sort of the in in language modeling I think the two big buckets to talk about are are pre-training and post- training.

Pre-training is sort of this massive uh training effort where you take internet scale data you know trillions and trillions of tokens. You throw a ton of compute at it through this you know architecture the transformer and you train a neural net to get really good at sort of learning patterns in language. And you know the thing that falls out of this is some form of intelligence. So essentially what what pre-training is is is is this idea of compression where you're able to actually take you know all of human knowledge i.e. the internet and put it into a set of weights that actually understands you know the patterns in language how to think about things whatnot. The problem is once you have this pre-trained model which by the way takes like orders of magnitude more compute than post- training you actually need to go and align this thing. So it's just next sequence next token prediction. So if you write a sentence like you know um who should I invite to dinner and it starts to basically say like a bunch of random names or something like that. That makes no sense because like it you really the model should be like oh I have no idea who's on your invite list or like who you know please tell me who these people are. So post- training is actually the the process of taking this model and telling it what good and bad outputs look like and you actually get a model that you know learns a chat format where there's a user message and an assistant message that responds to you. Um you learn you learn you know safety guidelines. So if a you know if a sort of user asks how do I make like a weapon or a bomb or something like that you can actually tell the model hey don't tell these people how to to go and make these these harmful weapons and sort of like in both of these these cases what's what's scarce is is is data. So um you know I have this slide here which is like very dense um you know you you guys probably don't need to look at too closely but um essentially what what pre-training does is it's just optimizing um for for for loss right so how do you get really good at predicting that next token um but what happened is you know the you know we talked about the chinchilla scaling laws you're scaling model size and you're scaling the data that you're trading these models on we've just ran out of data like there's only so much data on the And we're sort of I know the other side later talking about this, but we sort of have approached uh the frontier on like what data is available to these labs. Um and like sort of the labs are the only ones that can sort of do this level of of of of training because it requires so much compute and so much data. It's a huge capex um you know uh requirement.

Um and then you know I think on the post-training side uh you you sort of have all these different methods for uh you know training models from supervised approaches uh like SFT to you know this preference tuning RLHF to RLVR which we'll we'll we'll talk a bit more about [snorts] >> there's a really good uh article actually in

the readings if you guys read Karpathi's write up on RLVR in the readings that talks about what happened in 2025 you know RLVR really came to prominence in in 2025.

>> Um, we'll get into that in a second, but actually if you if you just click one slide forward. >> Oh, actually here. Can I >> Yeah. Yeah. >> I had a question on data for you. So, >> yeah, >> there's a lot of uh Oh, we spoke about this. >> Data. >> Yeah. >> Running out of data. This is the point you were making. Mhm. >> Um, you know, we had Ali here two weeks ago and he spoke about actually most of the data beyond this frontier is going to be AI generated. A lot of tokens on the world will be just AI generated given the volume of which they're being generated.

>> Yeah. >> Talk about that for the second like what is the what is the frontier of data? Where do we get more data from here on the model that will be trained in let's say 20 2030? What is the input to that and and where do we get it? >> Proprietary public. >> Yeah. So, so I think I think you know there's multiple layers to this question. The first is >> and there's a whole economy, sorry to interrupt you, there's a whole economy of these companies whose full-time job like Scale and Merur and others is to maybe touch on that as well.

>> Yeah, exactly. Yeah. So, so this I think is a is a visualization of of of pre-training data which is really just about like scale. Um you have people who are starting to like buy old libraries with ancient books in them going and scanning these books to get more tokens. You have a lot of investment in synthetic generation. So how can you take uh primary source documents and sort of explode them to multi you know orders of magnitude more tokens and see if you can learn more from that. Um so so I think pre-training you know that that is kind of going to be the methodology. Um really what people are focusing on now I think in pre-training is new architectural advances um to actually make better use of the data that we have today because on principle right like you shouldn't need internet scale data to learn a lot of this stuff so people are trying to be like okay how do we actually use the data better um so there's all these like data wall challenges that you know the frontier labs are are working on now what you're talking about is like RL environments so this is kind of a a different >> type of data this is Hey, let's actually construct the world that the model operates in, have it go and do a bunch of things and then exchange, you know, compute for less high quality data. So basically like we we can use way way way more compute and learn a lot more from a single sample or roll out.

So you know, we were talking about code a little bit and we could talk about this on the RLVR slide. when you're when you make a code environment, it's not like pre-training where you're going and you know training on a codebase and sort of just learning all the tokens in the codebase. You're saying, "Hey, I want you to go and implement this feature." You actually have the model try it >> hundreds or thousands of times. >> You have a way of checking if the model actually did the correct thing. Right?

So that's that verifiable reward and you get this distribution of rewards because sometimes the model will go and do it correctly, sometimes it will do it wrong. And you're actually able to learn way more from that type of training than you are from pre-training alone. So just next token prediction. >> Fascinating. >> Yeah. >> And um maybe the same question for eval. >> Yeah, I think you had something on evals, but like talk to us about eval.

Why is um why are eval important? >> Why do labs guard their eval? You know, there's a lot of chatter about this being the most >> um the the most protected asset. >> Tell us all about it. And as you you know, you referenced this being the >> hairy job to be done. Why is that the case? So I think like you know as we as we start to train these these models on um essentially reward functions what becomes uh sort of the most important thing is actually knowing what good and bad look like. So eval are a way of of benchmarking your your your model and and sort of you know given a certain task understanding how the model acts.

Um the reason why you know eval are so important to the labs is because it eval set the road map. So if you know if we want to go and train a really really good code model um basically Sweetbench I think was was kind of the the eval that sort of started the whole you know co code model race and that was because people had something that they could sort of optimize towards in terms of like what does useful coding look like now threebench I think is a very flawed eval and there's been a lot of new evals that have come out since that are that are much better but that's kind of the whole point is like whatever hill you want to climb you first define it with an eval Then RL is kind of this like eval maxing machine. So you go and create a you know a training pipeline that looks very much like your eval. Um obviously different data because you don't want to overfit directly to those eval data points and then you just climb that hill and then it's on to the next eval.

>> So this is also particularly important when it comes to enterprises. Enterprises h in internally have their own idea of what good and bad looks like. Right? good and bad is not the same across you know like a JP Morgan and a Goldman Sachs right like they they have different standards they have different ways of operating so they will have their own EVs and you sort of get this like tiered effect where you know there's these EVs that the model labs optimize towards and then there's these EVLs that the the enterprises optimize towards and we're actually that layer that specialization layer applied compute to sort of help enterprises optimize to their specific >> ELS >> fascinating >> good segue into applied compute >> yeah yeah So what led you to start applied compute? Um yeah, why better to do it as an independent business than inside of OpenAI where you were before applied compute?

>> Uh and what do you guys do? >> Yeah. So so um you know like I mentioned we started applied compute a little bit over over a year ago. Um and it was you know I started with my co-founders uh Rhythm Rhythm and Lyndon who were actually both students here at Stanford. Um we were also all at OpenAI together. Um funny funny story is like when I joined um you know Sam was basically like who's the smartest person you know that was rhythm couple months later he asked rhythm the same thing that was Lynden and that's how we all sort of ended up there um but we really started applied compute uh based on this core idea that the future is is very specific to enterprises where you'll have these general models which are workhorse models um but actually going and specializing them towards individual enterprises needs is actually going to be how people differentiate. So general models sort of set the floor but in order to set the ceiling you need to go and build train models create these specialized systems in order to differentiate yourself from all your competitors.

>> So an [clears throat] example here is you know Door Dash is a customer of ours. Um I'm sure you know a ton of people order it all the time. I'm I'm very guilty of of ordering it like way more than I should but um >> it's all a part of the RL environment. Right. >> I know exactly. we were just testing the testing the product. But um so

one of one of the the sort of tasks that we worked on Door Dash with and you know I'm picking this one because it's very practical. It kind of shows what what what what we do. Um Door Dash onboards like 100,000 plus merchants every year to their platform. And when these these merchants come to the platform, they basically supply a bunch of unstructured information about their business, including menus and menu extraction.

Actually being able to go from um you know images like this to a Door Dash storefront is actually a really really hard task because >> I see digitizing that. >> Exactly. Door Dash has these this very specific style guide for how modifiers are supposed to be attached on top of items. you know, what you [clears throat] can mix and match, what's an add-on versus like a, you know, special ingredient, things like that. And when we tried using the general models on this, they just weren't able to sort of like do that task. So instead of and we you know we tried prompting and all this sort of stuff >> what actually ended up being the solution is you could take you know outputs of of our model you could have humans go and correct those those menus and understand the delta and then we could basically during training have a model uh have a model's output be checked against the ground truth and essentially we had a way of uh you know quantifying the the loss or the you know the reward like how the error rate essentially and we were able to just optimize directly against reducing error rate. So this is a very clear example of how like a company just needs to go and define what good and bad looks like and you actually don't need to do prompting or any of sort of this stuff. You can just directly optimize towards like the outcomes that you want.

>> Fascinating. Can I ask you a followup on this? Um >> so >> you know the prior gen let's say before transformers this would be a problem that an OCR model would have would would have been applied to like a vision model of some kind. >> Yeah. is the part where you guys come in optimizing the specific problem and this is not using a vision model using a transform model. >> This is using a VLM. Yeah. So, so it's like a a vision model with a you know transformer architecture. Yeah.

>> Fascinating. Um why would Door Dash and and sorry to ask you a hard question on the spot which was offcurriculum. No. >> Um why would why would you guys specialize an existing model when maybe there's a chance that I'm making it up. GPT7 >> might be out of the box much better. >> Yeah. >> Are you incentivized to just wait >> for the next series of models or should should you work with applied compute to >> Yeah. No, no, it's branch out.

>> It's a great question. I'm glad you asked me. Um I think like what what people, you know, often don't realize is that enterprises care about being at the frontier at any point in time. So GPT17 is going to be, you know, quite a long time from here. I think you know by the time we have ASI or this this this model that kind of controls everything which uh actually I I I don't believe that's going to happen. I I I think the the world is just very fragmented place and you know if you just look at where the data is it's it's it's kind of you know dispersed >> but yeah the time to value is just way the ROI on being able to train your own models today with uh you know way less compute like RL has become very very data efficient. um you need to use an order of magnitude less compute than pre-training or these other types of of training and you're able to optimize performance um you know like way more than you were in SFT and RHF makes it a lot more appealing to be able to train these

models >> like the order of magnitude investment that goes into post- training uh pre-training versus post-training could you give us an estimate for that so let's say there was a $100 budget for training how much of that would be pre-training how much of that would be post training >> yeah yeah so I think I I think I looked up on on the way here. Um, Deepseek V3 uh was trained on about like 2.4 2.5 million H800 hours. Um, the RL training, so the the training that led to DeepSeek R1 was trained on like about 150K. So, >> comes out to about 5% of the training compute that's needed for for pre-training. But >> that's it.

>> That's it. But I think what's really interesting is that trend is is starting to to to change where people are pre-training these models, but then they're also doing data centerwide, multi-data centerwide RL runs because >> you have these scaling laws which I think I had a photo of Jensen and the the three scaling laws. There's pre-training scaling, then there's post-training scaling, and then there's test time scaling. Test time scaling is inference. But post-training scaling you can actually uh you know massively increase the batch size of of each you know each of these training steps and you get you know a lot you know a lot better performance. You can have these models do a lot more reasoning when they're attempting these tasks. So >> I think the trend you're actually seeing is that the compute spent on RL is actually increasing quite quite heavily.

>> Got it. So it's 5% today but you expect it to go up as >> Oh it is going up. Okay [clears throat] as relative percent of the total training budget. >> Yeah. So, so it I you know I don't know the latest uh stats on like a mythos or like a 55 um but you know up until kind of uh basically when when we were scaling out >> 01 03 you know codeex uh deep research we basically saw like more compute you put into RL the better performance you get. So these things stack >> makes sense.

>> Yeah. >> Um maybe a couple other examples. I found that to be very very helpful. Are there other examples of domains um outside of for example converting this menu to to a door dash that you could share with us of where this is a a particularly useful application? >> Yeah. Yeah. So I mean we we were talking about coding before. We recently just put a model in production with with cognition windsurf. Um basically the idea is when you're writing code and you write you know you save your file. How cool would it be if you had a model that kind of ran sub two seconds, checked what code you wrote, and then told you if there was a bug in it or not. So, you know, this is not something you can get with a general model because there's this paro frontier of performance, cost, and latency. So, if you take a really small model, you kind of post train the heck out of it on getting really good at this task of of bug catching, you're able to get the benefits of cost and latency and the performance of some of these larger um sort of bigger models.

>> Got it. And so the value ad for your for for for Cognition and Vinsurf is that they're extending their product suite from just writing code but also now to testing and and and bug uh >> Exactly. Yeah. And and I think this gets into this really interesting idea of sort of model harness context code development which is like you know you >> you never really can focus on just one layer. um you know a lot of these application layer companies are um doing a ton of innovation on the harness and that's actually how they're able to squeeze uh value out of these these models especially when it you know relates to like the service that they're providing and then context is just like often times if you don't have access to the right data you won't know the right thing to do. So being able to plug into all of these different data sources inside of companies that's also extremely extremely important.

>> Fascinating. So this is in this case in the case of cognition it is it might be a true competitive advantage for them to expand their product frontier. >> Exactly. Yeah. Yeah. I mean I I I think like you're seeing uh people start to push the frontier on what these models can do and and it's usually an ensemble of models right like general models extremely powerful really good orchestrators but you know fast sub agents or agents trained on proprietary data that's out of distribution of these models like like something like this um those can be orchestrated with the general models and to create a really powerful system so like I think today actually um you know ramp ramp labs the the you know the corporate card company which we're actually really good friends with them and you know know a lot of folks there they like trained this RL model um to basically do fast search inside of your spreadsheets >> and that's a way you can actually go and improve the product experience >> fascinating >> we'll we'll change topics Yosh and talk about a bunch of these emerging model training techniques uh that we've been hearing about and maybe you can decompose those for us >> love to yeah >> we'll start with continual learning >> what is it a lot of smart people we know talk about that as the next frontier including you. Break it down for us.

>> Yeah, so continual learning is is really the kind of what I what what I mentioned before which is like how do you learn from extremely sparse rewards, right? So um you know like if you have a system deployed in production um how are you actually able to understand how that that AI model is being used? understand, you know, the downstream consequences of of of its actions and then use that to update the system so it gets better over time. So, you know, what I what I have here is is kind of like two examples of of uh >> yeah, two examples of what what I think like these this is starting to look like. And to be clear, I think this is going to be a very gradual thing. So, um you know, a lot of continual learning is blocked on just having access to the right data. So when you go and deploy an agent in production, like are you putting it in front of the right people to get feedback? Are you actually deeply understanding all of the context necessary to know um what good and bad looks like? That's like just a data access problem. So I I think this is going to look like kind of a slow gradual roll out rather than like oh there's some like uh you know extremely valuable insight that that someone comes up with. But a couple of examples of of how how you're seeing this today is you know cursor they have this model called composer which is essentially their um their own coding model trained on their coding data uh on top of an open source model. And what they did was really cool. They basically took this model um had people use it in production were able to capture a bunch of that telemetry take steps online. And so take a training step based off of like some implicit rewards that they um that they sort of calculate. So basically they would look at like did the user accept this code suggestion or did they revert this code or >> realistic for success.

>> Exactly. Yeah. And and they kind of optimized towards it and they were able to see improvement by doing this sort of like online training where you're collecting data taking a train step collecting data training to trade up. Then this is something that we've been doing >> actually just to follow up on cursor [clears throat] the x-axis here yash is steps >> could you convert that for us into time like order of magnitude how much time did cursor have to invest in improving cursor's performance >> yeah it's like days or weeks or hours or >> it's a good question we're I think we're talking about days or days or weeks here um and then I think couple hours per step >> got it >> I don't know the exact uh terms but um one one interesting thing here is like in in RLVR when you're training offline, you have this like replayable environment where you're um sort of like it's the same task. It has a a defined reward and then you're rolling out that sample, you know, hundreds or thousands of times in parallel. You can't do that in production, right? Because people are using this, you know, they have dynamic

environments or whatnot. So what they really experimented with was could we just take a massive batch so like many many many conversations d noiseise the gradient that way and then take a step and hopefully that'll be like directionally the the way to improve the model. So yeah so that's this is like um I think like hours per step and then each of those steps is quite quite quite big or a lot of samples in them. Mhm.

>> Um on the right here is is something we actually have been been working on which is um this this we we this idea of context base which is like can you actually go and use agents expend compute offline to be able to go and analyze a bunch of documents analyze a bunch of past traces that humans uh have had with agents and extract learnings from that that will improve performance um downstream. So one thing that we were able to see is like yeah we were able to see uh at different reasoning efforts a massive increase in performance um while sort of using the same amount of of of of tokens. So yeah so these are just kind of two examples but I I I think like high level you're going to see innovations at weight updates context and the harness itself to actually be able to capture this information.

>> Fascinating. >> Yeah. More to come on this. Um, >> second topic is nontransformer models. >> Yeah, >> a lot of talk in the class about transformers not being a very efficient architecture. Takes up a lot of power. >> Uh, we had Ali Gocei, he said that well look uh flying like the airplanes is far less efficient than like the boards do. Turns out we're heavy a heavy specy heavier than birds. uh is the transformer like that in that it it while not efficient will be the dominant way because the world's infrastructure has morphed and moved along that way or do you think there's a there's a shot for a non-transformer architecture like like the Mamba architecture or one other to be a dominant player in uh AI models going forward? Yeah, I think I think my honest take is like scaling transformers is working and as you know there's a very simple recipe to be able to make these things smarter and better and um >> you know probably more likely that the AI will tell us what the better architecture is if we just continue scaling it up then try to come up with one ourselves. um you know if there were if there was kind of a a wall in terms of you know what this architecture would allow us to do then I think you know we we we would see some um innovation there and there there certainly is really cool research happening but my opinion is just like concentrate on scaling scaling transformers >> very smart people on the other side of this debate as you know Ilia Jan lun others >> going at it uh if you if you can share do you know what the core insight is that that leads them to believe the other side of the debate and disagree with you Yeah. So, I think I think like um you know the core insights are you kind of like what I said before is you don't need pre-training levels of of of of data to be able to actually learn the underlying representations of language.

I think Yan Mukun talks about this a lot where it's like you know humans don't need that therefore the architecture that we developed shouldn't shouldn't um require that. Um so I think that's the underlying argument. It's just like first principles like >> you shouldn't need it. Therefore, there must be a better solution out there. >> But I think like kind of to your point, the investments we're making in our compute scale outs um >> you know, I think there's people who are actually optimizing for the architecture directly in the chips.

>> That is kind of I think a a big ship to turn and so far what we've seen from the labs which are going to I think control a lot of this uh this buildout is is just investing more in the transformer architecture. They're definitely doing research on like new new stuff, but um I think it's it's all experimental and you know, >> yeah, >> could

work. >> Could work. What a time because you know >> big big large sums of money are going into these these techniques. So, we'll see. We'll see how it shakes out.

>> Yeah. >> Um rapid fire. Last five minutes. >> Um you know, a lot of folks in the class deciding where to build, what part of the stack to build, what to start. >> If you were not if you had if you were not building applied compute today, >> yeah, >> what would you be doing? what's your next best idea? >> Yeah. So, been thinking a lot about this cuz I think like what one thing that we've been running into at at Applied Compute and I know a lot of other AI companies are running to into is like scarcity of compute.

>> Mhm. Um, like I think uh the the the demand is just far outpacing the supply for for compute and I think there's going to be massive innovations in like the energy sources needed to power this compute and then also making more efficient chips themselves. >> So, you know, not that I have a background in like hardware or or chip design or anything like that, but I think we could be making way better hardware um to sort of optimize the code of training and um >> and and chip design. So, I I would probably like look into hardware.

>> Okay. Thank you. The long short game, uh, pick a business, a product, a person >> that you like a lot, that you're excited about. >> Yeah. >> Uh, and the counterfactual, some something that is there's more, uh, hype than there's reality. >> Yeah. Yeah. So, I think I think, you know, um, kind of kind of goes along with my last answer like compute and the chip providers like Nvidia, I'm very long long them. I think they're gonna continue to win. They're going to continue to supply all of the the um you know all the labs. And actually it it is interesting though like once once you kind of look at the compute economics of of of Nvidia you know they take like a 75% margin on top of their chips you have these labs spending hundreds of billions of dollars. Um you know question is like hey like maybe we take a couple hundred billion dollars and invest that in in actual like our own chip design. Then we could do all the co-development of uh you know model training architecture and chips internally and you know maybe our chips are like 80% as effective but we'll just make like you know 1 you know two or whatever it is x more. Um so I I think like that that is generally I think chips and you know Nvidia is going to be like the the the sort of leader here. I think that's valuable, but I do think there's also a risk of like, hey, the labs are the biggest customers are maybe just going to in-house do this stuff themselves, but it's very hard.

>> Um, one thing I'm a little less bullish on, I think, is like the data the data market is just really tough. Um so you know if you think about an RL task um you have basically like models that are not very smart you train them to be very smart at particular tasks and then it becomes that much harder to go and create new tasks that you can actually hill climb on. Right. So when we were >> because because the problem is much harder.

>> Exactly. you're you're you're sort of getting squeezed where it's like, okay, yeah, I'm selling to my customer and I'm improving their model, but I'm also making it harder for myself because the next time they come to me and say, hey, I want you to to go and build this task. I have to spend way more money. It's going to take a lot longer. All that sort of stuff. And then I also think the models are just getting really really good. So, um, you're I think you're starting to see a lot of synthetic data generation >> because if you think about an RL task, a lot of times what it is is exploiting some sort of generator verifier gap where like for code, right? Um, you hold out the

unit tests, you have the model attempt the task and then you run the tests against the the model's output.

Um, that's not something that you really need a human to do. You can the smarter your models get, the better pipelines you can build around synthetic data. So, I'm not like like I think data has been a thing that people have been like, "Oh, it's going to die every couple of years." And it hasn't. But I do think it's going to change. And the best the best founders in the data market um kind of are just really good at pivoting and sort of like, you know, the ne what's the next wave? Robotics data, egocentric data, like put on a you know, put on a GoPro and like collect a bunch of stuff. So I think like >> RL environments today is going to be tough, but it'll get you know they'll go on to the next wave of things.

>> Awesome. Final question. Um, your favorite AI product or favorite AI modality? >> Yeah. Oh, this is a good one. I I love GPT uh or sorry, image GPT2. >> Image DO. >> Image 2. Image Duo. Yeah. I you know, when it when it came out um I was having a ton of fun with it. It It's like >> I think it's massively like for people who can't do design like me or who want to like look at things visually. Like a lot of times I'll just take some paper or something and like drop it into uh image 2 and it gives me this nice visual sort of like walkthrough of how things work. So um that's probably been my favorite AI product.

>> I'll show you an artifact. The most beautiful slide on this uh on this presentation was an image tool representation. When I fed it the course syllabus, it was like how about this >> this one? >> Oh wow. >> And the signature for those who you can tell is this one right here. That's the images to watermark. >> Oh, really? >> Oh, interesting. >> Um, anyway, awesome. Well, thank you so much for being here. >> No, thank you.

>> We'll uh continue this conversation and see how these bets play out. >> Let's do it. Awesome. Thanks for having me.