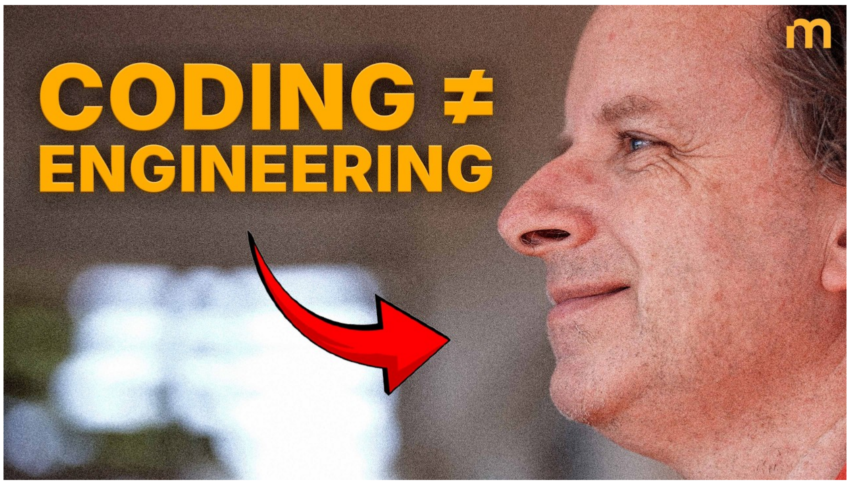
/mlst

Jeremy Howard: ULMFiT, Fine-tuning, and Intuition in ML (FINAL)

March 3, 2026



Machine Learning Street Talk

Podcast Transcript

Revision: 21064923

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

Contents

Overview

Dive into the realities of AI-assisted coding, the origins of modern fine-tuning, and the cognitive science behind machine learning with fast.ai founder Jeremy Howard. In this episode, we unpack why AI might be turning software engineering into a slot machine and how to maintain true technical intuition in the age of large language models.

GTC is coming, the premier AI conference, great opportunity to learn about AI. NVIDIA and partners will showcase breakthroughs in physical AI, AI factories, agentic AI, and inference, exploring the next wave of AI innovation for developers and researchers. Register for virtual GTC for free, using my link and win NVIDIA DGX Spark (https://nvda.ws/4qQ0LMg)

Jeremy Howard is a renowned data scientist, researcher, entrepreneur, and educator. As the co-founder of fast.ai, former President of Kaggle, and the creator of ULMFiT, Jeremy has spent decades democratizing deep learning. His pioneering work laid the foundation for modern transfer learning and the pre-training and fine-tuning paradigm that powers today's language models.

Key Topics and Main Insights Discussed

The Origins of ULMFiT and Fine-Tuning Jeremy shares the backstory of developing ULMFiT on a gaming GPU, proving that language models could be pre-trained on a general corpus like Wikipedia and fine-tuned for specific tasks. He explains the intuition behind discriminative learning rates, progressive unfreezing, and the critical importance of fine-tuning batch normalization layers to prevent dead neurons.

The Vibe Coding Illusion and Software Engineering We critically examine the current hype around AI-assisted software engineering. Jeremy highlights the profound difference between merely typing syntax and actual software engineering. He warns that AI coding tools often act like slot machines, exploiting gambling psychology by offering intermittent rewards and an illusion of control, which can ultimately lead to a decline in real productivity.

Cognitive Science, Friction, and Learning Drawing on philosophy and cognitive science, the conversation explores whether LLMs truly understand the world or just cosplay intelligence. Jeremy emphasizes that human knowledge requires friction to grow. By delegating too much cognitive load to AI, developers risk skill atrophy and organizations risk losing their foundational capacity to evolve.

The Future of Developers Jeremy offers a stark warning for mid-level developers. While AI tools are great for absolute beginners and highly experienced architects automating boilerplate, they actively remove the desirable difficulty necessary for developers to build deep mental models. We discuss a real-world example of using AI to fix complex IPykernel bugs, resulting in working code that no human actually understands, creating a dangerous new form of technical debt.

Introduction & GTC Sponsor ●

Jeremy Howard 00:00:00

It it it it literally disgusts me. Like, I literally think it's it's inhumane. My mission remains the same as it has been for, like, 20 years, which is to stop people working like

Dr. Tim Scarfe 00:00:12

this. Jeremy Howard, a deep learning pioneer, a Kaggle grandmaster, he is a huge advocate for actually understanding what we are building through an interactive loop, a notebook, a REPL, the act of poking at a problem until it pushes back. He argues this is where the real insight happens.

Jeremy Howard 00:00:35

And the funny thing is they're both right. LLMs cosplay understanding things. Like, they pretend to understand things. No one's actually creating 50 times more high quality software than they were before. So we've actually just done a study of this and there's a tiny uptick, tiny uptick in what people are actually shipping. The thing about AI based coding is that it's like a slot machine and that you you have an illusion of control, you know, you can get to craft your prompt, and your list of mcps, and your skills, and whatever, and then but in the end, you pull the lever. Right? Here's a piece of code that no 1 understands. And am I going to bet my company's product on it? And I the answer is I don't know. Because like I I don't I don't like, I I don't know what to do now because no one's, like, been in this situation. They're they're really bad at software engineering. And then I think that's possibly always gonna be true. The idea that a human can do a lot more with a computer when the human can, like, manipulate the objects in inside that computer in real time, study them, move them around, and combine them together. Whoever you listen to, you know, whether it be Feynman or whatever, like, you always hear from the great scientists how they build deeper intuition by by building mental models, which they get over time, by interacting with the things that they're learning about. A machine could kind of build an effective hierarchy of abstractions about what the world is and how it works entirely through looking at the statistical correlations of a huge corpus of text using a deep learning model. That was my premise. This video is brought to you by NVIDIA GTC. It's running March 16 until the nineteenth in San Jose and streaming free online. The key topics this year are agentic AI and reasoning, high performance inference and training, open models, and physical AI and robotics. I'm so excited about the DGX Spark. I've been on the waiting list for over a year now. It's a personal supercomputer that is about the size of a Mac Mini. It's the perfect adornment to a MacBook Pro by the way, and you can fine tune a 70,000,000,000 parameter language model with 1 of these things and I'm giving 1 away for free. All you have to do is sign up to the conference and attend 1 of the sessions using the link in the description. As for the sessions, I'm interested in attending Aman Sanger's talk. So, he's the co-founder of Cursor and his session is code with context build an agentic IDE that truly understands your code base. Now, obviously Jensen's keynote is on March 16. He said he's going to unveil a new chip that will surprise the world. Their next generation architecture Vera Rubin is already in full production and there's speculation we might even get an early glimpse of their new Feynman architecture. So don't forget folks, the link is in the description. If you're attending virtually, it's completely free. Don't miss it.[1]

Dr. Tim Scarfe 00:03:57

Jeremy Howard, welcome to MLST.

Jeremy Howard 00:03:59

I mean, welcome to my home. Thanks for

Dr. Tim Scarfe 00:04:02

coming. Yeah. Well, where are we

Jeremy Howard 00:04:04

now? We are in beautiful Moreton Bay in Southeast Queensland. We are by the sea in my backyard.

Dr. Tim Scarfe 00:04:12

Weather didn't disappoint.

Jeremy Howard 00:04:13

It certainly didn't. It doesn't often but if you were here yesterday, it would have been very different.

[1]fast.ai Blog: Self-Supervised Learning — Blog Post Referenced during the ULMFiT and fine-tuning discussion, covering self-supervised learning foundations.

Dr. Tim Scarfe 00:04:18

Well, don't know where to start. I've been I've been a huge fan probably since about 02/1718. Of course, you had the famous ULMFiT paper and when I was at Microsoft I remember doing a presentation about that because it was actually, I mean now we take it for granted that we fine tune language models on a corpus of text and then we kind of like continue to train them and and specialize them. But apparently this was not received wisdom.

ULMFiT & The Birth of Fine-Tuning

Dr. Tim Scarfe 00:04:18

Well, don't know where to start. I've been I've been a huge fan probably since about 02/1718. Of course, you had the famous ULMFiT paper and when I was at Microsoft I remember doing a presentation about that because it was actually, I mean now we take it for granted that we fine tune language models on a corpus of text and then we kind of like continue to train them and and specialize them. But apparently this was not received wisdom.

Jeremy Howard 00:04:42

No. This was the first time it happened. Yeah. Kind of the first or second. So, McCann and Andrew Dai had done something a few years ago, but they had missed the key point, which is the thing you pre train on has to be a general purpose corpus. So no 1 quite realized this key thing, and maybe I had a bit of fortune here that my background was in philosophy and cognitive science, and so I spent some decades thinking about this.

Dr. Tim Scarfe 00:05:06

The technical architecture of of

Jeremy Howard 00:05:08

of ULMFiT,

Dr. Tim Scarfe 00:05:08

just just sketch that

Jeremy Howard 00:05:10

out. I'm a huge fan of regularization. I'm a huge fan of taking a model that's incredibly flexible, and then making it more constrained, not by decreasing the size of architecture, but by adding regularization. So even that at the time was extremely controversial, but that was by no means a unique insight of of ours. So what Stephen Merity had done is he'd taken the extreme flexibility of an LSTM, a kind of the classic state for recurrent neural net towards which things are kind of gradually heading back towards nowadays, added added 5 different types of regularization. He added every type of regularization you can imagine, and then that was my starting point, was to say, now have a massively flexible deep learning model that can be as powerful as I want it to be, and it can also be as constrained as I need it to be, and then I needed a really big corpus of text. Funnily enough, this is also Steven. He had been at Common Crawl, and I think he helped or made the Wikipedia dataset. Then I realized actually the Wikipedia dataset made lots of assumptions. It had all these like, unk for unknown words, because it all assumed classic NLP approaches. So I redid the whole thing, created a new Wikipedia dataset. Now it's my general corpus. And then I used an AWD-LSTM, trained it so it's actually overnight. So for 8 hours on a gaming GPU, you know. Because I was at the University of San Francisco, we didn't have heaps of resources. Probably like a 20 80 TI or something, I suspect. And then the next morning when I woke up, I then it's the same 3 stage architecture that we do today. You know, pre training, mid training, post training. Mhmm. So then I figured, okay, now that I've trained something to predict the next word of Wikipedia, it must know a lot about the world. I then figured if I then fine tune it on a corpus specific, so what we could now call supervised fine tuning dataset, which in this case was the dataset of movie reviews. It would become especially good at predicting the next word of those, so I'd learn a lot about movies. Did that for like an hour, and then like a few minutes of fine tuning the downstream classifier, which was a classic academic data set. It's kind of considered the hardest 1, which was to take like 5,000 word movie reviews and to say like, is this a positive or negative sentiment? Which today is considered easy. But at that time, you know, the only things that did it quite well were highly specialized models that people wrote their whole PhDs on, and I beat all of their results, you know, 5 minutes later when it fine tuning that model. It was

Dr. Tim Scarfe 00:08:03

amazing. And the other interesting thing is this kind of methodology around how you do the fine tuning.

Jeremy Howard 00:08:10

Yeah. So the how we do the fine tuning was something we had developed at Fast AI. So this is kind of year 1 of Fast AI. So this is still in our very early days. And 1 of the extremely controversial things we did was we felt that we should focus on fine tuning existing models, because we thought fine tuning was important. Some other folks were doing work contemporaneously with that, so Jason Yosinski did some really great research. I think it's during his PhD on how to fine tune models, and how good they can be, and some other folks in the in the computer vision world. We were, you know, amongst the first. There's a bunch of us kind of really investing in fine tuning. And so, yeah, we we felt that using a single learning rate to fine tune the whole thing all at once made no sense, because the different layers have different behaviors. This is 1 of the things Jason Yosinski's research also showed. We developed this idea of like, well, it's also way faster if you just train the last layer. Right? Because it only has to back prop the last layer. And then once that's pretty good, back prop the next the last 2 and then the last 3. And then we use something called discriminative learning rates. So different layers, we would give different learning rates to. And then another critical insight that no 1 realized for years, even though we told everybody, was that you actually have to fine tune every batch norm. So all the normalization layers, you do actually have to fine tune, because that's moving the whole thing up and down, know, changing its scale. So yeah, when you do that, you can often just fine tune the last layer or 2. And we found that actually with ULMFiT, although we did end up unfreezing all the layers, only the last 2 were really needed to get the close to state of the art result. So it like took like seconds.[2]

Dr. Tim Scarfe 00:10:01

Yeah. Because the discriminative learning rate thing is interesting because I I think the received wisdom at the time was when you fine tune a model, if the learning rate is too high, you kind of blow out the representations. So I guess the wisdom was if if you don't have a really low learning rate, you'll just destroy the representations. I mean,

[2]M. Chirimuuta: The Brain Abstracted — Book Book on abstraction and cognition referenced during the learning mechanics discussion.

Jeremy Howard 00:10:17

there there was no received wisdom because nobody talked about it. No 1 cared, you know. It's just this sort of like, nearly no 1 cared. Transfer learning was just not something anybody thought about. And Rachel and I felt like it matters more than anything, you know, because only 1 person has to train a really big model once, and then the rest of us can all fine tune it. So we thought we just should learn how to do that really well. So we spend a lot of time just trying lots of things. But in the end, the intuition was pretty straightforward, and what intuitively seemed like it ought to work, basically always did work. Which is another big difference between how people still today tend to do ML research, is I think it's all about ablations, and you can't make any assumptions or guesses, and it's not at all true. I find nearly everything that I expect to work almost always works first time, because I spend a lot of time building up those intuitions, that kind of understanding of how gradients behave.

Dr. Tim Scarfe 00:11:27

I I think there's a dichotomy though between continual learning, which is when we want to keep training the thing but maintain generality, versus fine tuning a thing to do something specific. So there's always been this idea that, yes, you can make a model specific. You can bend it to your will. But you lose generality, and you kind of degrade the representation. So tell me about that.

Jeremy Howard 00:11:48

Yeah, there's some truth in that, although not as much as you might think. On the whole, the big problem is that people don't actually look at their activations and don't actually look at their gradients. So something we do in our software, in our fast AI software, is we have built into it this ability to to see in a glance what your entire network looks like. And once you've done it a few times, it just takes a couple of hours to learn, you can immediately see, oh, I I see. This is over trained or under trained or this layer that something went wrong. It's not a mystery, you know. So basically what happens is, for example, you end up with with dead neurons that go to a point where they they've got 0 gradient regardless of what you do with them. That often happens if they, you know, head off towards infinity. You can always fix that. So yeah, it's it's not as bad as people think by any means. Something that trains well for continuous learning, when done properly, can also be done well to train well for a particular task, if you're careful.

Intuition & The Mechanics of Learning ●

Jeremy Howard 00:11:48

Yeah, there's some truth in that, although not as much as you might think. On the whole, the big problem is that people don't actually look at their activations and don't actually look at their gradients. So something we do in our software, in our fast AI software, is we have built into it this ability to to see in a glance what your entire network looks like. And once you've done it a few times, it just takes a couple of hours to learn, you can immediately see, oh, I I see. This is over trained or under trained or this layer that something went wrong. It's not a mystery, you know. So basically what happens is, for example, you end up with with dead neurons that go to a point where they they've got 0 gradient regardless of what you do with them. That often happens if they, you know, head off towards infinity. You can always fix that. So yeah, it's it's not as bad as people think by any means. Something that trains well for continuous learning, when done properly, can also be done well to train well for a particular task, if you're careful.

Dr. Tim Scarfe 00:12:54

In a sense, you do want the neurons to die out. And I'll explain why what I mean by this. Like, we want to bend the behavior of models to introduce implicit constraints. Because without constraints, there is no creativity, there is no reasoning, and and so on and so forth. So so in a sense, actually want it to say, don't do that. You want it to do something else.

Jeremy Howard 00:13:16

I don't think of it that way. Like, to me, it's more like, I find thinking about humans extremely helpful when it comes to thinking about AI. I find they behave more similarly than differently, and my intuition about each tends to work quite well. You know, with a human, when you learn something new, it's not about unlearning something else. And so something I always found is when I got models to try to learn to do 2 somewhat similar tasks, They almost always got better at both of them than 1 that only learned 1 of them.[3][4][5]

[3]DeepMind Blog: Gemini Deep Think — Blog Post Referenced during the AI reasoning and abstraction discussion.

[4]MLST Archive: Why Creativity Cannot Be Interpolated — Archive Article MLST archive piece on creativity and interpolation referenced in the abstraction discussion.

[5]Mathilde Caron et al.: Emerging Properties in Self-Supervised Vision Transformers (DINO) — Research Paper Self-supervised Vision Transformer paper referenced at 13:54. Authored by Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin at Facebook AI Research (not Yann LeCun, as stated in the episode).

Dr. Tim Scarfe 00:13:52

I was reminded a little bit of, you know, the DINO paper from LeCun's lab. So this whole kind of regime of self supervised learning with with I mean, that that was that was a vision model. But the idea was, Okay, so we're doing pre training. And we want to maintain as much diversity and fidelity as possible so that when we do the downstream task, we can kind of we've got more things that we can latch on.

Jeremy Howard 00:14:14

Yeah, yeah. And you know, semi supervised and self supervised learning was such an unappreciated area. And, yeah, Yann LeCun was absolutely 1 of the guys who was also working on it.

Dr. Tim Scarfe 00:14:27

Yeah.

Jeremy Howard 00:14:27

I actually did a post, because I was so annoyed at how few people cared about semi supervised learning, did a whole post about it years ago. Jan LeCun looked at it for me as well, and, you know, suggested a few other pieces of work that I'd I'd missed, and but I was kind of surprised at how, know, how incredibly useful it is to basically say, like, basically come up with a pretext task, right? So envision so we did this envision before ULM fit, so it was like, in medical imaging, you know, take a histology slide and predict, you know, mask out a few squares, and predict what used to be there. So some of my students at USF I had doing stuff with that. It was basically entirely taking stuff that we and others had already done in vision.

Dr. Tim Scarfe 00:15:16

Yep.

Jeremy Howard 00:15:17

So like this idea of masking out squares. We didn't invent it. Masking out words was the obvious thing, you know, and this idea of gradually unfreezing layers, we had done before in computer vision. The whole idea of starting with a pre trained model that was general purpose had been in computer vision. There was a really classic paper, actually, in computer vision in might have been around 2015, was entirely an empirical paper saying, look what happens when we take a pre trained ImageNet model predicting what sculptor created this or predicting what architecture style this is, and like in every task, it got the state of the art result. And it really surprised me people didn't look at that and think like, I bet that ought to work in every other area as well, whether it be genome sequences, or language, or whatever. But people have a bit of a lack of imagination, I find. They tend to assume things only work in 1 particular field. That's really true.

Dr. Tim Scarfe 00:16:15

Yeah. I mean, I guess there's 2 things there. Mean, of all, we were kind of hinting at this notion of almost Goodhart's Law, the shortcut rule that you get exactly what you optimize for at the cost of everything else. But that doesn't seem to be the case because we can optimize for perplexity in the case of language models. And as you say, what seems to happen is we're getting into the distributional hypothesis here a little bit. So you know the word by the company it keeps. So when we have an incredible amount of associative data, it might be masked auto prediction or any of these things like that, the model seems to build something that we might call an understanding.

Jeremy Howard 00:16:49

Well, I I have always thought of it as a hierarchy of of abstractions, You know, it it it needs if it's gonna predict, you know, if the document is here was the, you know, opening that, you know, that Bobby Fischer used, and has chess notation to predict the next thing, it needs to know something about chess notation, or at least openings. If it's like, you know, and this was vetoed by the 1956 US president, comma, you need to know it's bit like, you don't even you don't just need to know who the president was, but the idea that there are presidents. And therefore, the idea that there are leaders, and therefore, the idea that there are groups of people who have hierarchies, and therefore, that there are people, and therefore, that there are objects, and like, you can't predict the next word of a sentence well without knowing all of these things. So that knowing my hypothesis for why I created ULMFiT was to say it would end to to to compress that as well as possible to get that knowledge, it would have to create these abstractions, these hierarchies of abstractions somewhere deep inside its model. Otherwise, how could it possibly do a good job of predicting the next word? You know, and because deep learning models are universal learning machines, you know, and we had a universal way to train them, I figured if if we get the data right and if the hardware is good enough, then in theory, we ought to be able to build that next word predicting machine, which ought to implicitly build a hierarchical structural understanding of the things that are being described by the text that it is learning to predict?

Abstraction Hierarchies & AI Creativity •

Jeremy Howard 00:16:49

Well, I I have always thought of it as a hierarchy of of abstractions, You know, it it it needs if it's gonna predict, you know, if the document is here was the, you know, opening that, you know, that Bobby Fischer used, and has chess notation to predict the next thing, it needs to know something about chess notation, or at least openings. If it's like, you know, and this was vetoed by the 1956 US president, comma, you need to know it's bit like, you don't even you don't just need to know who the president was, but the idea that there are presidents. And therefore, the idea that there are leaders, and therefore, the idea that there are groups of people who have hierarchies, and therefore, that there are people, and therefore, that there are objects, and like, you can't predict the next word of a sentence well without knowing all of these things. So that knowing my hypothesis for why I created ULMFiT was to say it would end to to to compress that as well as possible to get that knowledge, it would have

to create these abstractions, these hierarchies of abstractions somewhere deep inside its model. Otherwise, how could it possibly do a good job of predicting the next word? You know, and because deep learning models are universal learning machines, you know, and we had a universal way to train them, I figured if if we get the data right and if the hardware is good enough, then in theory, we ought to be able to build that next word predicting machine, which ought to implicitly build a hierarchical structural understanding of the things that are being described by the text that it is learning to predict?

Dr. Tim Scarfe 00:18:36

I think that they can know in quite a, you know, they they know in quite a superficial way. So there's a myriad of surface statistical relationships, and they generalize extraordinarily well.

Jeremy Howard 00:18:47

It's

Dr. Tim Scarfe 00:18:47

it's miraculous.

Jeremy Howard 00:18:48

It is.

Dr. Tim Scarfe 00:18:49

But the thing is, I want to contrast this with other comments you've made about creativity. So I think knowledge is about constraints. And I think creativity is the evolution of knowledge, respecting those constraints. Therefore, AI is not creative. And you've said the same thing. You've said AI isn't creative. So like, on the 1 hand, how can you say that they know and not think that they can be created?

Jeremy Howard 00:19:10

I mean, I don't think I've used that exact expression. You know, I know I've actually I remember chatting with Peter Norvig on camera, and both of us said, well, actually, they kind of are creative, like, we just gotta be a bit careful about our choices of words, I guess. So, you know, Piotr Wozniak, who's a guy I really really respect, who kind of rediscovered spaced repetition learning, built the SuperMemo system, and is the modern day guru of memory. The entire reason he's based his life around remembering things is because he believes that creativity comes from having a lot of stuff remembered, which is to say putting together stuff you've remembered in interesting ways is a great way to be creative. LLMs are actually quite good at that, but there's a kind of creativity they're not at all good at, which is, you know, moving outside the distribution. So which I think is where you're heading with your question. But I'm just kind of I'm framing it this way to say, you have to be so nuanced about this stuff because if you say like they're not creative, it gives you the can give you the wrong idea because they can do very creative seeming things. But if it's like, well, can they really extrapolate outside the training distribution? The answer is no, they can't. But the training distribution is so big, and the number of ways to interpolate between them is so vast, we don't really know yet what the limitations of that is. But I see it every day, you know, because I my my work is R and D. I'm constantly on the edge of and outside the training data. I'm doing things that haven't been done before. And there's this weird thing, I don't know if you've ever seen it before, I see it but I see it multiple times every day, where the LM goes from being incredibly clever to like worse than stupid, like like not understanding the most basic fundamental premises about how the world works. Yeah.[6][7]

Dr. Tim Scarfe 00:21:14

Yeah.

Jeremy Howard 00:21:14

And it's like, oh, whoops, I fell outside the training data distribution. It's gone dumb. And then, like, there's no point having that discussion any further because Yes. You know, you've lost it at that point.

[6] Modular Blog: Claude C Compiler analysis — Blog Post Analysis of Claude Code's C compiler implementation discussed during the Claude Code section.

[7] Anthropic Engineering Blog: Building C Compiler — Blog Post Anthropic's engineering blog post about Claude Code's C compiler implementation.

Dr. Tim Scarfe 00:21:26

Yes. I mean, I love, you know, Margaret Boden. She had this kind of hierarchy of creativity. So there's like combinatorial, exploratory and transformative. And the models can certainly do combinatorial creativity. But for me, it's all about constraints. That I mean, this is what Boden said. And even Leonardo da Vinci, he said that creativity is all about constraints. And you've spoken about we'll talk about this dialogue engineering. But what happens is when we talk with language models, it's a specification acquisition problem. So we go back and forth. And actually, when we think, the process of intelligence is about building this imaginary Lego block in our mind and respecting various constraints. And when you respect those constraints and you just continue to evolve, then those things are said to be creative. So language models, when you add constraints to them, so this could be via supervision, via critics, via verifiers, Then they are creative. And AlphaEvolve, we've seen many examples of this. But the illusion is, on their own, sans constraints obviously, they have this behavioral shaping stuff that we're talking about. They don't have hard constraints. And that's why they can't go outside their distribution.

Jeremy Howard 00:22:29

I mean, I think they can't go outside their distribution because it's just something that that type of mathematical model can't do. You know, I mean, it can do it, but it won't do it well. You know, when you look at the kind of 2 d case of fitting a curve to data, once you go outside the area that the data covers, the curves disappear off into space in wild directions, you know. And that's all we're doing, but we're doing it in multiple dimensions. Yep. I think Boden might be pretty shocked at how far compositional creativity can go when you can compose the entirety of the human knowledge corpus. And I think this is where people often get confused, because it's like So for example, I was talking to Chris Latner yesterday about how Claude Anthropic, you know, had had got Claude to write the C compiler. And they were like, oh, this is a clean room C compiler. You can tell it's clean room because it was created in Rust, you know, and so Chris created the kind of, you know, I guess it's probably the top most widely used c c plus plus compiler nowadays playing on top of LLVM, which is the most widely used kind of foundation for compilers. They're like, Chris didn't use rust. This is, you know, and we didn't give it access to any compiler source code. So it's a clean room implementation. But that misunderstands how LLMs work. Right? Which is all of Chris's work was in the training data. Many many times LLVM is used widely and lots and lots of things are built on it, including lots of c and c plus plus compilers. Converting it converting it to Rust is an interpolation between parts of the training data, you know. It's a style transfer problem. So it's definitely compositional creativity at most, if you can call it creative at all. And you actually see it when you look at the the repo that it created. It's copied parts of the LLVM code, which today Chris says like, oh, I made a mistake. I shouldn't have done it that way. Nobody else does it that way, You know? Oh, wow. Look. They're the only other 1 that did it that way. That doesn't happen accidentally. That happens because you're not actually being creative. You're actually just finding the kind of nonlinear average point in your training data between, like, Rust things and building compiler things?

Claude Code & The Interpolation Illusion ●

Jeremy Howard 00:22:29

I mean, I think they can't go outside their distribution because it's just something that that type of mathematical model can't do. You know, I mean, it can do it, but it won't do it well. You know, when you look at the kind of 2 d case of fitting a curve to data, once you go outside the area that the data covers, the curves disappear off into space in wild directions, you know. And that's all we're doing, but we're doing it in multiple dimensions. Yep. I think Boden might be pretty shocked at how far compositional creativity can go when you can compose the entirety of the human knowledge corpus. And I think this is where people often get confused, because it's like So for example, I was talking to Chris Latner yesterday about how Claude Anthropic, you know, had had got Claude to write the C compiler. And they were like, oh, this is a clean room C compiler. You can tell it's clean room because it was created in Rust, you know, and so Chris created the kind of, you know, I guess it's probably the top most widely used c c plus plus compiler nowadays playing on top of LLVM, which is the most widely used kind of foundation for compilers. They're like, Chris didn't use rust. This is, you know, and we didn't give it access to any compiler source code. So it's a clean room implementation. But that misunderstands how LLMs work. Right? Which is all of Chris's work was in the training data. Many many times LLVM is used widely and lots and lots of things are built on it, including lots of c and c plus plus compilers. Converting it converting it to Rust is an interpolation between parts of the training data, you know. It's a style transfer problem. So it's definitely compositional creativity at most, if you can call it creative at all. And you actually see it when you look at the the repo that it created. It's copied parts of the LLVM code, which today Chris says like, oh, I made a mistake. I shouldn't have done it that way. Nobody else does it that way, You know? Oh, wow. Look. They're the only other 1 that did it that way. That doesn't happen accidentally. That happens because you're not actually being creative. You're actually just finding the kind of nonlinear average point in your training data between, like, Rust things and building compiler

things?[8][9][10]

Dr. Tim Scarfe 00:25:09

All of that is true. I mean, first of all, I I think we shouldn't underestimate the size of how big this combinatorial creativity is. So all of that is true. So the code is on the Internet, but also, they had a whole bunch of tests which were scaffolded, which meant that every single time some code was committed, they could run the test and they basically had a critic. And they could then do this autonomous feedback loop. So in a sense, it's very similar to the recent research by OpenAI and Gemini, where you're trying to solve a problem in math and you already have an evaluation function, the same on the ARC prize, right? You have an evaluation function. And what people discount is even knowledge of what the evaluation function is, is partial knowledge of the problem. So you can then brute force search. You can use the statistical pattern matching. Use the verifier as a constraint, and you can actually

Jeremy Howard 00:25:58

they don't even need to do that. Right? Like, they literally already know how to pass those tests because there's lots of software that already does it. Right. So it just uses that and translates them to Rust. Like, that's that's all it did, which is impressive.

Dr. Tim Scarfe 00:26:14

Yeah.

Jeremy Howard 00:26:14

And if you I'm much less familiar with math than I am computer science, but from talking to mathematicians, they tell me that that's also what's happening with like, Erdős's problems and stuff. It's some of them are newly solved.

Dr. Tim Scarfe 00:26:31

Yeah.

[8] METR Study: AI OS Development — Research Study Study on AI software engineering capabilities referenced during the coding vs. engineering discussion.

[9] Fred Brooks: No Silver Bullet — Paper Classic software engineering essay on essential vs. accidental complexity.

[10] Oxford VGG: Sculptor Identification Paper — Research Paper Computer vision paper on sculptor identification, if referenced in the discussion.

Jeremy Howard 00:26:34

But they are not sparks of insight. You know, they're solving ones that you can solve by meshing up together very closely related things that humans have already figured out.

Dr. Tim Scarfe 00:26:47

So on the subject of Claude code. Now I know you've spoken extensively about vibe coding. Actually, Rachel had some interesting writeup. I mean, she she quoted the the METR study, which showed that productivity actually went down when people were vibe coding. But I think

Jeremy Howard 00:27:00

And they thought that they went up, which is the

Dr. Tim Scarfe 00:27:02

most

Jeremy Howard 00:27:02

interesting And

Dr. Tim Scarfe 00:27:03

then also there was the Anthropic study. Mean, know, maybe we should rewind a little bit. I mean, Dario had this essay out the other day. I think it's called the Adolescence of Technology or something like that. And then he was basically saying, look, you know, we have all of these amazing software engineers at Anthropic. And they are just so productive. And he was extrapolating to the average software engineer. So there's gonna be mass unemployment because soon, we're gonna be able to automate all of this with AI.

Jeremy Howard 00:27:26

I mean, it it doesn't make any sense. Elon Musk said something a bit similar a few days ago, saying like, oh, LLMs will just spit out the machine code directly. We won't need libraries, programming languages.

Coding vs. Software Engineering ●

Jeremy Howard 00:27:26

I mean, it it doesn't make any sense. Elon Musk said something a bit similar a few days ago, saying like, oh, LLMs will just spit out the machine code directly. We won't need libraries, programming languages.

Dr. Tim Scarfe 00:27:40

Yeah.

Jeremy Howard 00:27:42

Yeah. Look, the thing is none of these guys have have been software engineers recently. I'm not sure Dario's ever been a software engineer at all. Software engineering is a unusual discipline, and a lot of people mistake it for being the same as typing code into an IDE. Coding is another 1 of these style transfer problems. You take a specification of the problem to solve and you can use your compositional creativity to find the parts of the training data which interpolated between them solve that problem, and interpolate that with syntax of the target language, and you get code. There's a very famous essay by Fred Brooks written many decades ago, no silver bullet, and which it almost sounded like he was talking about today. It it he was specifically saying something he was just pointing something very similar, which is in those days it was all like, oh, what about all these new fourth generation languages and stuff like that, you know. We're not gonna need any coders anymore, any software engineers anymore, because software is now so easy to write, anybody can write it. And he said, well, he guessed that you could get at maximum a 30% improvement. He specifically said a 30% improvement in the next decade, but I don't think he needed to limit it that much. Because the vast majority of work in software engineering isn't typing in the code.

Dr. Tim Scarfe 00:29:12

Yep.

Jeremy Howard 00:29:15

So in some sense, parts of what Dario said were right, just like for quite a few people now, most of their code is being typed by a language model. That's true for me. Say, like, maybe 90%. But it hasn't made me that much more productive, because that was never the slow bit. It's also helped me with kind of the research a lot and figuring out, you know, which files are gonna be touched. But anytime I've made any attempt to getting an LLM to like design a solution to something that hasn't been designed lots of times before, it's it's horrible. Because what it actually, every time, gives me is the design of something that looks on its surface a bit similar. And often that's gonna be an absolute disaster, because things that look on the surface a bit similar and like I'm literally trying to create something new to get away from the similar thing. It's very misleading.[11]

Cosplaying Intelligence: Dennett vs. Searle ●

Jeremy Howard 00:29:15

So in some sense, parts of what Dario said were right, just like for quite a few people now, most of their code is being typed by a language model. That's true for me. Say, like, maybe 90%. But it hasn't made me that much more productive, because that was never the slow bit. It's also helped me with kind of the research a lot and figuring out, you know, which files are gonna be touched. But anytime I've made any attempt to getting an LLM to like design a solution to something that hasn't been designed lots of times before, it's it's horrible. Because what it actually, every time, gives me is the design of something that looks on its surface a bit similar. And often that's gonna be an absolute disaster, because things that look on the surface a bit similar and like I'm literally trying to create something new to get away from the similar thing. It's very misleading.[12][13]

Dr. Tim Scarfe 00:30:23

First of all, I'm I'm exasperated by what I see as the tech bro predilection to misunderstand cognitive science and philosophy and and what not. Because we've we've spoken to so many really interesting people on MLST like for example César Hidalgo, he wrote this book, The Laws of Knowledge. And and even Mazviita Chirimuuta, she's a a philosopher of neuroscience and she was talking all

[11]Daniel Dennett: Consciousness Explained — Book Dennett's book referenced in the consciousness and intelligence cosplaying discussion.

[12]Daniel Dennett: Consciousness Explained — Book Dennett's book referenced in the consciousness and intelligence cosplaying discussion.

[13]John Searle: Minds, Brains, and Programs — Paper The Chinese Room argument paper referenced in the AI consciousness debate.

about, you know, like flipping basically that knowledge is protean. So yeah, I think that knowledge is perspectival. I don't think that knowledge can be this abstract perspective free thing that can exist on Wikipedia. And I also think that knowledge is embodied and it's alive. It's something that exists in us. And the purpose of an organization is to preserve and evolve knowledge. So when you start delegating cognitive tasks to language models, you actually have this weird paradoxical effect that you erode the knowledge inside the organization.

Jeremy Howard 00:31:16

Well, that's true. And that's terrifying. There's often these these arguments online between people who are like, LLMs don't understand anything. They're just pretending to understand. Mhmm. And then other people are like, don't be ridiculous. Look what this LLM just did for me. Right? And the funny thing is they're both right. LLMs cosplay understanding things. Like, they pretend to understand things. And this is the interesting thing about the early kind of work with, like, cognitive science work with, like, Daniel Dennett. That's basically what the Chinese room experiment is. Right? It is you've got a guy in a room who can't speak Chinese at all, but he sure looks like he does because you can feed in questions and he gives you back answers, but all he's actually doing is looking up things in a huge array of books or machines or whatever. The difference between pretending to be intelligent and actually being intelligent is entirely unimportant, as long as you're in the region in which the pretense is actually effective, you know. So so it's actually fine for a great many tasks that LLMs only pretend to be intelligent, because for all intents and purposes, it it it just doesn't matter until you get to the point where it can't pretend anymore. And then you realize, like, oh my god. This thing's so stupid.

Dr. Tim Scarfe 00:32:41

I'm a fan of Searle, by the way. So, know, he said that understanding is causally reducible but ontologically irreducible. And he was saying there was a phenomenal component to understanding. You don't even need to go there. Like the interesting thing about knowledge being protean is this idea that, you know, it's basically this Kantian idea. The world is a complex place. None of us understand it. It's like the blind men and the elephant. We all have different perspectives. It's a very complex thing. And so we all we all do this kind of modeling. But the the interesting thing is that the language model, sometimes they seem to understand. And they understand because the supervisor places them in a frame. So inside that frame, so when you have that perspective of the elephants, they're actually surprisingly coherent. But we discount the supervisor placing the models in that frame.

Jeremy Howard 00:33:25

Yeah. Yeah. So that so Searle versus Dennett, or is it versus Searle and Dennett, was what everybody was talking about back when I was doing my undergrad in philosophy, you know, so and Consciousness Explained came out about then, probably Chinese Room a little bit before. That's interesting because the discussions were the same discussions we're having now, but they've gone from being abstract discussions to being real discussions. It's helpful if people go back to the abstract discussions, because it helps you get out of your you know, it's very distracting at the moment to look at something that's cosplaying intelligence so well, and go back to the fundamental question. So anyway, I just wanted to mention that's kind of it's it's this interesting situation we're now in, where it's very easy to really get the wrong idea about what AI can do. Particularly when you don't understand the difference between coding and software engineering. Yeah. Which then takes me to your point or your question about the implications of that for organizations.

Dr. Tim Scarfe 00:34:40

Yep.

Jeremy Howard 00:34:41

You know, a lot of organizations are basically betting their futures on a speculative premise, which is that AI is gonna be able to do everything better than humans, or at least everything in coding better than humans. I I worry about this a lot, both for the organizations and for the humans, you know. For the humans, when you're not actively using your design and engineering and coding muscles, you don't grow. You might even wither. But you at least don't grow. And, you know, speaking of the CEO of an r and d startup, you know, if if my staff aren't growing, then we're gonna fail. You know? We that we can't let that happen. And getting better at the particular prompting skills, whatever details of the current generation of AI, CLI frameworks isn't growing. You know, that's that's like that's as helpful as learning about the details of some AWS API when you don't actually understand how the Internet works, you know. It's not it's not reusable knowledge. It's ephemeral knowledge. So like, if you wanted to, you can actually use it as a learning superpower. But also, it can do the opposite. You know, the natural thing it's gonna do is remove your confidence over time.

Dr. Tim Scarfe 00:36:15

I agree that that's the natural thing. And this this is especially pertinent for you because your your career has been around basically educating people to to get, you know, technology and AI literacy. So the default behavior is very similar to a self driving car. But there's this tipping point where at some point, you're not engaged anymore. You're not paying attention. And you get this delegation of competence. And you get understanding debt. That's the default thing. So this study from Anthropic a couple of weeks ago, it contradicted Dario completely because it even said that, yeah, there were a few people in the study that were asking conceptual questions that are actually kind of keeping on top of things. And they had a gradient of learning, but most people didn't. And my hypothesis about that is that the ideal situation for Gen AI coding is that like us, we've been writing software for decades. We already have this abstract understanding. We're using it in domains that we know well. And we can specify, we can remove loads of ambiguity, we can track and we can go back and forth and we can we can stay in touch with the process. But what happens is that the the default attractor is for people to just go into this autopilot mode and they've got no idea what's happening and it's actually making them dumber.[14]

Automation, Radiology & Desirable Difficulty ●

Dr. Tim Scarfe 00:36:15

I agree that that's the natural thing. And this this is especially pertinent for you because your your career has been around basically educating people to to get, you know, technology and AI literacy. So the default behavior is very similar to a self driving car. But there's this tipping point where at some point, you're not engaged anymore. You're not paying attention. And you get this delegation of competence. And you get understanding debt. That's the default thing. So this study from Anthropic a couple of weeks ago, it contradicted Dario completely because it even said that, yeah, there were a few people in the study that were asking conceptual questions that are actually kind of keeping on top of things. And they had a gradient of learning, but most people didn't. And my hypothesis about that is that the ideal situation for Gen AI coding is that like us, we've been writing software for decades. We already have this abstract understanding. We're using it in domains that we know well. And we can specify, we can remove loads of ambiguity, we can track and we can go back and forth and we can we can stay in touch with the process. But what happens is that the the default attractor

[14] Anthropic Paper: AI Skill Formation — Research Paper Paper on AI learning and skill formation referenced during the automation discussion.

is for people to just go into this autopilot mode and they've got no idea what's happening and it's actually making them dumber.[15][16]

Jeremy Howard 00:37:19

I I created a the the first deep learning for medicine company called Enlitic back in, what was that, like, '14. And our initial focus was on radiology, and a lot of people were worried that this would cause radiologists to become less effective at radiology. And I strongly felt the opposite, which is and I did quite a bit of research into this, of like what happens when there's like fly by wire in airplanes or anti lock brakes in cars or whatever. If you can successfully automate parts of a task that really are automatable, you can allow the expert to focus on the things that they need to focus on. And we saw this happen. So in radiology we found if we could automate identifying the possible nodules in a lung CT scan, we were actually good at it, which we were, and then we've the radiologist then can focus on looking at the nodules and trying to decide if they're malignant or what to do about it. So again, it's 1 of these subtle things. So if there's things which you can fully automate effectively in a way that you can remove that cognitive burden from a human, so that they can focus on things that they need to focus on, that can be good, you know. I don't know where we sit in software development because, you know, I've been coding for 40 ish years. So I've written a lot of code and I can glance at a screen of code and then, you know, unless it's something quite weird or sophisticated, I can immediately tell you what it does and whether it works and whatever. I can kind of see intuitively things that could be improved, you know, possible things to be careful of. I'm not sure I could have got to that point if I hadn't have written a lot of code. So the people I'm finding who can really benefit from AI right now are either really junior people who can't code at all, who can now write some apps that they have in their head. And as long as they work reasonably quickly with the current AI capabilities, then they're happy. And then really experienced people like like me or like Chris Latner, because we can basically have it do some of our typing for us, you know, and some of our research for us. People in the middle, which is most people, most of the time, it really worries me because how do you get from 0.1 to point b

Dr. Tim Scarfe 00:40:04

Yeah.

[15] Anthropic Paper: AI Skill Formation — Research Paper Paper on AI learning and skill formation referenced during the automation discussion.

[16] Ebbinghaus: Memory / Spaced Repetition — Historical Reference Ebbinghaus' work on memory and spaced repetition referenced in the desirable difficulty discussion.

Jeremy Howard 00:40:05

Without typing code? It might be possible, but we don't have a we have no experience of that. We don't is is it possible? How would you do it? Like, is it kind of like going back to school where at primary school we don't let kids use calculators so that they develop their number muscle. Do we need to do that for like first 5 years as a developer? You have to write all the code yourself. I I don't know. But if I wasn't between, like, 2 and 20 years of experienced developer, I would be asking that question of myself a lot. Because otherwise, you might be in the process of making yourself obsolete.

Dr. Tim Scarfe 00:40:48

Yeah. Well, this is another thing about knowledge that this Cesar Hidalgo guy said. So he said that knowledge is non fungible, which means it can't be exchanged. So what he means by that is the process of learning is in some important sense not reducible. So you have to have the experience. And the experience has to have friction. And when we build models of the world, we actually learn like there's this phrase, reality pushes back. So we make lots of mistakes, and we update our models, and we're just placing these coherence constraints in our model. And that's how we come to learn. So you use called code, and there's so little friction in the process. That's exactly what this study from Anthropics said. It said there was so little friction, they didn't learn anything.

Jeremy Howard 00:41:29

Right. Yeah. No. Exactly. Desirable difficulty is the concept that kinda comes up in education. But even going back to the work of Ebbinghaus, who was the original repetitive space learning guy in the nineteenth century, And then Piotr Wozniak more recently, we find the same like, we we we know that memories don't get formed unless it is hard work to form them. So, you know, that's where you kind of get this somewhat surprising result that says revising too often is a bad idea, because it comes to mind too quickly. And so with repetitive spaced learning, with stuff like Anki and SuperMemo, the algorithm tries to schedule the flashcards at a just before the moment you're about to forget. So then it's hard work. So I I studied Chinese for 10 years in order to try to learn about learning myself. And I really noticed this that I used Anki, and because it was always scheduling my cards just before I was about to forget them, it was always incredibly hard work.[17]

[17] Cesar Hidalgo: Infinite Alphabet / Laws of Knowledge — Book Book on organizational knowledge referenced during the knowledge embodiment discussion.

Organizational Knowledge & The Slope ●

Jeremy Howard 00:41:29

Right. Yeah. No. Exactly. Desirable difficulty is the concept that kinda comes up in education. But even going back to the work of Ebbinghaus, who was the original repetitive space learning guy in the nineteenth century, And then Piotr Wozniak more recently, we find the same like, we we we know that memories don't get formed unless it is hard work to form them. So, you know, that's where you kind of get this somewhat surprising result that says revising too often is a bad idea, because it comes to mind too quickly. And so with repetitive spaced learning, with stuff like Anki and SuperMemo, the algorithm tries to schedule the flashcards at a just before the moment you're about to forget. So then it's hard work. So I I studied Chinese for 10 years in order to try to learn about learning myself. And I really noticed this that I used Anki, and because it was always scheduling my cards just before I was about to forget them, it was always incredibly hard work.[18]

Dr. Tim Scarfe 00:42:43

Yeah.

Jeremy Howard 00:42:43

You know, to do reviews, because almost all the cards were once I was on the verge of forgetting. That was absolutely exhausting. But my God, it worked well. Here I am, I don't really haven't done any study for 15 plus years, and I still remember my Chinese.[19]

Dr. Tim Scarfe 00:43:00

Well, I mean, also, coming back to your radiology example, 1 example people give is call centers. So we have this notion that in an organization, we have high intelligence roles and low intelligence roles. And for me, intelligence is just the adaptive acquisition and synthesis of knowledge. So we assume that the low intelligence roles doing the call center stuff, it doesn't adapt, which means we can there are certain things that an organization does that do not change. So we could automate them and we don't need to update our knowledge. And I think that discounts actually maybe with the radiology example that having this holis-

[18]Cesar Hidalgo: Infinite Alphabet / Laws of Knowledge — Book Book on organizational knowledge referenced during the knowledge embodiment discussion.
[19]John Ousterhout: Slope vs Intercept — Technical Note Referenced for the ”slope makes up for y-intercept” career growth concept.

tic knowledge. Like, you know, in a call center, there are so many weird edge cases that come in. So many weird things happen and that filters up in the organization and we adapt over time. So when you start to automate things, and you actually lose the competence to create the process which created the thing in the first place, and you lose the evolvability of that knowledge in the organization, you're actually kind of cutting your legs off.

Jeremy Howard 00:43:53

Yeah, absolutely. And so, you know, all I know is in my company, I just I tell our staff all the time, almost the only thing I care about is how much your your personal human capabilities are growing. You know, I I don't actually care how many PRs you're doing, how many features you're doing. Like, there's that nice, you know, John Ousterhout, the Tcl guy, recently released some of his Stanford Friday takeaway lectures, and he has this nice 1 called a little bit of slope makes up for a lot of intercept. Just basically the idea that that, you know, in your life, if you can focus on doing things that cause you to grow faster

Dr. Tim Scarfe 00:44:44

Yeah.

Jeremy Howard 00:44:45

It's way better than focusing on focusing on the things that you're already good at, you know, that has that high intercept. So the only thing I really care about, and I think is the only thing that matters for my company, is that my team are focusing on their slope. Yeah. If you focus on just driving out results at the limit of whatever AI can do right now, you're only caring about the intercept, you know. So I think it's basically a path to obsolescence through both the company and the people who are in it. And so I'm really surprised how many executives of big companies are pushing this now, because it feels like if they're wrong, which they probably are, and they have no way to tell if they are because this is an area they're not at all familiar with and never learned it in their MBAs. They're basically setting up their companies to be destroyed.

Dr. Tim Scarfe 00:45:37

Yeah.

Jeremy Howard 00:45:38

And really surprised that, you know, shareholders would let them do that. It'll set up such an incredibly speculative action. Yeah. Here we are. It feels like a lot of companies are gonna fail as a result of the amassed tech debt that causes them to not be able to maintain or build their products anymore.

Dr. Tim Scarfe 00:45:58

There are loads of folks out there like François Chollet. Like, he he he really gets it. He he

Jeremy Howard 00:46:02

understands this. Yeah.

Dr. Tim Scarfe 00:46:03

And, you know, so he's always said that it it's it's about this kind of mimetic sharing of cognitive models about the domain and how we refine it together. On the sharing thing, this is another big scaling problem with Gen AI coding, right? So the ideal case, I've done this. I know a domain really well and I can specify it with exquisite detail and I tell Claude code, go and do this thing and the models in my mind doesn't matter. And then you go into an organization and now I need to share like my knowledge with all of the other people, right? And I'm sure you have this in your company as well. You need to that that this knowledge acquisition bottleneck is a real serious problem in in organizations. So when it's just me, I I think I'm probably about 50 times more productive using Claude Code. It's absolutely magic. And I can see why people are so excited about it. But people don't seem to understand the bottleneck and and how that doesn't really translate to many real world organizations.

Jeremy Howard 00:46:54

No one's actually creating 50 times more high quality software than they were before. So we've actually just done a study of this, and there's a tiny uptick, tiny uptick in what people are actually shipping. That's the facts. Obviously, I'm an enthusiast of AI and what it can do. But also, my wife Rachel recently pointed out in an article, all of the pieces that make gambling addictive are present in

Dr. Tim Scarfe 00:47:30

Oh, yeah. Dark flow. Yeah. I was gonna bring that up.

Jeremy Howard 00:47:33

Yeah. It's this really awkward situation where it's very almost everybody I know who got very enthusiastic about AI powered coding in recent months have totally changed their mind about it when they finally went back and looked at, like, how much stuff that I built during those days of great enthusiasm am I using today? Are my customers using today? Am I making money from today? Almost all the money is being made by influencers, you know, or by the companies that produce the tokens. The thing about AI based coding is that it's like a slot machine, and that you you have an illusion of control, you know, you can get to craft your prompt, and your list of MCPs, and your skills, and whatever, and then in the end, you pull the lever. Right? You put in the prompt, and something comes back, and it's like cherry, cherry, it's like, oh, next time I'll change my prompt a bit, I'll add a bit more context, pull the lever again, pull the lever again. It's the stochastic thing. You get the occasional win. It's like, oh, I won. I got a feature. So it's got it's got all these hallmarks of like, loss disguised as a win, somewhat stochastic, feeling of control, all the stuff that gaming companies try to engineer into their gaming rooms. Now, none of that means that AI is not useful, but gosh, it's hard to tell.[20]

Vibe Coding as a Slot Machine

Jeremy Howard 00:47:33

Yeah. It's this really awkward situation where it's very almost everybody I know who got very enthusiastic about AI powered coding in recent months have totally changed their mind about it when they finally went back and looked at, like, how much stuff that I built during those days of great enthusiasm am I using today? Are my customers using today? Am I making money from today? Almost all the money is being made by influencers, you know, or by the companies that produce the tokens. The thing about AI based coding is that it's like a slot machine, and that you you have an illusion of control, you know, you can get to craft your prompt, and your list of MCPs, and your skills, and whatever, and then in the end, you pull the lever. Right? You put in the prompt, and something comes back, and it's like cherry, cherry, it's like, oh, next time I'll change my prompt a bit, I'll add a bit more context, pull the lever again, pull the lever again. It's the stochastic thing. You get the occasional win. It's like, oh, I won. I got a feature. So it's got it's got all these hallmarks of like, loss disguised as a win, somewhat stochastic, feeling of control, all the stuff

[20]Cursor Blog: Scaling Agents — Blog Post Referenced during the AI coding tools and vibe coding discussion.

that gaming companies try to engineer into their gaming rooms. Now, none of that means that AI is not useful, but gosh, it's hard to tell.[21]

Dr. Tim Scarfe 00:49:12

I know. And and Rachel, just just to be clear as well, she she also said that 1 of the hallmarks of gambling is that you kind of delude yourself that you have some awareness of what's going on, but but actually you don't. But let let's do the bull case a little bit though. So I do I do think in restricted cases, it it is it is very useful. And these are cases where we understand and we and we can place constraints and specification. But even in those cases, could argue on the 1 hand that we're not, you know, we're not gonna be unemployed anytime soon because you just do more work. On the addiction thing, I've noticed that. So I've had 14 hour Claude code marathon sessions and and I actually feel addicted to it. It's like a slot machine. You know, it it really is.

Jeremy Howard 00:49:49

And there too. Absolutely.

Dr. Tim Scarfe 00:49:51

Yeah. Know. It's and it just I've never felt more drained writing code. I actually need to take a rest afterwards, like a few days rest because it completely

Jeremy Howard 00:49:59

crap, you know. Yeah. Definitely. I've had some successes. Right? And so, in fact, we've spent the last couple of years building a whole product based around where we know the successes are gonna be, which is when you're working on reasonably small pieces that you can fully understand, and that you can design, and you can build up your own layers of abstraction to create things that are bigger than the parts that you're building out of. Had a very interesting situation recently where I just it's kind of an experiment basically, which is we we rely very heavily on something called IPy kernel, which is the thing that powers Jupyter notebooks. And there had been a major version release of IPykernel from '6 to '7, and it stopped working. And it stopped working in both of the products that we were trying to use it with. 1 was was called nbclassic, which is the original Jupyter notebook. And then our own product called solve it. They would just randomly crash. And Ipy kernels over 5,000 lines of code. It's very complex code, multiple threads, events, locks, interfaces with IPython, you know, with ZMQ, you know,

[21] Cursor Blog: Scaling Agents — Blog Post Referenced during the AI coding tools and vibe coding discussion.

all kinds of different pieces, DebugPy, and I I couldn't get my head around it, and I couldn't see why it was crashing. The tests are all passing. I wonder if AI can solve this. You know, it's like I'm always interested in the question of like how big a chunk can AI handle on its own right now. The answer turned out to be yes. I think it can just. It was like so I spent a couple of weeks, I didn't develop a lot of understanding about how IPykernel really worked in the process, but I did spend quite a bit time kind of pulling out separate comp like, so the answer was in 2 hours, codecs 5 point I think it was 5.2 at that time or maybe 3 had just come out, couldn't do it. Then if I got the $200 a month GPT 5.3 pro to fix the problems it could. And so by rolling back between those 2 pieces of software and those 2 models, I could get things working over a couple of weeks period. And like you say, it wasn't at all fun. It was very tiring and it felt stressful because I wasn't really in control. But the interesting thing is I now am in a situation where I have the only implementation of an of a Python Jupyter kernel that actually works correctly, as far as I can tell, with these new version 7 protocol improvements. And now I'm like, well, this is fascinating because we don't have a kind of a software engineering theory of what to do now. It's like, here's a piece of code that no 1 understands.

Dr. Tim Scarfe 00:53:02

Yeah.

Jeremy Howard 00:53:03

Am I going to bet my company's product on it? And I the answer is I don't know. Because like I I don't I don't like I don't know what to do now, because no one's like being in this situation, and like, it does it have memory leaks? Will it still work in a year's time if there's some minor change to the protocol? Is there some weird edge case that's gonna destroy everything? No 1 knows because no 1 understands his code. It's a really curious situation.

Dr. Tim Scarfe 00:53:36

I mean, first of all, we should acknowledge the pernicious erosion of control. So at the very beginning, you have 10% AI generated code, and then you can just see how it creeps up and up. And then at some 0.6 months down the line the PR comes in and now you know 60% of the code is AI generated and do you see what happens? You just you slowly become disconnected but the bull case for this is you know in AI there's this idea called functionalism that you know we don't care what the intelligent thing is made out of as long as it does all of the right things, then we know, you know, we would say it's AI. And it's the same thing with software. So the bull case is, I I understand the domain. I don't need to write. I don't need to know how to write the quick sort algorithm. I just need to understand it. Right? And then and then, you know, so I just need to have all of these tests. And it needs to go into deployment. And these things need to happen. And at that point, you know what? I don't actually care. And I and I could also

The Erosion of Control in Software

Dr. Tim Scarfe 00:53:36

I mean, first of all, we should acknowledge the pernicious erosion of control. So at the very beginning, you have 10% AI generated code, and then you can just see how it creeps up and up. And then at some 0.6 months down the line the PR comes in and now you know 60% of the code is AI generated and do you see what happens? You just you slowly become disconnected but the bull case for this is you know in AI there's this idea called functionalism that you know we don't care what the intelligent thing is made out of as long as it does all of the right things, then we know, you know, we would say it's AI. And it's the same thing with software. So the bull case is, I I understand the domain. I don't need to write. I don't need to know how to write the quick sort algorithm. I just need to understand it. Right? And then and then, you know, so I just need to have all of these tests. And it needs to go into deployment. And these things need to happen. And at that point, you know what? I don't actually care. And I and I could also

Jeremy Howard 00:54:33

And I And to be clear, I quite like that framing. But you know, what that actually does is it says, well, software engineering sure is important then. Because software engineering is all about finding what those pieces are, and how they should behave, and then how you can put them together to create a bigger piece, and then how you can put them together to create a bigger piece. And if we do that well, then in 10 years time, we could have software that is far more capable than anything we could even imagine today.

Dr. Tim Scarfe 00:55:02

But

Jeremy Howard 00:55:05

you're only gonna get that with really great software engineering. Yeah. You wanna be careful. I think in the end like IPykernel, I'm finding for example, it's just too big a piece, right? Because in the end, the the team that made the original IPykernel were not able to create a set of tests that correctly exercised it, and therefore real world downstream projects, including the original nb classic, you know, which is what IPykernel was extracted from, didn't work anymore. So this is this is kind of where our focus is on now on the development side at Answer dot ai, is finding the right sized pieces and making sure they're the right pieces. Knowing how to recognize what those pieces are, and how to design them, and how to put them together is actually something that normally requires some decades of experience before you're really good at it. Certainly, it's true for me. I reckon I got pretty good at it after maybe 20 years of experience. Yeah. It's a big question. It's like how do you build these software engineering chops which are now even more important than they've ever been before, they're the difference between somebody who's good at writing computer software and somebody who's not. That feels like a challenging question.

Dr. Tim Scarfe 00:56:32

I know. And there's also this notion that there are so many different ways to abstract and represent something. Know, the world is a very complex place. And maybe the way we've been abstracting and representing software is mostly a reflection of our own cognitive limitations, right? And even in the sciences and in physics, you tend to have a lot of quite reductive methods of modeling the world. And then you've got complexity science, is just embracing the constructive, dissipative, gnarly nature of things. And I think a lot of software today, we don't understand. Right? So for example, there are many globally distributed software applications that use the actor pattern. And this is just this ins it's basically like a complex system. Right? And the only way we can understand it is by doing simulations and tests because no 1 actually knows how all of these things fit together. So you could argue, I guess, as a bull case that maybe we already are doing this at the top of software engineering, and that is what we want to do eventually anyway.

Jeremy Howard 00:57:30

Yeah. I'd say probably not. You see companies like Instagram and WhatsApp dominate their sectors whilst having 10 staff, and beating companies like Google and Microsoft in the process. I would argue this way of building software in very large companies is actually failing. And I think we're seeing a lot of these very large companies becoming, you know, increasingly desperate. And, you know, for example, the quality of Microsoft Windows and Mac OS has very obviously deteriorated greatly in the last 5 to 10 years. You know, back when Dave Cutler was looking at every line of the NT kernel and making sure it was beautiful, it was a elegant and marvelous piece of software, you know. And this I don't think there's anybody in the world who's gonna say that Windows 11 is an elegant and marvelous piece of software. So I actually think we do need to find these smaller components that we do fully understand, and that we need to build them up. And here's the problem. AI is no good at that. So and and so I say that empirically. They're really bad at software engineering. And then I think that's possibly always gonna be true, because, you know, we're we're asking them to often move outside of their training data, you know, if we're trying to build something that literally hasn't been built before and do it in a better way than has been done before, we're saying, like, don't just copy what was in the training data. So and again, this is a confusing point for a lot of people, because they see AI being very good at coding. And then you think like, that's software engineering. You know, it's like, it must be good at software engineering. But it's they're different tasks. There's not a huge amount of overlap between them. And there's no current empirical data to suggest that LLMs are gaining any competency at software engineering. Every time you look at a piece of software engineering they've done, like the browser, for example, which Cursor created, or the C compiler, which Anthropic compared created. Like I've read the source code of those things quite a bit. Chris Latner is much more familiar with the compiler example than me. But they're they're very very obvious copies of things that already exist. So that's the challenge, you know, is if you want to build something that's not just a copy, then you can't outsource that to an LLM. There's no theoretical reason to believe that you'll ever be able to, And there's no empirical data to suggest that you'll ever be able to.[22]

Dr. Tim Scarfe 01:00:34

Yes. I think the punch line of this conversation is, and I'm sure you would agree of this, that we need to have the combination of AI and humans working together. Right? Because Right. The humans provide the understanding and all of the stuff we were saying about knowledge. But we can still use AIs as a tool. But we to

[22] Bret Victor: Inventing on Principle — Video Referenced during the interactive programming and REPL environments discussion.

design operating models or ways of working that make that we say we don't want to diminish our competence and understanding. Right. So it's very it's a very fine

Interactive Programming & REPL Environments

Dr. Tim Scarfe 01:00:34

Yes. I think the punch line of this conversation is, and I'm sure you would agree of this, that we need to have the combination of AI and humans working together. Right? Because Right. The humans provide the understanding and all of the stuff we were saying about knowledge. But we can still use AIs as a tool. But we to design operating models or ways of working that make that we say we don't want to diminish our competence and understanding. Right. So it's very it's a very fine

Jeremy Howard 01:01:03

line. That's that's been our focus, and we both focus on that for teaching and for our own internal development. The stuff I've been working on for 20 years has turned out to be the thing that makes this all work. Stephen Wolfram should get credit for this. He was the guy that created the notebook interface. Although also lots of ideas kind of go back to Smalltalk and Lisp and APL. But basically, the idea that a human can do a lot more with a computer when the human can, like, manipulate the objects in inside that computer in real time, study them, and move them around, and combine them together. Yeah. That's what small talk was all about, you know, with objects, and APL was the same with arrays. Mathematica basically is a super powered Lisp, which then also added on this very elegant notebook interface that allowed you to construct kind of a living document out of all this. So I built this thing called nbdev a few years ago, which is a way of creating production software inside these notebook interfaces, inside these rich dynamic environments. And I found that made me dramatically more productive as a programmer. And like today, even though I've never been a full time programmer as my job, when you look at my kind of GitHub repo output, I think GitHub produced some statistics about it, and I was like, just about the most productive programmer in in Australia. You know, like it it's working. And a lot of the stuff I build has lots and lots of people use it, because it's such a rich, powerful way to build things. And so it turns out, we've now discovered that if you put AI in the same environment with a human, again, in a in a rich, interactive environment, AI is much better as well, which perhaps isn't shocking to hear. But the normal, like, you use Claude code, which I know you do and it's a very good piece of software, but the environment we give Claude code is very similar to the environment that people had 40 years ago, you know. It's a it's a line based terminal interface. You know, it can use MCP or whatever. Most of the times, it just nowadays uses bash tools, which again, very powerful. I love bash tools. I use them all the, you know, the CLI tools all the time. But it's still just it's using text files, you know, as its as its interface to the world. It's it's it's really meager. So so we put the human and the AI inside a Python interpreter. And now suddenly you've got the full power power of a very elegant expressive programming language that the human can use to talk to the AI. The AI can talk to the computer. The human can talk to the computer. The computer can talk to the AI. Like, you have this really rich thing, then we let the human and the AI in real time build tools that each other can use. And that's what it's about to me. Right? It's about, like, creating an environment where humans can grow and engage and share. It's like for me, when I use Solveit, it's the opposite of that experience you described with Claude Code. After a couple of hours, I feel energized and happy and fulfilled.

Dr. Tim Scarfe 01:04:38

I'll give you my take. I think that the thing that you're pointing to here is there's something magic about having an interactive, stateful environment that gives you feedback.

Jeremy Howard 01:04:49

And

Dr. Tim Scarfe 01:04:50

that is because our brains kind

Jeremy Howard 01:04:52

of

Dr. Tim Scarfe 01:04:52

they can do a certain unit of work. So we actually think through refining and testing with reality. And that's why, I mean, during my PhD, I used Mathematica and MATLAB. And I agree. So we've got this REPL environment and, you know, here's the matrix, let's do an image plot, you know, do a change. This is what it looks like now. And it's actually a wonderful way to kind of just just refine my mental model about something. But Claude code does a lot of this stuff. I think it's mostly a skill issue. I think the people that use Claude codes effectively do this. I've written a content[23]

[23] Joel Grus: I Don't Like Notebooks — Video The famous JupyterCon talk critiquing notebooks referenced in the notebook debate.

The Notebook Debate & Exploratory Science ●

Dr. Tim Scarfe 01:04:52

they can do a certain unit of work. So we actually think through refining and testing with reality. And that's why, I mean, during my PhD, I used Mathematica and MATLAB. And I agree. So we've got this REPL environment and, you know, here's the matrix, let's do an image plot, you know, do a change. This is what it looks like now. And it's actually a wonderful way to kind of just just refine my mental model about something. But Claude code does a lot of this stuff. I think it's mostly a skill issue. I think the people that use Claude codes effectively do this. I've written a content[24][25]

Jeremy Howard 01:05:26

management

Dr. Tim Scarfe 01:05:26

It's

Jeremy Howard 01:05:27

possible. It is

Dr. Tim Scarfe 01:05:28

possible. It is possible. Yeah. So I've written a content management system called Rescript. And when I'm putting together documentary video, it can pull transcripts. And then I can verify the claims. And part of AI literacy is just understanding the asymmetry of language models, right? So when you give them a sort of discriminative task, they're actually quite good. So if I tell it in a sub agent to go and verify every individual claim, it's much more accurate than if I was in generation mode and I was generating a bunch of claims. And the stateful feedback thing, again, I can have some kind of schematized XML dump. And I can have like, an application here on the side, which is visualizing, and it's like a feedback loop. And for me, this is an AI literacy thing. Like, the the good people at AI are already doing this.

[24] Joel Grus: I Don't Like Notebooks — Video The famous JupyterCon talk critiquing notebooks referenced in the notebook debate.
[25] fast.ai Blog: NB Dev Merged Driver — Blog Post Referenced during the notebooks and nbdev discussion.

Jeremy Howard 01:06:12

Yeah. So I don't fully agree with you. I agree you can do it in Claude code, and I agree it is a AI literacy thing as to whether you can. But also Claude code was not designed to do this. It's not very good at it, and it doesn't make it the natural way of working with it. I don't wanna say it's an AI literacy problem, because that's like saying like, oh, it's a you problem. To me, if a tool is not making it the natural way for a human to become more knowledgeable, more happy, more connected, with a deeper understanding and a deeper connection to what they're working on, that's a tool problem. That that should be how tools are designed to work. So so many models and tools expressly are being evaluated on can I give it a complete piece of work and have it go away and do the whole thing, which feels like a huge mistake to me, versus have you evaluated whether a human comes out the other end with a deep understanding of a topic, you know, so that they can really easily build things in the future?

Dr. Tim Scarfe 01:07:23

I agree with all of that. But then there's the other interesting angle, which is that there was a famous talk by Joel Grus, and we'll talk about this. And and he said that notebooks are terrible. They're they're really bad from a software engineering point of view. And and at the time, and maybe still now to a certain extent, I I agree with him because, you know, I've I've I've done ML DevOps. I've worked in large organizations, you know, like trying to figure out how do we bridge like data science and software engineering. And Claude code is already more towards the software engineering side and what that means is it creates idempotent, stateless, repeatable artifacts, right? So as you say, from a pedagogical point of view, it's really good having this stateful feedback because I can understand what's going on. But then I need to translate that into something which is deployable. And Can you tell us the story of you you you responded to Joel Grus, didn't you? And and it was a bit of a fiasco, wasn't it? But what just just tell us about that story.

Jeremy Howard 01:08:19

He did a really good video called I don't like notebooks. It was hilarious. It was really well done. And, yeah, I was totally wrong. And all the things he said notebooks can't do, they can. And all the things he said you can't do with notebooks, I do with notebooks all the time. So it was a very good, amusing incorrect talk. So then I did a kind of a parody of it called I like notebooks, in which I basically copied with credit most of his slides and showed how every 1 of them was totally incorrect. But, like, I actually think your comment about it does come down to the heart of it, which is this this this difference between, like, how software engineering is normally done versus how scientific research and similar things is normally done. And I think and I agree there is a dichotomy there. And I think that dichotomy is a real shame because I think software development is being done wrong. It's being done in this way, which is, yeah, all about reproducibility and these like dead these dead pieces, you know. It's all dead code, dead files. I will never be able to express this 1 millionth as clearly as Brett Victor has in his work, so I'd encourage people who haven't watched Brett Victor to to watch him. But, you know, he he shows again and again how a direct connection, you know, a direct visceral connection with the thing you're doing is is all that matters, you know. And that's his mission, is to make sure people have that connection. And that's basically my mission as well. So for me, traditional software engineering is as far from that as it is possible to get. I think it's I think it's gross. Like, I I I find it disgusting. And I find it sad that people are being forced to work like that. It's like, I think it's inhumane. And I just don't think it works very well. I mean, empirically, it doesn't work very well. And it's much less good for for AI as well, as it's much less good for humans. It hasn't always been that way. Like, you know, with with Alan Kay and Smalltalk and Iverson and APL, you know, Lisp Wolfram with Mathematica. To me, these were the golden days when when people were focused on the question of how do we get the human into the computer to work as closely with it as possible. You know, that's where the the mouse came from, for example. I've got to like click and drag and visualize entities in your computer as things you can move around. So I feel like we've lost that. I think it's really sad. Yeah. With Claude code and stuff, the the default way of working with them is to go super deep into it. It's like, okay, there's a whole folder full of files, you never even look at them, your entire interaction with it is through a prompt.

Dr. Tim Scarfe 01:11:44

Yeah.

Jeremy Howard 01:11:45

I it it literally disgusts me. Like, I literally think it's it's inhumane, and it's my mission remains the same as it has been for, like, 20 years, which is to stop people working like this.

Dr. Tim Scarfe 01:12:00

I I know. But the so casting my mind back. I used to work with data scientists. They were using Jupyter Notebooks. And what I found was typically, I mean, back then, you couldn't if you check them into Git, it wouldn't look very good. Most of these data scientists didn't know how to use Git. They would run the cells out of order, which means it wouldn't be reproducible. There are all sorts of things like that. The thing is, I agree with you that you you can use them in this in this workflow. But it comes back to what I was saying before about, you know, we we were talking about the call center and it being like a low intelligence job. You know, the data scientists, the reason why they they are doing intelligent work is they are creating something that doesn't exist. They are figuring out the contours of a problem. They're actually working in a domain that is poorly understood. But you could argue now the bull case is when the data scientists can succinctly describe the contours of the problem, maybe we could go to Claude code, and we could implement it properly. But how do we bridge between those 2 worlds?

Jeremy Howard 01:12:54

I think that would be a terrible, terrible idea. Like, you don't wanna remove people from their exploratory environment, you know? Research and science is developed by people building insight, you know? Whoever you listen to, you know, whether it be Feynman or whatever, like, you always hear from the great scientists how they build deeper intuition by by building mental models, which they get over time, by interacting with the things that they're learning about. And like in Feynman's case, because it was theoretical physics, he couldn't actually pick up spinning quark, but he did literally study spinning plates, you know. You gotta find ways to to deeply interact with with what you're working with. Like, so so many times I've seen data science teams because you're right. Data science teams aren't very familiar with Git and aren't very familiar with things that they do need to understand. And so often, I've seen a software engineer will become their manager, and their fix to this will be to tell them all to stop using Jupyter notebooks. And now they have to use all these reproducible blah blah virtual, you know, virtual end blah blah. They destroy these teams over and over again. I've seen this keep happening. Because the solution is not create more discipline and bureaucracy. It's solve the actual problem. So for example, we we built a thing called an n b merge driver, which so a lot of people don't realize this, but actually notebooks are extremely git friendly. It's just that git doesn't ship with a merge driver for them. So git only ships with a merge driver for line based text files. But it's fully pluggable. And so you can easily plug in 1 for JSON files instead. And so we wrote 1. So now when you diff, you know, when you get a git diff with our merge driver, you see cell level diffs. If you get a merge conflict, get search level cell level merge conflicts. The notebook is always openable in Jupyter. NBDime did the same thing, so 2 independent implementations of this. So yeah, there were problems to solve, you know, but the solution to it was not throw away Brett Victor's ideas and make people further away from from their exploratory tools. But to fix the exploratory tools, and I think all software developers should be using exploratory based programming to deepen their understanding of what they're working with. So that they end up with a really strong mental model of the system that they're building and they're working with. And then they can come up with better solutions, more incrementally, better tested. I basically never have to use a debugger, because I basically never have bugs. And it's not because I'm a particularly good programmer, it's because I build things up small little steps, and each step works and I can see it working and I can interact with it. So there's no room for bugs, you know?

Dr. Tim Scarfe 01:16:20

You know, I'm so torn on this because I agree with you. And I'm also skeptical of people who say that organizations, they they they converge onto ways of doing things, they no longer need to evolve. They no longer need to adapt. Innovation is adaptivity, right? And we should increase the surface area of adaptivity as much as we possibly can. So we need people that are constantly testing new ideas, finding these constraints. But by the same token, we need to use the cloud, we need to use CICD. We need to get this stuff into production.

Jeremy Howard 01:16:50

Yeah. So do you do you do but, like, there's absolutely no like, so NB dev ships with out of the box CI integration, and that's like the tests are literally there, like, because the source is a notebook, the entire exploration of like, how does this API work, you know, what does it look like when you call it, The implementation of the functions, the examples of them, the documentation of them, the tests of them are in 1 place. So it's much easier to be a good software engineer in this environment. So, yeah, like, do do both, you know.[26]

[26] Jeremy Howard: Response to AI Risk Letter — Blog Post Jeremy's response to AI existential risk discussions referenced in the AI safety section.

AI Existential Risk & Power Centralization ●

Jeremy Howard 01:16:50

Yeah. So do you do you do but, like, there's absolutely no like, so NB dev ships with out of the box CI integration, and that's like the tests are literally there, like, because the source is a notebook, the entire exploration of like, how does this API work, you know, what does it look like when you call it, The implementation of the functions, the examples of them, the documentation of them, the tests of them are in 1 place. So it's much easier to be a good software engineer in this environment. So, yeah, like, do do both, you know.[27]

Dr. Tim Scarfe 01:17:33

So do you remember there was there was that existential risk should be an urgent priority, and it was signed by folks like Hinton and and Demis Hassabis. And you responded, basically with a rebuttal, and that was with Arvind, you know, the the snake oil guy. Tell me about that. Do you think we should be worried about AI existential risk?

Jeremy Howard 01:17:52

I mean, that was a certain time, wasn't it? And I feel like things have changed a bit. Thank God. I feel like we we, not just being Aravind, but broadly speaking, the community of which we're a part, kind of probably won that. Now we have other problems to worry about. But you know, basically, at that point, the prevailing narrative was AI is about to become autonomous. It could become autonomous at any moment, and could destroy the world. So it very much comes from, you know, Alizia Yukowski's Yep. Work, which I think clearly has been shown to be wrong at many levels at this point.

Dr. Tim Scarfe 01:18:37

That they would refute that, obviously.

Jeremy Howard 01:18:39

Of course, they would. Yeah. It's 1 of those things that they can always refute just like any doomsday cult unless you give it a date and the date passes.

[27] Jeremy Howard: Response to AI Risk Letter — Blog Post Jeremy's response to AI existential risk discussions referenced in the AI safety section.

Dr. Tim Scarfe 01:18:48

Well, I've I've updated a little bit in the sense that I I now think I I would now say that these models can be said to be intelligent in restricted domains. The ARC challenge showed that. So if you place constraints into the problem, you you can you can go faster towards a known goal. Even agency, you you can put a planner on that and you can go if you if you know where you're going, you can go there faster. But that doesn't help you. Like, you can have all the intelligence and agency in the world. But if you don't have the knowledge and the constraints, then you're going in the wrong direction faster. And I think they don't seem to appreciate that these models don't actually know the world.

Jeremy Howard 01:19:22

Like, none of that was even relevant to Ivan and my point, which was and is, that it's misunderstanding where the actual danger is. Yep. Which is that when you have a dramatically more powerful technology entering the world, that can make some people dramatically more powerful, people who are in love with power will seek to monopolize that technology. And the more powerful it is, the more strong that urge from those power hungry people will be. So to ignore people so here's the problem. If you're like, I don't care about any of that. All I care about is autonomous AI taking off, you know, singularity paper clip, nano goo, whatever. The obvious solution to that is, oh, let's centralize power. And this is which is what we kept seeing, particularly at that time. Let's give either very rich technology companies, or the government, or both, all of this power, and make sure nobody else has it. In in my threat model, that's the worst possible thing you can do, because you've centralized the ability to control in 1 place. And therefore, these people who are desperate for power just have to take over that thing.

Dr. Tim Scarfe 01:20:49

Could could we distinguish, though, what you mean by power? Because we've we've just spent some of this conversation talking about how it's not actually as powerful as people think it is.

Jeremy Howard 01:20:57

But I'm I'm not even that's what but mine is an even if thing. Right? So, like, I I I just say, even if it turns out to be incredibly powerful. Right, like, don't I don't even wanna argue about whether it's gonna be powerful because that's speculative. Even if it's gonna be incredibly powerful, you still shouldn't centralize all of that power in the hands of 1 company or the government.

Dr. Tim Scarfe 01:21:24

Yeah.

Jeremy Howard 01:21:25

Because if you do, all of that power is going to be monopolized by power hungry people, and used to destroy civilization, basically. You'll end up with a case where all of that wealth and power will be centralized with the kinds of people who who want it centralized. So, like, society for hundreds of years have faced this again and again and again, you know. So when it's like, you know, writing used to be something that only the most exclusive people had access to knowing about writing. And the same arguments were made. If you let everybody write, they're gonna use it to write things that we don't want them to write, and it's gonna be really bad, you know. Ditto with printing, ditto with the vote, like like and again and again, society has to fight against this natural predilection of the people that have the status quo power to be like, no, this is a threat. So when we're saying like, okay, what if AI turned out to be incredibly powerful? Would it be better for society to be that to be kept in the hands of a few or spread out across society? My argument was the latter. Now, there's also an argument which is like, hey, don't worry about it. It's not gonna be that powerful anyway. I I just didn't wanna go there, because it's not an argument that's easy to win, because you can't really say what's gonna happen. We're all just guessing. But I can very clearly say, like, well, if it happens, would it be a really good idea to only let Elon Musk have it? Or would it be a good idea to only let Donald Trump have it?

Dr. Tim Scarfe 01:23:29

Dan Hendricks spoke about this offense defense asymmetry. Uh-huh.

Jeremy Howard 01:23:33

So

Dr. Tim Scarfe 01:23:33

it's actually very important for us to have to have countervailing, you know, defenses. But let let's just take that as a given for a minute. Because obviously, when we look at something like Meta and Facebook, it's it's quite clear what the power imbalance is, you know, they they they control all of our data. They they know what we're doing. With with something like OpenAI and Claude, so it's not as good as we thought it was because actually humans still need to be involved. But for example, they have all of our data, right? And you might be working on some new innovative technology and you're using Claude and you're sending all of your information up there and they can now copy you. Mean, what what kind of risks are you talking about to be more concrete?

Jeremy Howard 01:24:09

Yeah. No. I mean, so I was not talking about any of those things, right? So at the time, I was talking about this speculative question of what if AI gets incredibly powerful.

Dr. Tim Scarfe 01:24:17

Well, I mean, like like now, for example, they they say that this is the new means of production. And that's that seems completely hyperbolic to me. But, like, in your best estimation now, if there are risks, what are they?

Current Risks, Privacy & Enfeeblement •

Dr. Tim Scarfe 01:24:17

Well, I mean, like like now, for example, they they say that this is the new means of production. And that's that seems completely hyperbolic to me. But, like, in your best estimation now, if there are risks, what are they?

Jeremy Howard 01:24:29

If there are risks with the current state of technology, I mean, I think some of them are the ones we've discussed, which is people enfeebling themselves by basically losing their ability to be to become more competent over time. That's that's that's the big risk I worry about the most. The privacy risk, it's there, but I'm not sure it's much more there than it was for Google and Microsoft before. Like, you know, you used to work at Microsoft. You know how much data they have about the average Outlook, Office, etcetera, user. Ditto for Google, you know, the average Google Workspace or Gmail user. Those privacy issues are real. Although I think there are bigger privacy issues around these companies which the government can outsource data collection to. So back in the day, it used to be companies like ChoicePoint and Acxiom. Nowadays, it's probably more companies like Palantir. The US government is actually prohibited from building large databases about US citizens, for example. But it's not prohibited companies are not prohibited from doing so, and the government's not prohibited from contracting things to those companies. So, I mean, that's a huge worry, but I don't think it's 1 that AI is uniquely creating. It certainly so, like, you're in The UK. As you know, in The UK, surveillance has been universal for quite a while now. It certainly makes it easier to use that surveillance. But a sufficiently well resourced organization could just throw 1000 bodies at the problem. So, yeah. I'm not sure these are due privacy problems. There's maybe more common ones than they used to be.

Dr. Tim Scarfe 01:26:28

Yeah. Jeremy, I've just noticed the time. I need to get to the airport.

Jeremy Howard 01:26:31

Alright.

Dr. Tim Scarfe 01:26:32

This has been amazing.

Jeremy Howard 01:26:33

Thank

Dr. Tim Scarfe 01:26:33

you,

Jeremy Howard 01:26:33

sir. Thank you

Dr. Tim Scarfe 01:26:34

for

Jeremy Howard 01:26:34

coming.

Dr. Tim Scarfe 01:26:34

Yeah.

Jeremy Howard 01:26:35

Hope you had a nice trip.

Dr. Tim Scarfe 01:26:36

Thank you so much.

References

[1] **fast.ai Blog: Self-Supervised Learning**
`https://www.fast.ai/posts/2020-01-13-self_supervised.html`
Blog Post Referenced during the ULMFiT and fine-tuning discussion, covering self-supervised learning foundations.

[2] **M. Chirimuuta: The Brain Abstracted**
`https://mitpress.mit.edu/9780262548045/the-brain-abstracted/`
Book Book on abstraction and cognition referenced during the learning mechanics discussion.

[3] **DeepMind Blog: Gemini Deep Think**
`https://deepmind.google/blog/accelerating-mathematical-and-scientific-discovery-w`
Blog Post Referenced during the AI reasoning and abstraction discussion.

[4] **MLST Archive: Why Creativity Cannot Be Interpolated**
`https://archive.mlst.ai/read/why-creativity-cannot-be-interpolated`
Archive Article MLST archive piece on creativity and interpolation referenced in the abstraction discussion.

[5] **Mathilde Caron et al.: Emerging Properties in Self-Supervised Vision Transformers (DINO)**
`https://arxiv.org/abs/2104.14294`
Research Paper Self-supervised Vision Transformer paper referenced at 13:54. Authored by Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin at Facebook AI Research (not Yann LeCun, as stated in the episode).

[6] **Modular Blog: Claude C Compiler analysis**
`https://www.modular.com/blog/the-claude-c-compiler-what-it-reveals-about-the-futu`
Blog Post Analysis of Claude Code's C compiler implementation discussed during the Claude Code section.

[7] **Anthropic Engineering Blog: Building C Compiler**
`https://www.anthropic.com/engineering/building-c-compiler`
Blog Post Anthropic's engineering blog post about Claude Code's C compiler implementation.

[8] **METR Study: AI OS Development**
`https://metr.org/blog/2025-07-10-early-2025-ai-experienced-os-dev-study/`
Research Study Study on AI software engineering capabilities referenced during the coding vs. engineering discussion.

[9] **Fred Brooks: No Silver Bullet**
`https://www.cs.unc.edu/techreports/86-020.pdf`

Paper Classic software engineering essay on essential vs. accidental complexity.

[10] **Oxford VGG: Sculptor Identification Paper**
`https://www.robots.ox.ac.uk/~vgg/publications/2012/Arandjelovic12a/arandjelovic12`
`pdf`
Research Paper Computer vision paper on sculptor identification, if referenced in the discussion.

[11] **Daniel Dennett: Consciousness Explained**
`https://www.amazon.com/Consciousness-Explained-Daniel-C-Dennett/dp/0316180661`
Book Dennett's book referenced in the consciousness and intelligence cosplaying discussion.

[12] **John Searle: Minds, Brains, and Programs**
`https://www.cambridge.org/core/journals/behavioral-and-brain-sciences/article/minds-brains-and-programs/DC644B47A4299C637C89772FACC2706A`
Paper The Chinese Room argument paper referenced in the AI consciousness debate.

[13] **Anthropic Paper: AI Skill Formation**
`https://arxiv.org/pdf/2601.20245`
Research Paper Paper on AI learning and skill formation referenced during the automation discussion.

[14] **Ebbinghaus: Memory / Spaced Repetition**
`https://www.loc.gov/item/e11000616/`
Historical Reference Ebbinghaus' work on memory and spaced repetition referenced in the desirable difficulty discussion.

[15] **Cesar Hidalgo: Infinite Alphabet / Laws of Knowledge**
`https://www.amazon.com/Infinite-Alphabet-Laws-Knowledge/dp/0241655676`
Book Book on organizational knowledge referenced during the knowledge embodiment discussion.

[16] **John Ousterhout: Slope vs Intercept**
`https://gist.github.com/gtallen1187/e83ed02eac6cc8d7e185`
Technical Note Referenced for the "slope makes up for y-intercept" career growth concept.

[17] **Cursor Blog: Scaling Agents**
`https://cursor.com/blog/scaling-agents`
Blog Post Referenced during the AI coding tools and vibe coding discussion.

[18] **Bret Victor: Inventing on Principle**
`https://vimeo.com/906418692`
Video Referenced during the interactive programming and REPL environments discussion.

[19] **Joel Grus: I Don't Like Notebooks**
`https://www.youtube.com/watch?v=7jiPeIFXb6U`
Video The famous JupyterCon talk critiquing notebooks referenced in the notebook debate.

[20] **fast.ai Blog: NB Dev Merged Driver**
`https://www.fast.ai/posts/2022-08-25-jupyter-git.html`
Blog Post Referenced during the notebooks and nbdev discussion.

[21] **Jeremy Howard: Response to AI Risk Letter**
`https://www.normaltech.ai/p/is-avoiding-extinction-from-ai-really`
Blog Post Jeremy's response to AI existential risk discussions referenced in the AI safety section.