/mlst

29.4% ARC-AGI-2 (TOP SCORE!) - Jeremy Berman

September 27, 2025



Machine Learning Street Talk

Podcast Transcript

Revision: b0582d47

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

Hook & Introduction 3

Background & ARC v1 Solution 5

ARC v2: From Python to Natural Language Evolution 7

Continual Learning & Catastrophic Forgetting 11

Turing Completeness, SGD & The Symbolic Gap 14

Efficiency, Search & the Leaderboard Landscape 20

Knowledge Trees vs Knowledge Webs 24

Creativity, Invention Circuits & Epistemic Trees 29

Spelke Priors, Physical Grounding & Closing 33

References 36

Overview

Jeremy Berman took the top spot on the ARC-AGI v2 public leaderboard with a score of about 30% — using an approach that trades code for natural language. Where his first attempt evolved Python programs to solve abstract reasoning puzzles, this version evolves plain English descriptions of transformation rules, then uses a strong thinking model as a checker agent to verify them against training examples. The shift to natural language is the interesting part: English can describe ARC v2 tasks in five bullet points where Python takes dozens of lines, and the higher expressiveness lets the system explore solution spaces that rigid code simply cannot reach.

The conversation goes deep on why this matters for intelligence research. Berman and Tim work through the relationship between reinforcement learning and genuine reasoning — whether RL can replace the messy pretrained knowledge web with a clean deductive tree, what catastrophic forgetting really blocks, and why the meta-skill of reasoning (the ability to create new skills) is the actual target for AGI. Berman makes a sharp distinction between knowledge that is memorized and knowledge that is deduced, arguing that pretraining treats everything as an interconnected web when what we actually need is causal structure.

They also dig into composability (freezing expert layers, Docker-for-models), whether neural networks can ever run Turing-complete algorithms the way biological brains seem to, and what it would take to build an invention circuit — the machinery for genuine creative synthesis rather than pattern recombination. The discussion lands on a shared framework where intelligence is the efficiency of building epistemic trees, reasoning is constructing them, and understanding is possessing them.

Hook & Introduction

Jeremy Berman 00:00:00

You can describe every single ARC v2 task in 10 bullet points of plain English, most of them in 5 bullet points. And I think that this actually gets to the heart of ARC. Right? Everything is quite simple. It's not very hard. And And I think this is also how we do it too. Right? Like we when we look at these ARC graphs, we're coming up with these bullet points in our head and we're, you know, checking them. Okay. This was right. This was right. And Python doesn't have these features. It's just not as expressive as natural language.

Tim Scarfe 00:00:24

MLST is sponsored by Cyber Fund. Link in the description.

Jeremy Berman 00:00:28

I get actually even more fundamentally like the ideal system would be we have a set of data. Our language model is bad at a certain thing. We can just give it this data and then all of a sudden it keeps all of its knowledge and then also gets really good at this new thing. We we are not there yet and that to me is like a fundamental, missing part. Really what you want is more expressive program. And so that's why I switched from Python to English, which is a much more expressive program. You can language you can always teach a language model skill, Right? But it's the meta skill. It's the skill to create the skills that is AGI. And to me, that's reasoning. Like, reasoning is that meta skill. And so, to put it another way, I think if you fundamentally learn the skill of reasoning, you should be able to then, apply that skill to learn all the other skills. That is the meta skill. You know, kick whatever weights out you need to, align the model to reason, and then from there, you have a foundation from which you can actually build general intelligence.

Tim Scarfe 00:01:25

Okay, folks. Hot off the press. Many of you would have seen last week that Jeremy Berman, who is a research scientist at Reflection AI, is now the winner of the ARC-AGI v2 leaderboard, the public version of the leaderboard. He's using an evolutionary approach. Now, remember last year in December, he published a similar approach. Generating Python functions and then refining those functions in a kind of iterative loop. His new architecture is generating descriptions of algorithms rather than code. And iteratively, in an evolutionary sense, refining those ones and discarding the ones that don't work. He's

now at the top of the leaderboard. It's a really really cool and elegant algorithm. And by the way, he works for reflection AI. So he's doing reinforcement learning with verifiable feedback. And he's trying to address the biggest gap in AI at the moment, which is that we want systems that can synthesize new knowledge and new understanding. Current systems just get trained with a whole bunch of data and they only know what they've been trained on. They can't kind of think outside the box by creatively synthesizing new knowledge. Prolific are really focused on the contributions of human data in AI. And the reason this is important is actually the dirty secret of Silicon Valley. The extent to which human data is used to evaluate and fine tune AI models. The reason for that, as we discuss in today's show, is that current AI does not understand the world in a grounded way. It doesn't have a deep abstract understanding of the world, which is why the only way that we can make AI work effectively is by grounding the generation and supervising the training of AI models with human data. Prolific are putting together a report on how human data is being used in AI systems and they need volunteers. You can just go and fill out this form to help them produce this report and you will get privileged access to see the report before anyone else. The link is in the description. Oh, there was an amazing part in, I think it was in your first paper where you said, a parrot that lives in a courthouse will regurgitate more correct statements than a parrot that lives in a madhouse.

Jeremy Berman 00:03:40

Thank you. Thank you. My sister, who doesn't know anything about language models or AI, she pointed that out and said that was a great line. So at least I have that.[1]

Tim Scarfe 00:03:51

I've already used it. I I credited you that, I'll be using that quite a lot. Well, Jeremy, it's amazing to have you on MLST. I've wanted to to have you on ever since you released your first blog post. You know, it was December. I I was at NeurIPS at the time. And at at the time, you you actually got the highest score on the the public ARC v1 leaderboard just before the famous o 3 launch. Do you remember when when they did this ridiculous $200 per task thing and and they knocked you off the board? But, Jeremy, can can you just tell the audience a little bit about yourself and and and and maybe we should start with your with your first ARC solution?

[1] Jeremy Berman's ARC-AGI v1 Blog Post — Blog Post

Background & ARC v1 Solution

Jeremy Berman 00:04:30

Yeah. Sure. So I actually have only been working in research for about 8 months. Before that, I had a company, right out of college. I was got into Y Combinator, and so I've been running a company, for the last 4 and a half years as CTO. And I've always been very interested in reasoning in the brain. And I actually picked up Jeff Hawkins' book, A Thousand Brains, and I read that. And at the same time, I was kind of coming into language models, and something just clicked inside of me and I just knew I had to be working on this. And I believe that artificial general intelligence will be the most important invention of hopefully my lifetime. So I decided to drop everything. Stepped down as CTO. Company is still going well, so it was a difficult decision. And I actually had gotten in touch with Mike and Francois because I thought ARC-AGI was such an elegant way of describing the problems with current language models and their difference between the human brain. And so I just kind of dug in. That was my first research project independently. And, yeah, I ended up getting the top score on that, and that was really great. And after that, I got recruited into Francois and Mike's AGI lab, NDEA, where I was working on program synthesis. And as you described earlier, over time, I've become more convinced that language modeling with reinforcement learning will yield generalization far beyond what we see today. And so I decided to move to a company that was focused purely on language models, and that's where I am now. So I'm currently working on reasoning and post training at Reflection, which is we're building frontier foundation models.[2]

Tim Scarfe 00:06:16

Very cool. Maybe we should save that bit for tiny bit later. But 1 of the take home messages in your well, no, I mean, it's super interesting. And 1 of the take home messages on your new approach is that rather than producing explicit programs, you are evolving descriptions of programs. And Francois is a neurosymbolic guy. He thinks that we need to have a symbolic substrate where we, you know, represent the the kinds of problems that that we can do. And we need to do this kind of compositional form of of of intelligence. So we need to kind of be working in the symbolic layer, but perhaps guided by, you know, deep learning models. But maybe we we should get to that in in a minute. So in your in your first solution, it was an evolutionary approach. It was using Sonnet 3.5, and you had about 4 iterations, I think. And and, essentially, you you know, you were working on the ARC challenge, you were producing these programs through evolution. Maybe just for folks that don't know about the ARC challenge as well, could you intro-

[2] NDEA — Company

duce that and and get into your solution?[3]

Jeremy Berman 00:07:20

Sure. Yeah. So the ARC challenge is kind of like an IQ test for machines. It's a set of input output grids, and the whole point is to be able to figure out to transform input grids into output grids given a common transformation rule. And so, what's interesting is these are really easy for humans. Right? The average human gets around 75% accuracy on ARC v1, and at the time, the best language models, GPT 4, Sonnet 3.5, was getting maybe 5%. And so, yeah, basically, the idea is you have a few training examples and then you're you're you're trying to extrapolate the transformation rule on the final test example. And so I approached this. I was actually inspired by Ryan Greenblatt who had a solution earlier, which was to generate a ton of Python programs that would encapsulate the transformation role. And Python programs are great because they're deterministic, and you can pretty quickly check whether or not the Python program works or not, which is really cheap, so it's cheap to verify. And you can be relatively sure if the Python program works on all of the training examples, that it'll work on the test example. So, I started with his approach, but then I noticed that, you know, the language models actually struggled on first attempts. Even if you ask the language model a thousand times to generate Python programs, they were always off by small amounts on easy tasks, which I thought, you know, presumably, it's in their distribution. They should be able to solve this. So, what I found is that actually by taking the top, programs, the top performing programs, and then running that in a revision loop, so asking, Sonnet 3.5, hey, here's what you got wrong, here are the cells you got wrong, here's your original Python program, improve it, that started to really work well. And then I thought, well, why not just increase the depth, right? Why not ask it 10 times to revise until I'm happy with the solution, passes some sort of accuracy threshold? So that's kind of how I was inspired by it. And I didn't think of it as evolutionary at first. I was just thinking about broadly what would work. And over time, I kind of understood there was something a bit deeper going on here, which is that evolving solutions is a powerful technique generally. And I think it's actually going to play a role, you know, in future technologies. But, yeah, that's generally guided by just intuition.

[3] Getting 50% on ARC-AGI with GPT-4o — Blog Post

Tim Scarfe 00:09:38

Yeah. I I had Ryan Greenblatt on the show. I'm a huge fan of of his. He's a very, very smart guy. And I asked him a similar question because he he did this iteration, right, where where you have a certain depth of of iterations. And I guess 1 approach is that you have a, like, a a shallow method. Right? So you just tried 200 different variations. And in your in your blog post, you kind of said there's a there's a Goldilocks zone where you want to have a certain number of tries of, you know, different, you know, different variations of things. But you also want to be able to refine your solution because that allows you to do this kind of composition. And composition is very, very important for things for problems that require iteration. And indeed, the second version of the ARC challenge, I I think the tasks were selected so that they had at least a couple of iterations in them, which meant that they needed to have this depth. Can you talk about that

ARC v2: From Python to Natural Language Evolution ●

Tim Scarfe 00:10:33

trade off?

Jeremy Berman 00:10:33

First of all, the second the ARC v2 is in a sense fundamentally different than ARC v1 because of what you're describing. They're compositional. There are many rules that you have to go through, and this is partially why I found that my solution on ARC v1 did not perform well. Yeah. So there's a constant trade off between how deep you go, how many revisions you take, and then how how broad you start out. The problem with, you know, going deep and not going so broad is there are some edge solutions where you'll never you'll never get to. Right? But then, of course, most solutions end up being somewhere within the bounds of your first broad attempt. So that's generally the trade off, and, the trade off is different for ARC v1 or ARC v2. Interestingly, I found that for ARC v2, it was more important to be broad. And I think this is surprising to a lot of people, and partially this is because the models now think, and that's great. So the models actually do a lot of the deep revision for you in their thinking block. And this is a fundamental change from when ARC v1 existed and when I, you know, just started out in the field. I think I'm a bit embarrassed by a lot of the things that I wrote in my first post because it was 2 weeks before o 1 was released, and everything about o 1 changed how I think about these things, which is before, you could simulate, emulate thinking to to the quote you you described with the the parrot, the stochastic parrot. I think before you had reinforcement learning over before you

actually taught the machines, the language models to think with reinforcement learning, you were almost doing this stochastic guessing. That was not a very efficient revision loop, basically internal revision loop, and so you needed to artificially create that revision loop with code. But in v 2, I was able to use a very powerful thinking model, which has a lot of the deep revisions for it. So I found it was best to increase entropy, let it explore the space itself, and then I'll add a revision loop on top of that. But the revision loop, is less important in v 2.

Tim Scarfe 00:12:35

Yeah. So the first 1 was Sonnet 3.5. So that didn't have this thinking thing built into it. So in your prompt, you told it to think step by step. And now I I think you're inspired by Ryan Greenblatt's Greenblatt's prompt. Right? So you had a a whole bunch of ways in there for representing the board state, and you said, want you to think now, and I want you to know, that this is an abstract reasoning challenge and and I want you to think from first principles. And they would kind of go through that and then they'll give you the answer. But you're saying that on the the the RL trained models, it was significantly better at doing that.

Jeremy Berman 00:13:05

Right. Exactly. You could think of RL trained models as having inbuilt revision loops. I you know, they are trained to explore the space in a deep way, thinking generally for themselves in a way that so you really don't, know, thinking you don't need to prompt thinking models to think step by step. They already do it.[4]

Tim Scarfe 00:13:27

Yeah. I mean, I wanted to challenge you on this a tiny bit. Right? So you kind of said in in your I think it was in the second version of of the blog post that you just released last week that at the moment, the models can do domain specific thinking. So they can do math thinking and they can do code thinking. And what we want to do is is imbue, like, the core machinations of thinking into these models. And I'm a little bit skeptical. I feel that these models, because they're not Turing complete, because they're not symbolic, you know, similar to what Francois believes that I'm sure you read that LLM biology paper as well. They were talking about these circuits that we can find in papers that do things like multiplication and and and addition. And what we what we saw was that they are quite patterned. They're quite templated. They're not thinking in a very general sense. And my suspicion is it will always be that way because the models don't have semantics. They're non symbolic and and so on. Do you think we could ever make them truly think in a general way?

[4] On the Biology of a Large Language Model — Paper

Jeremy Berman 00:14:24

Yeah. I think fun I think fundamentally taking a step back, the fact that our brains can do it and our brains are generally running similar algorithms, to me, this means that we will eventually be able to, inject general reasoning into the language models. I don't think there's a fundamental reason why, neural networks can't behave like biological neural networks. So that's, I guess, the higher the higher level point. And then, zooming in, right now, you know, the models are as bad as they're ever going to be. There, there's generally more compute going into pretraining than there is reinforcement learning, and of the compute going into reinforcement learning, a subset is going into specific general reasoning. And so I think that over time, you're going to see the models get better and better at general reasoning. But I guess a question I would have for you is do you think there's a fundamental difference between the way the brain works where there's some sort of symbolic nature to the brain and and it's not possible to inject that type of nature into an artificial network?

Tim Scarfe 00:15:24

Yes. Yeah. I mean, you mentioned Jeff Hawkins. I I interviewed Jeff. He's absolutely amazing. And of course, his HTM algorithm is computationally stronger than a neural network. It's it's Turing complete. And our brains, even though they are finite, they run a Turing complete algorithm, which means our brains know how to expand their memory. Right? Our memory. We can go and write things on a whiteboard and we can go and, you know, get another notebook. And that is a special type of algorithm which is not traversable with stochastic gradient descent. So, you know, the the the rough argument is, yes, there is there is a difference there. And I also wanted to touch on this, you know, RL with verifiable rewards thing, which is that we do that at training time. I'm very excited in the future about an active inference version of that, like an agentic version where we're actually doing this kind of transductive active fine tuning in an agential way. Right? So, you know, I I take an action. I get some new information from the environment, and I update my weights. And that would be truly adaptive. That would be intelligent. But what we do now is we do all of this stuff at training time, and the resulting frozen artifact is still an LLM. It still has just a bunch of patterns in there. And I think that while that can uplift reasoning in many ways, I don't think it it has the intelligence. And and according to Chollet, intelligence is simply the ability to search through the space of Turing programs. Right? And I don't think that's what's happening with these LLMs at the moment.

Jeremy Berman 00:16:54

I I think you're generally correct that it's not happening at the moment, but I still think, like, fundamentally, I don't think there is a, like, a fundamental blocker physically for why they won't be able to do it in the future. And it's possible that SGD, right, like stochastic gradient descent, is is an issue fundamentally, and I think we're going to overcome that. I guess what I would say is artificial neural networks, I think, have the structure, capable of, yeah, basically, being as smart in every way as a human brain. And I I subscribe to, Francois' definition of of general intelligence as well.

Tim Scarfe 00:17:31

Yeah. I mean, I I think we mostly agree. I I think I mean, you know, let's look at alpha 0 or mu 0 or something like that. They did this training loop where they were actually updating the, you know, like, the the value network and the policy network. And then it was frozen, and they did some kind of, you know, Monte Carlo tree search. So they were achieving adaptivity through exhaustive search during the actual games. And in an ideal world, we would have this adaptivity that's actually updating the weights. Now I believe the only reason we can't do that at the moment is just computational tractability. Right? We have these huge models. We couldn't possibly have a a dynamically updating model for every single person that's using ChatGPT. It it would it would just be ridiculously slow. But I think you and I agree that if that were possible, that would be an entirely different kind of form of intelligence.

Jeremy Berman 00:18:24

I don't think that's so intractable, actually. I think, my guess is and, you know, NVIDIA just put in a $100,000,000,000 into OpenAI. Open Sam Altman's plan is to produce a gigawatt of, what, of of compute, a week, something like that, I actually don't think with, you know, ever efficient, algorithms that that is, like, crazy far off. I mean, right now, you could buy a GPU, You could have it running in your house, it could be running OSS 1 20 b. Right? And fine tuning is relatively trivial compared to, you know, the entire process for pretraining. I actually think that is a that is totally within the realm of possibilities in the next 10 years. And I think that that actually is is potentially where this goes.

Continual Learning & Catastrophic Forgetting

Tim Scarfe 00:19:12

Yeah. I mean, you know far more about this than I do, but I think the reason why fine tuning is so expensive is, you know, we have this continual learning problem. And when you fine tune a model on OpenAI, they're not just fine tuning it on the data you give them. They you know, just to stop this catastrophic forgetting problem, they presumably have to sample in a bunch of the original training data and maintain the distribution and so on. And and if if they did this for everyone, it would be insane. But I am excited about it just like you are because I I interviewed the architects and I I think they got first place on the on the private version last year. And they were doing this transductive active fine tuning. Right? So they and they actually said, by the way, that this is a a curious oddity with transformers that if you start with a, you know, almost like a virgin 8,000,000,000 transformer, it almost doesn't matter what it knew about before. You could just pretty much start training it from scratch on the ARC challenges. So they did a whole bunch of augmentation and active fine tuning, and they built an intelligent artifact. I mean, intelligence is domain specific as per Chollet, and they actually built the system which was per task adapting and solving the tasks, and they were updating the weights, and it was beautiful. So that was an existence proof of nothing else that this thing could work.

Jeremy Berman 00:20:22

And that was on a Kaggle notebook. Yes. You know, in 10 years, this is gonna be what? Like, the, you know, Apollo mission computer. I think that I think that what you're describing I'm actually not even totally convinced that continual learning is fundamentally the blocker, but I think if it is the fundamental blocker, that's actually incredible because we will solve continual learning. Like, that's something that's physically possible, and and I actually think, like, it's not so far off. Now the the, forgetful issue, that is a much more fundamental issue in my mind. And not just the the fact that you need to, every time you fine tune, you have to have some sort of very elegant, mixture of data that, you know, goes into this fine tuning process so that, there's there's no catastrophic forgetting. This is, I think, actually a fundamental problem. So I the the and and it's a fundamental problem that that, you know, even OpenAI has not solved. Right? And I think Francois is a great example, and I think this is an important example. You know, if you have the perfect weights for a certain problem and then you fine tune that model on more examples of that problem, the weights will start to drift and you will actually drift away from the, from the correct solution. His answer to that is, well, we could make these systems

composable. We can freeze the correct solution, and then we can add on top of that. I think there's something to that. I think, actually, it's possible that there's a research direction where maybe we freeze experts, or may maybe we freeze, layers for a bunch of reasons that isn't possible right now or but people are trying to do that. But I yeah. I I think, fundamentally, compute is not the issue. I think it's this catastrophic forgetfulness.

Tim Scarfe 00:21:57

Yeah. So I'm inclined to agree. I've I've long dreamed about there being a docker for language models. Right? You you know, in docker, you can kind of freeze dry a state of, you know, like, let's say Linux operating system with an application with the security updates. You have these kind of immutable layers. And the composability that we often talk about could actually happen at the architectural level, and we could do dynamic model merging between different layers and and whatnot. That would be very, very exciting. So, know but also, just to come back to what you said before, I've never really heard this before. You're distinguishing, like, forgetting and learning. Right? When when we were talking about, you know, catastrophic forgetting and continual learning. Can you just sketch out that distinction a bit more?

Jeremy Berman 00:22:40

So the way I think about it, you know, you have a neural network, and it has all of these weights inside of it. Right? Anytime you update those weights, you are pushing some weights out. And presumably, you are pushing some correct answers that you've previously, you know, trained, and they are getting pushed out. And the benefit, I think fundamentally, of symbolic systems is that doesn't happen. Symbolic systems are deterministic. When you get the right answer, you can be sure you have the right answer. You stash it away into your library of correct solutions. This is this is the problem with continuous structures, but this is also actually now why I think, you know, it's important to draw from the brain, which is this similar thing happens actually with the brain. I believe the brain is much more composable than neural networks biologically, but I think there's no reason why we can't, you know, we we won't be able to figure this out. Right? Like, again, it could be it's it's as easy as we end up freezing experts. Again, like, the freezing of the layers, I think this is an an underexplored, area, and I think we're going to go through basically this RLS curve. And then I think this is the next S curve, is figuring out how to make language models composable. I get actually, even more fundamentally, like, the ideal system would be we have a set of data. Our language model is bad at a certain thing. We can just give it this data, and then all of a sudden, it keeps all of its knowledge and then also gets really good at this new thing. We we

are not there yet, And that to me is like a fundamental missing part of general intelligence.[5]

Tim Scarfe 00:24:09

Yeah. Completely agree. So it it sounds like we have very, very similar intuitions. And and Charlie talks about this as well. Mean, in his measure of intelligence paper, it was actually about the measure of intelligence. He's never really spoken about the machinations of intelligence. He talks about it just casually. He says, you know, we need the Spelke core knowledge priors and we need to so those are like the basis functions and we need to do library learning and library transfer and we do some kind of like, you know, symbolic compositional process, you know, to adapt to novelty. And he's kind of sketched out the the mechanics of it, but he's never actually formally spoken about it. I assume that's what he's building at at at his company. But there was a famous guy called Jerry Fodor in 1988. He had this connectionism critique. He had this beautiful paper. And he was basically saying that symbolic systems have systematicity and productivity. And systematicity is this compositional thing. It's that, you know, it's the ability to generalize between Mary loves John and Mary loves Jane. Right? So you so you have semantics. You have these kind of like, you know, the the symbolic relations. And they have certain computational properties, like you can do a variable binding and and quantification over potentially infinite domains. Like, we intuitively understand that symbolic things have very interesting properties. And then what we're trying to do is, like, we know neural networks are really good, and we want to somehow graft this capability onto neural networks.

Jeremy Berman 00:25:33

Yes. Yeah. And I actually think neural networks are in some ways a superset of symbolic systems. I think you can generally you should be able to encapsulate a symbolic system with a neural network in the same way I think that you can do the I think you can do the same thing with the brain as well. I think there's there's nothing fundamentally blocking, but of course, once you have have the symbolic system in the neural network, it might catastrophically forget when you fine tune it. Like, guess that's a that might be where we we disagree a bit. I think everything you're describing is totally possible, but then when you're when you're coming to train it again, there's no guarantee that it keeps the same structure.

[5] Connectionism and Cognitive Architecture — Paper

Tim Scarfe 00:26:17

I think it's possible because a neural network is is not too incomplete. So I I think in principle, can't do many of these things, but you can build a controller. Right? So you you could just build a very simple kind of envelope just as you did with your solution. So you you you had a bunch of code and it was doing this in, you know, this basically compositionality in code on top of the neural network substrate. And that gives you many of those things. You know, for example, we we often talk about library learning and library transfer. And I'm not sure if you've seen Eric Pang's solution. I'm speaking to him in Hong Kong actually in a couple of weeks. But he rather than the DreamCoder approach where they do this explicit library learning, He was doing it in a kind of implicit way using the LLMs. And I think there's a whole spectrum between you know, you don't have to do it explicitly. I think you can kind of use neural networks and you can do some kind of implicit composition and you can get many of these features.

Turing Completeness, SGD & The Symbolic Gap

Jeremy Berman 00:27:13

Also, I I want to say, generally, I speak about language models, I assume that they have basically a Python terminal that they can run. Oh, okay. So not so I guess 2 things. The first is I think if you have a large enough neural network, I think generally almost everything is you could you could represent a symbolic system, but of course, it's not Turing complete. But given a neural network plus the ability to write programs, then I think we're we're basically at the sis the we're at the human brain equivalent.

Tim Scarfe 00:27:50

So, yes, So that that is a hybrid system and that certainly is significantly more powerful. I'm just regurgitating my cohost to Dr. Duggar because this is his, like, favorite point he always likes to make. But but he says that that's true, but stochastic gradient descent does not find the algorithms that allow the systems to behave as if they are Turing machines. God knows how it happened in our brains. There was some dint of evolution or something where, you know, we suddenly got the merge operator or god knows what happened and and we've got this this incredible, like, Turing complete algorithm in in our finite brain. And so so we're getting into that trainability thing. Yeah. You know? So, yes, maybe there is an out you know, there there is a set of weights that we might find 1 day and it can access like a Python, you know, tool and it can do all of those things. And what what you know, its capability now. Is it now effectively searching the

space of Turing machine programs? I think it's not. Like, there's lots of problems there. Like, how how would it know which ones halt and which ones don't? And how would it be able to efficiently search that space? It feels like there's a gap now, but I agree with you that it's significantly stronger than not being able to use the tools.

Jeremy Berman 00:28:56

Yeah. I think I I but but you think that the human brain is running a Turing system?

Tim Scarfe 00:29:01

Yes. I think the algorithm that runs in our brain is a is a Turing machine algorithm. So, you know, like a a Turing machine has a code book, which is a finite state autometer. And then it has this, you know, read write access to these 2 potentially infinite tapes. And, you know, the the the algorithm that you put in that Turing machine, that is very difficult to find.

Jeremy Berman 00:29:23

I I don't disagree with that, but I don't why why wouldn't we be able to find that algorithm for neural networks? Right? Like, why wouldn't, you know, we we train, neural networks much bigger than the brain. We put a lot of compute towards them. Do you you just don't think that, finding the same algorithm is possible with with SGD?

Tim Scarfe 00:29:41

I think I think with SGD because the fascinating thing is that, you know, if you look at all of the FSA algorithms, a a tiny sliver of those algorithms are capable of controlling, you know, a Turing machine and expanding their memory and so on. So it's in the space. And I don't know if you saw that amazing paper by Kenneth Stanley, the fractured and tangled representations paper. And he had this beautiful diagram and he said that, you know, SGD finds the algorithms over here and neuroevolution algorithms find the ones over here. And it just so happens that the neuroevolution algorithms find ones that have these factored, you know, representations, which means they they find representations that are about the world, that are grounded in the world, that carve the world up by the joints. And if only we could find those things. You know, when I spoke to Schmidhuber, he said the same thing. He said, like, you know, it is actually possible to find the right weights in a neural network to make it, you know, effectively Turing complete with some caveats and so on. But when we do SGD, because there are all of these shortcuts, right, it is a bit like Goodharting. It it will always just find the wrong thing.[6]

Jeremy Berman 00:30:43

I need to think about that a bit more. Okay.

Tim Scarfe 00:30:47

So on the first 1 as well, you were generating Python programs explicitly. And because of all the things that we're just talking about, I'm a big fan of that because I feel intuitively, and I think you did, that there's something special about Python programs. And and then you you did to this iterative updating of of those programs and and you converged on the right 1. You also had this amazing diagram in your first blog post where you kind of visualize the space of all the possible programs and you kind of showed what was happening in every iteration.

[6]Fractured Entangled Representation Hypothesis — Paper

Jeremy Berman 00:31:17

In the first 1, I used Python programs because Python programs are deterministic and it's really easy to verify whether or not, you know, it's correct. Did it did it run, and then, did it run on the training examples and produce the correct outputs. So it's like a perfect program. Right? Like, it is a program. The problem is it, you know, Python programs are brittle in that, you know, there are many things that are very difficult to describe with Python. Arc grids in v 2 being 1 of them. Right? So you have some grids that are very easily described by Python, but then almost the majority overwhelming majority in ARC v2 are very hard to describe in Python. The correct Python formulation is, you know, lines and lines and lines, and, really what you want is a more expressive program. And so that's why I switched from Python to English, which is a much more expressive program. You can you can describe every single ARC v2 task in 10 bullet points of plain English, most of them in 5 bullet points. And I think that this actually gets to the heart of ARC. Right? Everything is quite simple. It's not very hard. And I think this is also how we do it too. Right? Like, when we look at these ARC graphs, ARC grids, we're coming up with these bullet points in our head, and we're, you know, checking them. Okay. This was right. This was right. And Python doesn't have these features. It's just not as expressive as natural language. And I think another way to put it would be you have this inductive, transductive trade off. Right? You could think of language models as being trained inductively, and then they have an inductive bias, and you almost want to let that inductive bias express itself fully in a way. And the way you do that is to give it the full power of how it was trained. And I think this is the same thing with humans too. Right? If I told you to solve ARC with Python programs, you'd do a way worse job, even if you were an expert at Python. And so I think fundamentally, it's more general, and it leads to general and better solutions. I mean, the accuracy is much higher when you use natural language. Now, the problem is you actually have to then verify whether the instructions are correct. You can't run natural language on R grids. This was the fundamental problem with the solution. This is what made iteration challenging, especially because you know, for each grid, for each training example, you have to run the natural language instructions, and it takes a really long time, especially with this thinking model. So I originally started with a weak model. You know, it's the checker model. It's the checker agent. Let's just use GPT-4o mini, whatever. Nano. And it did terribly. So I ended up it was actually more important that the checker was stronger than the actual instruction creator, which I I think is, interesting. But, yeah, that just highlights, you know, the the trade offs with using natural language. It's it's you can express, you know, much more concisely programs that you wanna run, but then they're not runnable programs. You actually have to check them inductively. So that

was the trade off, but it was worth it for ARC v2.

Tim Scarfe 00:34:18

Yes. So so fascinating. And just for the audience, we've been using transduction and induction to distinguish predicting the solution space versus predicting a program. I had this this discussion with with Clement Bonnet. We need not detain us now, but I think in traditional machine learning, transduction means that the test example is a function of your prediction. I had this discussion with the architects as well that when you have this natural language description, natural language is more expressive, which simply means that there are more degrees of freedom. And this is the beauty of of LLMs that there's this huge kind of space that you're traversing around. And when you use natural language, you can just traverse to more places in that space more easily. So it seems like it would be a win. I'm And really fascinated to to find out whether that is just like a huge component of your solution. Because on Eric's solution, he's still predicting programs and still doing very well. So I'm not I'm not sure about that. And the other thing is I wasn't entirely sure whether you are actually using a transtactive method. So in your solution checker agent, is it directly going to the solution space, or is it generating a program and testing it?

Jeremy Berman 00:35:24

In the checker? Yeah. In the checker, it takes in the natural language, and then it outputs a grid. That's all it does. It just outputs a grid.

Tim Scarfe 00:35:32

Okay. Cool. So so you've moved to a to a transductive modality. Did you see any errors in that? So did it sometimes produce the wrong grid? Or

Jeremy Berman 00:35:42

All the

Tim Scarfe 00:35:42

time. Yeah.

Jeremy Berman 00:35:43

Yes. Yeah. Okay. All the time. And it it's worth noting, actually. You know, the Python solution was I obviously tried my v 1 solution on v 2. Right? And it wasn't so much worse. But part of what I wanted to do with v 2 is show that as language models get more powerful and we get to use thinking models, we can start using more general solutions. And I just thought there was some something elegant about using natural language. And then it also happened to be that there were problems that I could tell the Python, functions were never going to get. And there are basically no programs that ARC v2, my solution for ARC v2, won't get. So when Grok 6 comes out or GPT 7 comes out, you can use my v 2 solution, it will win. It will beat ARC. That is not the case for my v 1 solution.

Tim Scarfe 00:36:32

The the other important thing is you are now using Grok 4, which is very, very powerful. I I assume you you chose Grok 4 because it just happened to be the the winner on on the leaderboard for for the base ARC v2. And how much uplift is coming from that? So for example, if you used Grok 4 on your on your solution last year, how much better would it be?

Jeremy Berman 00:36:54

So I actually don't think it would be so much better. For some reason, and this is what I talk about in my blog post, these language models are very spiky in certain things where they were trained heavily on. Right? So I think what happened with Grok is there was a distribution of similar, shape tasks, right, grid tasks, just reasoning in the type of gen general direction that allowed Grok to have a special capability in this area. And I actually, like, tested each model. So I tested Grok versus GPT. You know? I didn't just go by the leaderboard, and Grok definitely outperformed. The problem is, you know, for my v 1 solution, you also have to generate code, And Sonnet 3.5 is really good at thinking about code and generating code, and I I prefer Sonnet to Grok, for code generation. So my guess actually would be that if you use my v 1 solution, it's highly possible, you know, Opus 4 1 would be the best. I haven't tested that. Would be very expensive on Opus 4.1, but maybe it's worth testing. I I think that the general idea is that, these networks are are very spiky. When you get into specific domains, the net it it actually very much matters which model you use. And the ARC is a great example of this. Right? Like, leaderboard is super spiky, in ways that, other benchmarks are not.

Tim Scarfe 00:38:06

I I did an interesting interview at NeurIPS last year, with the Google guys, and and they were talking about this adaptive temperature in language models for reasoning. Because, you know, there's this constant trade off between reasoning, we wanna be quite constrained. Right? So so we actually want to kind of, like, go go a particular pathway. We want to be constrained by our knowledge. And when we're being quite creative and flexible, we want to we want to be able to go in in different places. And I'm I'm really interested in creativity, for example, and and and I think creativity is like you you you it's very similar to reasoning as Charley talks about. You know, you're composing together these constraints. There's this phylogeny of of knowledge, and you need to respect it as much as possible because if you don't respect it, you're not grounded anymore. So it kind of feels to me that intuitively code is great because it means that I'm actually respecting the constraints and and the semantics are correct and it's grounded in in the real world. Do you feel in any way that by using these natural language descriptions that you're kind of creating something which might by dint of chance or search find the right solution, but is isn't correct and verifiable?

Efficiency, Search & the Leaderboard Landscape

Jeremy Berman 00:39:12

Yes. Okay. Yes. Tell tell me more. Yes. I I for sure. I think generally, when models think in natural language and they output a natural language, they, they are higher entropy. Right? Yeah. I think the when you the second you start prompting with code, they go into code mode. And this is you know, there are lot of papers that show, right, just by prompting it in a certain direction, it activates certain weights that are, you know, just naturally lower entropy. But that was part of the thing that I wanted. I actually wanted to introduce entropy because, you know, still most arc tasks for v 2, the models don't get close. Right? You know, my solution was the top, and it's at 30%. So I actually wanted to inject as much entropy as possible, which is partially why my, prompts are so broad. You know, I could definitely improve my accuracy on a few tasks by making the prompts more specific, but I wanted to just constantly berate it. More entropy. More entropy. So I actually found that to be a a positive, not a negative.

Tim Scarfe 00:40:14

Interesting. Interesting. On the efficiency of the solution, so the o 3 model from OpenAI, that was about $200 per task. And that was I think it was did we ever find out? I think it was sampling. Right? So they just sampled it a bunch of times. They had a basic verifier. Is is that correct?

Jeremy Berman 00:40:31

I don't think we ever figured that out.

Tim Scarfe 00:40:34

And so you think it could be? So because when I interviewed Chollet, he was he was being quite kind of not cagey, but it it seems like he was suggesting they were actually performing a search algorithm. And and I think the OpenAI guy said on Twitter, no. They were just doing very basic sampling and I think they even published the code that they used. Yeah. I'm not quite sure.

Jeremy Berman 00:40:57

Yes. I I also spoke to the OpenAI guys about this, and I I I'm not sure after all of this what they were doing. I I think it's probable that they were doing sampling. Okay. I mean, it took a very you know, hard to imagine they weren't doing they were certainly doing sampling. I I'm not sure what else they were doing, whether the model was fine tuned. My guess is my my best guess is that they were sampling, and it actually was not a fine tuned model.

Tim Scarfe 00:41:19

Oh, very interesting. Yeah. I remember there was that big, hoo at the time that they you know, it was scandalous that they were training on the training set. But, anyway, that's 1 side. There's also, like, the thought occurs that if they did did if they did do something like what you are doing, so, you know, like this approach of iterative refinement with verification at every single step, would they have done even better?

Jeremy Berman 00:41:42

For sure. Yeah. For sure. I think OpenAI, generally, they wanna do the right thing, and they want their solutions to be very general and broad. This And is the sense I get. I think it's to their culture. I spoke to the OpenAI guys. They did include the training data in that o 3 model, but I think that's fair game. Right? They also so I don't think they fine tune on it. Right? So it's just part of the corpus that went into pretraining, which to me is fair game. Right? This is this is totally fine.

Tim Scarfe 00:42:10

Okay. Very cool. So so on ARC v1, their their efficiency was $200 per task. What was your efficiency?

Jeremy Berman 00:42:17

Oh, on ARC v1, I maybe $10, something like that. I need to something like that.

Tim Scarfe 00:42:23

Okay.

Jeremy Berman 00:42:24

In order of mag may maybe. I need I need to check.

Tim Scarfe 00:42:27

Yeah. So talk to me more about this. So and and Eric Pang's solution, he was slightly came in slightly worse than you, but I think he was a fair bit more efficient. I think his 1 I'm going off my memory now. Was it about $8 per task? Was yours about $30 per task on ARC v2?

Jeremy Berman 00:42:43

0, this is the v 2. So my, yeah, my latest solution. Right. My latest solution was around $30 on v 2 and $8 on v 1.

Tim Scarfe 00:42:50

0, okay. And I think Eric's 1 was slightly more efficient, and and I and he was It was. Indicating that it was because he was doing the library learning and transfer. And and even that, I'd I was left kind of thinking, first of all, it's interesting that you got better results. And is that because there isn't much transfer? You know, where where does the library transfer come into this? Because, know, like, maybe, like, the broader question is, if you were to make your solution significantly more efficient, what would you do?

Jeremy Berman 00:43:18

I had a version that does do library transfer. Basically, I would, save the traces from training and try and basically pull those in during test time. And, I actually, just out of simplicity's sake, because I was getting such high scores with the simple solution, I wanted to just push the simple solution, and I might actually it's we'll see. If someone's gonna beat my score, I might bring that back in. I it's for sure that will improve the score, and it's useful. There is a lot of transfer efficiency. I just found what I was doing very elegant, and so I I actually liked liked to keep it. But, you know, no third party dependencies or anything like that. But that for sure helps accuracy. I think the fundamental reason why I got higher and also, I could match his efficiency, and I would still get higher because I was using natural language. Natural language is a much more efficient area to play in.[7][8]

Tim Scarfe 00:44:14

Yeah.

Jeremy Berman 00:44:15

That's at least what what I found.

Tim Scarfe 00:44:16

Yeah. I I just wonder how close do you think are we getting to the kind of Pareto optimal of of this approach. I mean, to give you a few examples, we interviewed the AlphaEvolve team. That was fascinating. And maybe you can contrast with with those guys. Sakana AI yesterday. Yeah. Robert Lange was the first author. They've released this. I think it's called Shinka Evolve. And that was a kind of similar kind of evolutionary, you know, program thing. And and they had some cool features in there like, you know, using Bandits and using UCB. And I I guess,

[7] AlphaEvolve — Video
[8] Shinka Evolve — Paper

like, are we getting to the point where we're gonna really figure out what is the most optimal way to do this? By way, they were also switching between different foundation models.

Jeremy Berman 00:44:56

I think improvements will be log. Right? Will be will be logarithmic. So I wouldn't expect using these basically, using the language models we have today, I would not expect anyone to break, let's say, 40%, but you could probably make my solution twice as efficient, I would say. You wouldn't get such so much about yeah. You wouldn't get, you wouldn't get more than a few percentage points more accurate is my guess, but you could make it a lot more efficient. There's a ton of efficiency gains to be made.

Tim Scarfe 00:45:25

We've been dancing around this a little bit that, you know, Cholet's measure of intelligence was all about resisting memorization. And there is this question now, you know, which is to what extent are we actually building systems that we might call intelligent? And he says that intelligence is simply like the efficiency of knowledge acquisition. And and I I'm really on board with that. And and I I think it's fair to say at the moment that, let's say, like, you know, your solution and Greenbat solution, it's quite ephemeral and stateless, which is to say that when you have a new task come along, you kind of start again from scratch, which means it's not really like adapting and acquiring new knowledge and transferring that knowledge. So maybe you would agree that in the spirit of Chollet's measure of intelligence at at at the moment, it's more of a kind of searching approach. But what do you think we would need to do to kind of, you know, make it more adaptable?

Knowledge Trees vs Knowledge Webs ●

Jeremy Berman 00:46:22

Right. So I think test time fine tuning would be the way to, like, fundamentally make it adaptable. But I also think, you know, Chollet hits it, a core problem with language models, which is they their reasoning is domain specific. In my blog post, I described that when you train a language model to reason about math, for some reason, most of the reasoning circuits it just gained live in the math weights. And then you try and train it on science, and it gets some generalization, but not as much as you would want, and I think not nearly as much as what what humans get. Humans have this this generalization engine that is

our reasoning capability, and this is this is the fundamental hole in language models today. And I think in fact, actually, would say I I actually I, you know, generally agree with Francois, and he says, you know, language you can always teach a language model skill. Right? But it's the meta skill. It's the skill to create the skills that is AGI. And to me, that's reasoning. Like, reasoning is that meta skill. And so, to put it another way, I think if you fundamentally learn the skill of reasoning, you should be able to then apply that skill to learn all the other skills. That is the meta skill, and we need to figure out and so that is the fundamental problem, and you need to do whatever you can, kick whatever weights out you need to align the model to reason, and then from there, you have a foundation from which you can actually build general intelligence. So I guess I don't know if that was that was maybe a higher level answer to your question, but I think, you know, how how we and what I'm what I'm focused on is really just fitting all of reasoning into these models, and I don't really care what else is left. I just want all of reasoning in.[9]

Tim Scarfe 00:48:07

Yes. I I I pretty much agree. And I mean, you probably know that I'm Chollet's biggest fan, so I've obviously, you know, been a huge fan of his for years. But by the way, he's just released revision 3 of his deep learning with Python book. And I recommend folks to read chapter 19. You can actually read it online for free. And he sketches out this entire vision, you know. It's it's so exciting. And I I think that just to see it so beautifully articulated. Because there is a bit of an elephant in the room and in the scene at the moment, I think so many people just just don't have such a crisp understanding. But the only departure that I make with Chollet and and and yourself, Jeremy, is that I think it's you know, Chollet really focuses on behavioral tests of intelligence, you know, like, so it's so it's reasoning if it can pass the test and it can actually, get the right answer. And I think we need to go this is where I was kind of talking about the systematicity and the symbolic AI. I think how you got there is important. Right? So I think it's possible to get the right answer for the wrong reasons. And I think that if we have a system which has semantics, so we actually know what these symbols mean and we've composed them together in a principled way. Not only to get the right answer for the right reasons, but to be evolvable. So to have like an efficient epistemic base that allows us to go on to acquire new knowledge in the future. And that to me points to this need to have a mechanistic, like, you know, like how are we acquiring this knowledge view. Did would you agree with that?

[9] Deep Learning with Python Rev 3 — Book

Jeremy Berman 00:49:34

Yes. And I I think about it a bit differently. So let me know if what I say is aligned with what you think.

Tim Scarfe 00:49:39

Okay.

Jeremy Berman 00:49:40

So to me, training is kind of the opposite of what you described. I view there's 2 2 types of knowledge. There's knowledge that is memorized like the capital of New York or the Spanish language, and then there's knowledge that is deduced. So that's, physics, special relativity, general relativity. Right? From axioms, you can deduce these things, and it's a causal tree. And then there's a another type of knowledge, what is the capital of North Dakota. That is a knowledge network. It's not deductive. It's not a tree. And I think pretraining treats all knowledge as a knowledge web. It's embeddings that are connected, but there's no there's no guarantee that you have the correct causal relationship between things, and this is where the memorization comes in. And I think this is actually where compression fits into intelligence. So my view is actually intelligence is compression in that you should be able to deduce you should be able to build a knowledge tree based on almost you can build a knowledge tree based on almost nothing. Right? You can deduce so much of math. You can deduce special relativity from the very, you know, the the very roots of physics. And Einstein was extremely intelligent because the hints that he needed to come up with special relativity are 0. Right? He could start from almost nothing and build up this deductive tree. And I think it's almost like reinforcement learning and reasoning is the process of pruning our knowledge network and replacing it with this tree. And and until we have weights that represent the actual deductive nature of knowledge, we won't actually get generalization. I don't know if this this fits in, but this is kind of how I think about reinforcement learning, which is replacing knowledge web with an with a knowledge tree.

Tim Scarfe 00:51:19

Yes. Yes. This is this is brilliant. We're getting to the center of the bull's eye here. Absolutely. And I I remember I read in in the first version of your blog post that you were talking about, we need to do this kind of deduction where we synthesize hypotheses, and then we we test them, and we do this kind of generate test loop. And and that is what creativity is. It's what reasoning is. Because, you know, when I first read Charle's paper, you know, years ago, I was kind of like, I didn't understand whether he was talking about acquisition or synthesis. And I now understand he's talking about synthesis. So reasoning is like you just like Lego, you you build you build this kind of tree, this epistemic tree. And actually, this is what we do. So there's a difference between knowing and understanding. So knowing is is kind of like at the high And understanding is is actually like there's this whole, you know, just imagine this big block of this big Lego structure and you're tracing down the structure of all the building blocks of how you got there. And I actually think even when you teach kids at university, what you're really doing, obviously, like, you teach you teach them facts, but then they kind of synthesize their understanding over time. So they're doing this composition and they're kind of getting there the way that they get there. And and we need to build systems to do this. And there's the perennial problem that you were saying that in deep learning, what we do is we kind of start with this big pattern network and we kind of specify it. And I think reasoning should be more about synthesizing from building blocks. And I I think that when you synthesize, you can actually do types of reasoning which are not in the training data. Right? You can build things that simply are not there. You can just think about things and and and figure things out. So do you think of that as a gap?

Jeremy Berman 00:52:57

Yes. I think that's exactly right. And then the question is, can you build the system with language models or not?

Tim Scarfe 00:53:02

Yeah.

Jeremy Berman 00:53:03

And I I think you can build that with language models. And I think the fact that we're slowly climbing, our ability to synthesize new information is a testament to what I'm saying, which is when reinforcement learning and reinforcement learning, you know, with verifiable rewards is is fundamentally ensuring whatever circuits led to the right answer, they must be consistent with the deductive tree. So it's basically like, can you replace all of your pretrained weights with weights that are, coherent from the environment? The problem is all you know, there's so many weights. There's so many weights from pretraining that it's very difficult. So I actually think a part of, you know, 1 of my, I guess, hot takes is that pretraining in many ways slows down reasoning. It makes it harder to reason because I I think I think the analogy I draw is, you know, you have consultants that know the names for things but couldn't deduce the thing. And then you have Feynman, right, who can deduce anything. And reinforcement learning is is turning your consultant into Feynman. And this is yeah. This is what I'm most interested in. And I think it's it's interesting because you get to play at both sides. You get to play with the pretraining of, okay. Well, maybe we shouldn't include these things and let the model figure it out in reinforcement learning. Well, there's no guarantee if we pre train it like this, you know, it's gonna have the proper reductive, circuitry, so maybe this is best, you know, left for post training. And I I think this is not this is this is a hot this is not what people currently think. I think people think, let's jam as much information as we can in pre training, and then we'll reinforcement learn when need when we need. But I think this this could be incorrect.

Tim Scarfe 00:54:39

Yeah. And I I pretty much agree with you just with the caveats we discussed previously that if we could do that on the basis of representations that are actually, like, grounded in the world rather than things that just, you know, happen to give you the right answer for the wrong reasons, that's that's absolutely true. Just a little bit of a a curve ball. So I think in the first version of the article, you said you're inspired by Jan Lakun's JEPPA, these joint embedding prediction architectures. And he's also a big advocate of energy based models, which are really cool. I don't know if you've seen the recent couple of papers that are applying it to transformers, you know, where essentially it's it's kind of a step towards probabilistic models where you actually have this uncertainty quantification. You can do counterfactuals and you actually have to solve an optimization problem at inference time so you can do adaptive computation. It's all very exciting. But I still have some reservations, but do you think, like, architectures like that are exciting?

Jeremy Berman 00:55:33

I think they are exciting. I am a bit less excited about them, not because of their merits, but because I think I was underrating transformers when I wrote that. So this was really before I started do it being an actual researcher and, you know, building transformers, actually coding with them. Since then, I've had a new appreciation for language models. I think where I was coming from it was it seems like language models are overfitting to the next token. JEPA is so interesting because, oh, all of a sudden, you have them predicting concepts. Fundamentally, we care about concepts, the words don't really matter, it's the concepts that matter. But I think that language models do operate on the conceptual level in the hidden layers. That was something that I slowly came to realization of. So I think there's lot of potential in JEPA frameworks. Think they're really cool. I hope people keep pulling on them. But I think most of the benefits I thought that came from JEPA exist in language models. I just didn't see it at the time.

Tim Scarfe 00:56:34

You know a lot of people get an earworm where they just get obsessed about an idea and you're kind of thinking about it all the time. Like, what is that thing

Creativity, Invention Circuits & Epistemic Trees

Tim Scarfe 00:56:41

for you?

Jeremy Berman 00:56:41

If we do have these, language models that have their weights that are aligned in this, I guess, the the tree of deduction, It seems like we're missing still 1 more thing, is creativity, which we touched on. Right? Yep. So you can have the correct deductive tree, but then how do you search through all of the possible premises you can add to this tree? How do you find the right ones? And I think there are a few experiments that I'm looking forward to doing. 1 of them is ablating at will pretrained data and then, basically, reinforcement building an environment to have the model regenerate that information. So for example, let's say you had the ability to ablate special relativity and all of the physics that came from special relativity from your pre training data. That is a gold mine of an environment, right? Because now you can prompt the model. You can do everything. You can really try and get it to to to deduce special relativity. And my hunch is that part of the reason why models are not yet great at coming up with novel solutions

and information is because they don't have that that the circuitry of invention. And I think that is actually a circuit that needs to be developed, and we don't have the environments to develop that circuit yet. And I actually just saw today, I think OpenAI released this math. I I skimmed it. This math paper, right, where they're coming up with they almost came up with novel conjecture or something like that. And I think that's exactly what was in my head, which is I I wanna be able to build environments where the model's never seen something, and, basically, it it, tries to deduce these new things, that that are, you know, outside of the distribution, and, over time, it it learns and practices and builds this invention circuit. So I think it's like it's 2 things. It's reinforcement learning to make sure that the knowledge tree is consistent, and then making sure that the circuitry to be able to pull from its entire corpus of understanding of the world and bring that in to fuel the innovation engine.

Tim Scarfe 00:58:44

Yeah. I think we're so close. I think the only slight disagreement is whether there could be such a thing as an invention circuit. It seems to me, just like that LLM biology paper and whatnot, that it would be very kind of patterned in the way. So it would be templated. It would be kind of specific to certain domains. And I feel that we can do it, but we would need to build a controller on the top. I also feel, by the way, that creativity is very domain specific. And what I mean by that is just with this Kenneth Stanley view that there's this big phylogeny of knowledge. And I've noted that when I hire creative professionals, they like an editor, for example, they can't edit my show. And even if they're really good at editing other people's shows, and it's because they simply don't know anything about machine learning. So, you know, like, I've I've just discovered the crazy degree to which creativity is domain specific. And I just wonder whether there is an algorithm for creativity. Because I've got this epistemic lens of creativity, which is that, you know, it's simply about, okay, well, I can access all of the ancestors in my tree and I can do some composition that's like building Lego and so on. And and if I wanna use something from the tree over here, this other branch, you know, maybe it's compatible, maybe I can bring it in, or maybe I just need to start a new branch, or maybe I need to jump over to the other branch. And I'm not sure whether I'm applying the same algorithm when I'm doing that.

Jeremy Berman 01:00:04

I think you I think you actually are, and that it's you it's not sufficient to just be creative in this case. You need to be creative, and you need to be knowledgeable. Otherwise, you're creative. Right? You you can't build the tree. I think your editors can't build the the tree because they don't have the deductive footing. That that would be my that would be my best guess, but that's an interesting perspective. I wanna think about that. I mean, fundamentally, right, creativity is knowing which axioms to include in the next branch of the tree. Right? So you're at level 5 of the tree. How do you get to level 6? It's knowing which assumptions to pile in to get to level 6. Right?

Tim Scarfe 01:00:43

It's it's so beautiful. We're writing an article about creativity at the moment. And I believe that in order to be creative, the the depth of understanding of the tree is very important. And, you know, just per Chollet, you know, intelligence is is the efficiency of how much of the history you can acquire. So like a university professor understands the tree very deeply. And that actually makes them unintelligible to a normal person. Like when Stephen Wolfram talks about the Rulliad all the time, people have got no idea what he's talking about. He's actually being very expressive. He's talking about things at a level of abstraction which, you know, can refer to anything, but it's beyond most people's cognitive horizon. But there is something to be said for that when when you can respect the history deep down into the epistemic tree, the creative stepping stones you take, because they respect the history, they actually have more evolvability. Right? Because you're you're you're you're still grounded to the real world. You're not becoming incoherent. So there's there's something there about really knowing things deeply that is important.

Jeremy Berman 01:01:41

Yes. Yes. I I'm for the record, I would define understanding as being able to I mean, I think understanding the spectrum, I think on 1 end, it's memorization, and then which is 0, understanding. And then on the other end, it's ability to deduce and, and also to deduce correctly. I agree with what you said. It's not enough to just, you know, have the right proof. It's you actually have to have, got you know, understood the tree. And that is like that this is understanding, and then intelligence is just how many things you understand. And so it's really just how how how wide and and high is your garden of trees.

Tim Scarfe 01:02:17

Yeah. We're we're pretty similar. I I would say intelligence is the efficiency that you can acquire the tree, and reasoning is building the tree. Discursive reasoning is executing the tree, and and and understanding is simply just possession of the tree.

Jeremy Berman 01:02:30

So intelligence to you is the speed at which you can build the tree, not how many trees you have. Yeah. Or not how large your tree is.

Tim Scarfe 01:02:37

Yeah. Yeah. Yeah. I I think understanding is how much of the tree do I have?

Jeremy Berman 01:02:42

I think that's correct. Yes. Yes. Because you like, as you could have a very intelligent child who doesn't know a lot about the world, but has the ability. It's like potential to build a tree. Yes. It's yes. I I think that's correct.

Tim Scarfe 01:02:56

There's a spectrum of understanding. So language models, famously, they don't understand the tree very deeply. So they they only understand the tree a few levels down. And so when language models are doing autonomous generation, the reason why we have to do so many different generations and select the best 1 is because obviously, like, it's not grounded. It doesn't understand the tree very deeply. Exactly. We can overcome that, but because we understand the tree deeply, so we can put a prompt in there that constrains their generation, and now we can make them act as if they understood the tree when they didn't. But, you know, we just need to build the models that do understand the tree deeply, and then we can just trust them to generate autonomously.

Jeremy Berman 01:03:33

Yes. That is that is a very good way of of saying what what I think, and that is what I'm focused on. I that's a really good way of putting it, right, which is, forcing the language models to develop these deep trees from the ground up. It you can only develop it from the ground up, I think. Yeah. And so you need to be you we need to come up with new techniques, new environments to, to grow the trees instead of pretraining, which is pre filling random you know, it's not random, but, it's a web. It's not a it's not a tree.

Spelke Priors, Physical Grounding & Closing

Tim Scarfe 01:04:03

Yes. There's also the vexed issue of what happens at the bottom of the tree. So, Chollet argues that this Spelke core knowledge, you know, these are knowledge primitives that are so fundamental that inside the deductive closure of those primitives, we can talk about anything. And, you know, you can still get lost in different parts of the tree, suppose. And there's some issues of intelligibility going between them. But if you understood the tree deeply enough, you can go anywhere. But maybe there are different trees. Maybe there are, know, like in physics, are different levels of, you know, levels of description of understanding the universe. So do you think it's like 1 big tree or lots of trees?

Jeremy Berman 01:04:38

I think it's kind of gated in the laws of whatever you're doing. And I looked into this a few times. I know there are some axioms that you need to take for granted in, let's say, deducing some forms of math. Right? And they would be their own tree because you can't get to 1 part without the other. You know, if you take this for granted, then you can't, you can't deduce this, but you can deduce this. And so I'd say basically, they have to, you know, they they have to be a logical chain, and so of course, there might be multiple chains. And maybe everything is grounded in logic. Logic is the fundamental I guess that is true, actually. Logic must be the fundamental block of a tree. Everything comes from logic because if we didn't have logic, we couldn't have trees. So I guess that yeah. I guess it would be all 1 tree, it's a logic tree. And, I guess there are people that subscribe to logic trees, then there are people that are illogical, and they they don't have their trees at all.

Tim Scarfe 01:05:30

Jeremy, it's been an absolute honor to have you on the show. Just before we go, are you hiring or anything you wanna say to the audience?

Jeremy Berman 01:05:37

For sure. Yeah. So at Reflection, we're building open intelligence models. So we're hiring across the stack, pretraining, post training, large language models. Have a lot of GPU. So, if you're interested in pre training, post training, we're in SF, we're in New York, and we're in London. So definitely, apply on our site, or you can just hit me up on Twitter.

Tim Scarfe 01:05:59

Amazing. Jeremy, I've really enjoyed this. Thank you so much.

Jeremy Berman 01:06:02

Awesome. Thank you.

Tim Scarfe 01:06:03

Just keep doing what you're doing, man. I I just I really, really think that you that that you you're onto something here. I mean, obviously, like, with the minor, you know, discussion about, you know, whether how how we're actually gonna do this. But I think the the direction is is quite clearly set. I know. It's such a fixed issue, though. I mean, I'm I'm into being a bunch of cognitive scientists in Japan next week. And I you can really go down the rabbit hole on this. So for example, I'm a big fan of, you know, ex like, externalism, you know, like inactive cognition. And there are also all these philosophical views where, you know, like, consciousness is basically a property of certain types of physical material. So because you know what we're talking about here with understanding. Like, if you actually abstract it into physics, we're talking about certain types of causal graph. And you can argue that certain types of cognition actually require certain types of physical instantiation, you know, where in the middle of that graph, have material which is capable of producing consciousness. So some components of understanding are phenomenal. They're conscious. Right? And, like, when when you start taking it to this philosophical level, there there's almost like no there's no end to it. You know? Because you'll always have people that argue against functionalism that that say that cognition must be physically instantiated in a certain way. And I'm not sure where I am on that because even the cognitive scientists are saying, okay. Well, guys, we have to admit that, you know, even though we can make all of these arguments, I mean, fuck me. These elements are they're they're doing so well. But, yeah. I I think it'd be a fascinating discussion because a lot of neuroscientists are internalists. They basically like Jeff Hawkins, they think that all of the shit happens in the brain. And we have these sensory motor circuits, and we've got this master algorithm in our neocortex, and that does all of the things. And I think there's there's something to be said for that. But I think it does ignore quite a lot of the other field. Yeah. Well, it all comes down to if we could build machines that actually have a deep grounded understanding of the world. And let's assume that that doesn't actually have to be physically grounded. Right? If if the representation is, like, you know, grounded in the sense that it's a faithful description of what is happening out there, and we can do this creative reasoning on that understanding, then what's to stop us from inventing new things? I mean, that's basically the the-

sis. Right? Yeah. Beautiful stuff. Jeremy, thank you so much, man. Really, really appreciate this.

References

[1] **Jeremy Berman's ARC-AGI v1 Blog Post**
https://jeremyberman.substack.com/p/how-i-got-a-record-536-on-arc-agi
Blog Post

[2] **A Thousand Brains**
https://www.amazon.com/Thousand-Brains-New-Theory-Intelligence/dp/1541675819
Book

[3] **NDEA**
https://ndea.com/
Company

[4] **Getting 50% on ARC-AGI with GPT-4o**
https://blog.redwoodresearch.org/p/getting-50-sota-on-arc-agi-with-gpt
Blog Post

[5] **On the Biology of a Large Language Model**
https://transformer-circuits.pub/2025/attribution-graphs/biology.html
Paper

[6] **The ARChitects**
https://www.youtube.com/watch?v=mTX_sAq--zY
Video

[7] **Connectionism and Cognitive Architecture**
https://uh.edu/~garson/F&P1.PDF
Paper

[8] **Fractured Entangled Representation Hypothesis**
https://arxiv.org/pdf/2505.11581
Paper

[9] **AlphaEvolve**
https://www.youtube.com/watch?v=vC9nAosXrJw
Video

[10] **Shinka Evolve**
https://sakana.ai/shinka-evolve/
Paper

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

[12] **Deep Learning with Python Rev 3**
https://deeplearningwithpython.io/

Book