/mlst

NEURAL NETWORKS ARE WEIRD! - Neel Nanda (DeepMind)

March 23, 2026



Machine Learning Street Talk

Podcast Transcript

Revision: 0f792928

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

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

Contents

Overview 2

Introduction and Core Thesis 3

Outdoor Interview: Mechanistic Interpretability Foundations 5

Sponsor Messages 17

Deep Dive: Mechanistic Interpretability 18

Biological Evolution Parallels 24

Universal Circuits and Induction Heads 27

Entity Detection and Knowledge Boundaries 28

Activation Patching as Causal Intervention 30

Golden Gate Claude and Feature Amplification 34

Model Personas and RLHF Effects 38

Steering Vectors and Linear Representations 39

Hallucinations and Model Uncertainty 40

Sparse Autoencoder Architecture 42

SAE Core Challenges: Dead Features and Training 49

Advanced Activation Functions and TopK 55

Research Applications in Transformer Circuits 57

Feature Learning and Width Parameters 59

Scaling Laws and Training Stability 64

Feature Identification and Bias Correction 68

Training Dynamics Analysis 70

Scale and Infrastructure Requirements 72

Chain-of-Thought and Latent Reasoning 75

Career Path and Advice for Researchers 77

References 84

Overview

Neel Nanda leads the mechanistic interpretability team at Google DeepMind. At 26, he's become one of the most prominent researchers working on the question of what's actually going on inside neural networks – systems that can win IMO medals and write complex software, but which nobody actually designed or understands.

This nearly four-hour conversation is a deep technical dive into the field. Nanda explains why machine learning is fundamentally weird: we produce artifacts that do impressive things, but unlike conventional software, no one wrote the code or planned the architecture. His team's goal is reverse-engineering these systems by finding the internal structures and algorithms that emerge during training.

The discussion covers the mechanics of sparse autoencoders at length – how they decompose model activations into interpretable feature vectors, the mathematical foundations (ReLU vs TopK activation functions), scaling laws for feature learning, and the engineering challenges of running them at the scale of frontier models. Nanda walks through the Golden Gate Claude experiment (amplifying a single feature to make Claude obsessed with the Golden Gate Bridge), induction heads (the circuits responsible for in-context learning), and activation patching as a causal intervention technique.

On AI safety, Nanda is pragmatic. He argues that mechanistic interpretability gives us genuine empirical evidence about questions that are otherwise stuck in philosophical debate – do models have goals? Do they deceive? He also discusses the limitations: sparse autoencoders haven't yet demonstrated capabilities beyond what fine-tuning already achieves, and at sufficient model complexity, models could potentially facade interpretability measurements. The conversation covers his path from pure maths at Cambridge through Anthropic to DeepMind, and why he thinks hands-on coding matters more than reading papers for new researchers entering the field.

Introduction and Core Thesis

Neel Nanda 00:00:00

You could you could just hit I could just hear the, like, slight, creepy sounds in the background of footsteps. 3 things you're gonna learn. Everything about sparse autoencoders, what is the right way to do causal interventions on models and how to think about it, why mechan mechanistic interpretability can help us make models safer. Machine learning is in some sense a really weird field because we produce these artifacts, neural networks, but no 1 designs them. They're not like a computer program that someone made a design and then wrote the code. Instead, we just create some flexible architecture, shove lots of data in, and an effective system comes out, but we have no idea what's going on inside or how it works. And these are really impressive systems. They can do lots of things, lots of complex software engineering tasks and complex of reasoning and get IMO silver medals. We just don't understand how they work and what's happening inside. We have computer programs, essentially, that can do things that no human programmer knows how to write, and my job is to try to fix this. Is AGI an existential risk is a pretty polarizing question with lots of prestigious people and strong opinions on both sides, but, like, frustratingly little empirical evidence. And I think 1 of the things that interpretability could potentially give us is, like, a clearer sense of what's going on inside these systems. Like, do they do things like you that we would call planning? Do they have any meaningful notion of goals? Will they do things like deceive us? And the more we can understand how these things manifest and, like, whether they occur in situations where they shouldn't, they're like, more I think we can learn. And this seems like a really important research direction[1]

Tim Scarfe 00:01:45

to me. The robustness to its representation post steering kind of indicates to me that it's more than just a keyword matching. It actually understands what the thing is.

Neel Nanda 00:01:54

It seems pretty clear to me that it's not just keyword matching because we observe things like multilingual features where the same text in different languages lights the feature up. You tend to see these beyond a certain model scale. In Scalimonosvantesti, they had multimodal features like the Golden Gate Bridge 1 lit up on pictures of the Golden Gate Bridge. And, like, with my interpretabil-

[1] Neel Nanda - Personal Website — person

ity handle, I don't find this very surprising because if you can map your inputs into a shared semantic abstract space, you can do efficient processing. So, like, of course, this will happen. And there's some interesting work. Like, there was the do llamas think in English paper from, I think, Chris Wendler. I think that seemed to show that the model decides what to say, and it decides what language to say it in at kind of, like, different points, and you can causally intervene on them differently. But I think if you have the kind of engram matching stochastic parrot perspective, you would not predict this. And I don't really understand the people who hold that perspective nowadays, to be honest. I think it's clearly falsified. So I think to understand sparse autoencoders, we need to first begin with what is the problem they're trying to solve? So you can think of a neural network as being made up a bunch of layers, and you pass in some input. It gets converted to vectors or a series of vectors in the case of a transformer, and then each layer transforms this into a new vector or series of vectors. These are the kind of activations in the middle. Often in the layers, we'll kind of in the middle of them have intermediate activations. We believe that these activation vectors often represent concepts, features, properties of the input, something interpretable, but it's a vector. We need some way to convert it into a thing that is meaningful. And sparse autoencoders are a technique that tries to do that. They basically decompose the vector into a sparse linear combination of some big list, sometimes called a dictionary of meaningful feature vectors. We hope that these feature vectors correspond to interpretable concepts, and it's sparse in the sense of most vectors are not part of this combination on any given input. I think that if you go on something like Neuronpedia, they'll often show you kind of different intervals, like what is between the fiftieth and seventieth percentile of activations. Let's, like, give you some stuff. And this is useful for getting a kind of broader view. Also, just looking at the causal effect is interesting, like feature that lit up on, like, fictional characters in general, but especially Harry Potter. But when I steered with it, it was kind of only Harry Potter related.[2]

Tim Scarfe 00:04:40

You can join Minds AI, which is part of Tufa AI Labs, to do cool arc research and eventually beat arc. After that, they're gonna work on LLM based reasoning following in the footsteps of o 1. All pure research, no product, well funded. Go to twoferlabs.ai, and we'll be covering some of their research talks in Zurich 2 in the coming months. We're also sponsored by Sentinel, the insanely optimized model serving platform. It's best in class compared to others, 400% faster tokens per second than the hyperscalers. It's over 70% cheaper, just 8¢ per million tokens on Lama 8,000,000,000 on your own private endpoint. You get 10 free credits to evaluate the platform. So what are you waiting for? Sign up at sensml.ai.

[2] Sparse Autoencoders Find Highly Interpretable Features in Language Models — paper

This is quite a long podcast. The first part of it was an outside interview that I recorded with Neil, and then the the rest of it is the inside component where we talked a lot about mechanistic interpretability and sparse autoencoders. Enjoy.

Neel Nanda 00:05:43

Machine learning is in some sense a really weird field because we produce these artifacts, neural networks, but no 1 designs them. They're not like a computer program that someone made a design and then wrote the code. Instead, we just create some flexible architecture, shove lots of data in, and an effective system comes out. But we have no idea what's going on inside or how it works. And these are really impressive systems. They can do lots of things, lots of complex software engineering tasks and complex of reasoning and get IMO silver medals. And I'm like, we just don't understand how they work and what's happening inside. We have computer programs, essentially, that can do things that no human programmer knows how to write. And my job is to try to fix this. So I run the Google DeepMind mechanistic interpretability team, and mechanistic interpretability[3]

Outdoor Interview: Mechanistic Interpretability Foundations ●

Neel Nanda 00:06:45

is a type of AI interpretability that says, I believe that neural networks, in general, learn human comprehensible structure and algorithms inside. And that with enough care and attention and rigor, we can go through them and at least start to uncover parts of this hidden structure and go beyond just treating it as, like, a weird black box system or a thing we take gradients through to line up parts of the input, but, like, actually understand how the algorithm is represented in the weights and activations. And it's really hard, but we made some progress, and I'll talk about that

Tim Scarfe 00:07:26

Yeah. What what is reasoning to you, Neil?

[3]Progress Measures for Grokking via Mechanistic Interpretability — paper

Neel Nanda 00:07:29

Do you mean what is it to know when an LLM is reasoning? Or just what is reasoning in the philosophical abstract?

Tim Scarfe 00:07:36

Philosophical abstract.

Neel Nanda 00:07:37

Oh, god. Philosophical abstract. I guess I'm pretty sympathetic to the applying some kind of logical rules of inference. Like, you have some knowledge, you do things with that knowledge to produce more knowledge. Though, it's kind of unclear whether there needs to be a sense of intentionality and agency behind it. Like, if a squirrel has the learned reflex of noise run up tree, Is this reasoning or not? I don't know. And I think there's also a interesting question with LLMs of is the LLM's forward pass the thing we are analyzing, Or is the LLM as a system including long generations and chain of thoughts, or ridiculously long chains of thought in the case of o 1? Is that a thing that we can consider to be reasoning? Because, like, that can clearly reason. Like, it produces a logical stream of inferences, and you can give it arbitrary things. Like, there's that ProntoQA benchmark that's got all tempuses are limpuses. All limpuses are blue. What are tempuses? And models can do that with chain of thought. While it seems a much higher bar to say that it can do that kind of thing when the Ford pass, And honestly, that seems more like a claim about the internal cognition. And I don't know. That's why we got the field of mechanistic interpretability, but we haven't got that yet. In terms of inference time compute versus, having kind of phase transitions or emergent things during training, it seems hard to compare to me. Like, when you train when you spend more compute training a model, it will have more capabilities. Some might be learned in a sudden way. Some might be learned gradually. I don't really care. I don't think that's very relevant here. If you want to train a model with less compute but spend lots of it on inference time compute, that will get you a different trade off. I basically just buy that for many use cases, the ratio of inference time to training time compute is like has been quite off historically, and that in future it's going to be more balanced. I think that we don't yet know what these new inference time compute focused systems are going to look like economically. Like, we have scaling laws of loss curves, but how does this cash out in economically useful things? So, for example, 1 thing I found quite interesting in the o 1 system card is it was only as good as Claw 3.5 SONNET at agentic tasks. I think that inference time compute systems are likely to be most useful for that kind of thing because you're happy waiting a while while it just goes off and does something. You don't need to interact with

it. And I predict we just haven't learned how to use them, and I predict that we haven't learned the right scaffolding to make them good at this. And this just gives me very wide error bars on what the economics are gonna shake out as.

Tim Scarfe 00:10:57

You know, having having a good Twitter community, I think, is paramount to getting intelligent commentary. So I have

Neel Nanda 00:11:06

occasionally seen intelligent comments on Twitter.

Tim Scarfe 00:11:08

Here's a good example. So Joe Smith said, ask him literally anything about why AI safety is important so he can dunk on you ding dongs.

Neel Nanda 00:11:18

Alright. Why is AI safety important? So, I think dunking is distasteful, but also I don't know. I think that human level AI is just a thing that is technologically feasible that will happen sooner or later. And a bunch of people are trying to build it. And this just seems like an incredibly big deal that will massively change the world. And if you have these intelligent autonomous agents, there are ways this can go well, and there are ways this can go badly. And it's just really important that we spend a ton of research energy on things like safety and interpretability so we can make this go right.

Tim Scarfe 00:11:58

Daniel Phelan, he said, how much progress have we made in MEK Interp over the last 5 years, and are we on track, if it were? I feel

Neel Nanda 00:12:10

reasonably good about where the field is at. I think that we're trying to solve a really hard problem, and we might fail. Though I also think that progress in Mekinterp often looks like uncovering more and more of the structure in a system such that even if we don't get to the kind of really ambitious goals in the field, which I think are often not very realistic, I think we can still get to the point where we are able to do interesting, useful things with systems. And is that on track? Is that enough? Kind of unclear. I feel like we've made quite a lot of progress. I feel like there's a lot of things I now understand much better, like superposition and what to do about it, how to do principled causal interventions to try to 1 another circuits, how to engage with the transformers, how to like, what kinds of structure to expect in these models. But also, there's a ton we don't know, so unclear.

Tim Scarfe 00:13:09

Is there a numeric benchmark that has a a measurable score in MEC Interp? You know, something like the the blur score in in NLP. Do you think it would be useful? And how close are we to to getting that?

Neel Nanda 00:13:23

I think the problem here is that summary statistics can always lie to you. You're taking a rich, complex, confusing, high dimensional object, and you're trying to compress it down to, like, a number or, like, a bunch of numbers. And if you have a kind of well defined technique where you know what it's supposed to be doing and you're just trying to measure something like, with the sparse autoencoder, how good is it at reconstructing the model at a given level of sparsity? That's just pretty measurable. Or if you're like, how good is it at helping me unlearn some facts? That's also pretty measurable. And these are just things that I would like sparse autoencoders to be good at. But there's lots of things like, does this model exhibit superposition? Have we truly understood this circuit? Or could there be, I don't know, a positive and negative bit canceling each other out that we hadn't noticed, but on some other input that comes apart and is important? And it's just really hard to come up with something that properly captures these. Though I think as the field matures and becomes more paradigmatic, this is a direction I wanna see.

Tim Scarfe 00:14:31

So Seb Krier, he said, how promising do you think control vectors, circuit breakers, and REPNG? REPNG? Do you know what that is?

Neel Nanda 00:14:42

So that stands for representation engineering.

Tim Scarfe 00:14:44

Oh, sorry.[4]

Neel Nanda 00:14:45

It is super yeah. Which is, like, similar to some other techniques like activation steering, control vectors. And yeah. So the key idea here is basically you define a vector in some way. Like, you give the model the prompt I love you and the prompt I hate you, and you take the difference in the residual streams. And now you've got a kind of love vector. And then you add this in and some change happens. Like, it outputs happy text. And, Circuit Breakers, I believe is, I mean, I can't really remember the details of the paper, but it's basically a way to make models, if they see something harmful, output, or, like, kind of not output the harmful thing using control vectors and some kind of training. But I don't really remember the details. So sorry. I think so I'll mostly focus on activation steering. I think it's an interesting direction. Seems like an interesting technique. I like the fact that this kind of fairly interpretable model and tunnelsy technique can do cool things. I mean, I think Golden Gate Claude was, like, very similar to this technique. It happened to use an SAE, but this probably wasn't necessary. And Golden Gate Claude was, like, hilarious. And there also seem to be ways you can make this more useful by, like, reducing hallucination rates or increasing truthfulness. Is it gonna be enough to align help us align a, like, human level system? Highly unclear. It might help.[5]

Tim Scarfe 00:16:23

Stylianos says, and this will echo Waleed Sabah. Now, Waleed Sabah is a friend of the show. He is a GoFi person. Just just to just to put that in there. And he said, assuming that explaining a decision means doing an inference in reverse, so from the decision, backtrack to find the steps that led to it. And given that neural networks are incapable of doing that because there's no invertible compositionality, how can we truly achieve explainability?

[4] Representation Engineering / Activation Steering — paper
[5] Golden Gate Claude — website

Neel Nanda 00:16:56

So that just isn't really how I think about the problem. Like, the way I think so first off, you don't need to care about the invertibility thing because models are a stack of layers. Each layer is a much simpler function, and you can just analyze the activations after each layer. So it's a bunch of vectors. So, like, it's not it's not like I just have the output and I have the input and I need to somehow reverse engineer this black box. We have the weights. We have the activations. We know every mathematical detail of what is going on in this system. We just have no idea what a we just, by default, have no idea what any of it means. And, that is the problem that we've been making progress on. And, you know, I think if you have, like, an activation, you've got another activation, and you're like, how did you go from a to b? You can make some inferences. You can look at the weights. You can do causal interventions. Like, my I'm not holding myself to the standard of proof of, like, we have inverted a mathematical function. I agree that neural networks are not an invertible function, but I am holding myself to the standard of, like, I have good evidence that my story is faithful to what's going at least part of what's going on inside the model. And there's a bunch of ways you can try to to find that, which I'm sure we'll chat about.

Tim Scarfe 00:18:25

I must admit, I'm genuinely confused about this. So I don't think well, I mean, may maybe our brains are compositional, but it it feels like our brains maybe aren't that different to neural networks. I mean, of course, they they are different. Right? Because the neurons can talk to each other almost independently, and you can get these repeating cycles and so on, whereas neural networks are are left to right. But if we just leave that aside, they are similar in the sense that they're very sub symbolic. They're very diffused. They're very complex. They're very entangled. Yet, when you look at our mind and when you do psychology, it it appears as if we can do compositional reasoning. So so isn't it isn't this this weird 2 ways of looking at things? You can look at a neural network and you think, oh, It's it's blown up and it's entangled and it's complex and different bits of the neural network, you know, apparently do the same type of thing. Could could it ever be compositional in in some sense?

Neel Nanda 00:19:19

It feels more confusing to me when we're discussing a single forward pass. But, like, if you give it a scratch pad to do chain of thought on, it can just obviously do all of this. It just writes it down and then does inferences on those. And, like, is this different from me thinking verbally through a problem? I'm not convinced it's that different.

Tim Scarfe 00:19:37

Why do you think that self prompting chain of thought rationales gives gives an uplift?

Neel Nanda 00:19:43

To me, it just seems really intuitive that you get an improvement in capabilities when you have chain of thought. Like, when I have paper to write down my thoughts, I am smarter. I can do harder problems. In terms of computational complexity, like, the model, rather than going through every layer once, can transmit some information back, and it can also, kind of just parallelize things more. So it can have 1 token where it's doing the what should I, what should my plan be calculation. And another 1 where it's like, what does step 1 of the plan tell me to do? And another 1 where it's like executing what step 1 tells you to do. And if you're trying to do these all in the same token, the model just it gets very crowded. There's, like, lots of concepts that interfere with each other in the residual stream, and this makes it more error prone for the model. And, also, there is just some inherently sequential nature to a lot of this computation.

Tim Scarfe 00:20:44

The rationale might be noisy. It it might be wondering what the average number of carrots are eaten every day in Belarus. And problems that require backtracking as well, the previous trajectories might become noisy and might become distractors. So it's it's not a definite win win, is it? I expect there are situations where it will make the model worse. For example, there was this delightful paper from Miles Turpin on unfaithful chain of thought where you give the model a bunch of, like, multiple choice a or

Neel Nanda 00:21:16

b questions, with a chain of thought response and then an answer. You give them, like, 10 of these as a few shot prompts and then give them another question. And, what they did is they made in the few shot prompts, the correct answer was a for all of the questions, and then in the final 1, the correct answer was b. And they found that if they up prompted it to give a chain of thought at the final thing, then it gave a bullshit chain of thought for why the correct answer was a and then said a. While if they didn't ask it for a chain of thought, it gave b. Well, it was more likely to give b. And, in some ways, this is kind of a boring example because telling it to deuterium thought probably triggered the fuchsia ness more. But it also I don't know. I find it wild that the model has an internal representation of the answer is a, I should generate a spurious explanation based on this fact. Like, what? I would love someone to do a chain of thought, to do a Mech and Terp project, what the hell is going on there?

Tim Scarfe 00:22:17

But from a model theory of mind perspective, does that in any way denigrate the idea that they have goals and intentionality? If if if it's such a complex token by token space, I mean, maybe maybe we don't have goals and and intentionality, but does that in any way denigrate the kind of the theory of mind of of LLNs?

Neel Nanda 00:22:38

I expect a goal mechanistically to basically look like you have some criteria, you consider actions, and you judge actions according to this criteria. I expect this to be easier with an internal scratch pad, but my general intuition is that if a bad model can do something with an internal scratch pad, a good model can do it without the internal scratch pad and can do even better stuff with a scratch pad. And, like, just algorithmically, it doesn't seem like evaluating actions should be that hard. I think this is this seems like quite an interesting thing to study in, like, more toy RL settings, where there seems like decent evidence that things engage in planning. Like, Eric Jenner had this delightful paper looking at Leela 0 that I think found evidence it was, like, thinking at least 2 moves ahead.[6]

Tim Scarfe 00:23:32

Do you have any advice for fresh PhDs in Mechinterp?

[6] Evidence of Learned Look-Ahead in Chess-Playing Neural Networks — paper

Neel Nanda 00:23:37

Don't spend too much time reading papers. I think a standard academic approach to getting into a new field is I will read all of the literature. And you should read some of the literature. I have a reading list we can probably link in the description. But, I think you should also just spend a lot of time coding. Like, play with small models. Play with past autoencoders. Play with whatever you're curious about. There's pretty good tooling and tutorials nowadays, and we'll hopefully have some of those in the description. And, it's Mechanism helps us a very empirical field. You wanna build intuition, and you can often get something like a cool blog post out of not that much work and playing around. Yeah. I generally yeah. Get mentorship. Like, ideally, your supervisor spends a bunch of time with you. Find postdocs in the lab, try to find ways to spend time with them, maybe focus on being useful to them, like helping with their projects. I'm a big fan of pair programming as a way of learning technical skills. If you can find anyone who's better at ML coding than you or interpreting than you and can sense that I'm pairing with them, maybe, like, as part of helping them with some project or something, I think that can be a great way to learn. And I think lots of people in academia somewhat neglect technical skills. Though, some are excellent, so no shade there. And just the more the better you are at coding, faster you are at experiments, and the more research you will do. Be skeptical. It's really easy to come up with a beautiful, pretty idea, and the idea is actually complete bullshit. You get attached to it so you don't notice. An adviser or even just peers who can red team your ideas are really helpful. You should also just spend a lot of time doing that yourself. Like, how could this be false? What are the ways this could not be true? What are experiments that would distinguish true from false?

Tim Scarfe 00:25:35

Yeah. Yeah. Well, Philip also said, how on earth, and that's how, full stop, on full stop, earth, full stop, did you manage to get to where you are in just 25 years of age? Share your journey of a hero, please.

Neel Nanda 00:25:49

Well, so the relevant life story is I finished a pure maths degree at Cambridge in 2020. I was gonna do a master's, but why on earth would you do a master's during a pandemic? So I spent a year doing internships at various safety labs, like DeepMind and the Future Humanity Institute and the Center of Human Compassible AI at Berkeley. I wasn't super thrilled about any of the specific agendas I'd worked on, but was like, I believe that AI safety is a big deal, and I should work on it. And then I massively lucked out and got a job offer to work with Chris Olah at Anthropic. Spent some time there, ended up leaving for health reasons, and then spent a while independent, and then, ended up at DeepMind. In terms of, like, how this happened, I'm kind of confused, to be honest. I think the key ingredients were getting into a field early that was growing really fast, and I'd like to think I've done some stuff to help that growth, which is also just useful to me. I'm really good at mentoring people, and I just find this really fun. Sorry. I'm really good in the sense of I can have people produce cool papers without spending that much time helping them. And I also just find this a very fun thing to do, which means that I both have my name on a lot of papers, and also I am a better researcher for it. And yeah. No. Luck. And having great mentors like Chris early on, and finding a field that was just like a very natural fit to my taste and talents. I do think it is unfortunately just harder to get into Mechantup now than it was 3 years ago. Sorry. The field has grown. More people are interested. It's terrible.

Tim Scarfe 00:27:49

Neil, 1 day, we will get Chris on the show.

Neel Nanda 00:27:52

I'd be so excited.

Tim Scarfe 00:27:54

Yeah. The man is an absolute living legend.

Neel Nanda 00:27:59

He's so great. Yeah. I think he's only ever done 1 podcast on 80000 hours. Yeah. But it's so good. People should definitely listen to it.[7]

[7] Chris Olah on Interpretability Research (80,000 Hours) — website

Tim Scarfe 00:28:07

Yes.

Neel Nanda 00:28:07

And yeah. And as context for people who have lives, Twitter has been very interested in the fact that I'm 25 recently. It's very confusing.

Tim Scarfe 00:28:19

Oh, what what are you seeing?

Neel Nanda 00:28:21

Let's see. So someone made a meme where they said, today is Neil Nanda's nineteenth birthday. Please comment, happy birthday, Neil, if you believe that reverse engineering neural networks is important for reducing AI risk. And he also wrote this hilarious post, that was completely, bullshit about Neil Mander, the 18 year old prodigy who was reshaping AI inside Google, who invented the World of Warcraft source code at 18 of age. And I'm very confused.

Tim Scarfe 00:28:58

Yeah. The the the Twitter meme sphere is is a very strange place. We have now produced the most comprehensive YouTube video on SAEs. What is an SAE?

Neel Nanda 00:29:09

SAEs, sparse water encoders, are a tool to let you look inside a model as it runs and see what it's thinking about. And I think this is 1 of our most exciting tools to date for interpreting. But, also, most people don't really get how to think about them, and I have a lot of hot takes. And Tim and I spent several hours discussing them. So if that sounds like you're a good time, maybe you should check this out.

Tim Scarfe 00:29:32

Neil, why is Mech Interp relevant to AI safety?

Neel Nanda 00:29:35

Yeah. So, yeah, so I think there's this interesting question of motivation where, in my opinion, Mekinterp is just like a scientifically fascinating field. We just don't get how these systems work internally, but we can make progress, and there is all of these beautiful puzzles to discover. And this is part of why I work in the field. But a large part of why I work in the field is I'm pretty concerned about existential risk from AGI, and I think this is a promising way to make progress on it. And I think at a very high level, I just think that it's much harder to make a system safe or evaluate how safe it is if you don't know what's going on inside. And I think that any progress we can make on deciphering its internals is great on that front. A particularly concerning capability that I think may arise is deception, systems that are capable of deceiving us and, I don't know, trying to evade our evaluations. If it can evade your evaluations, every other evaluation is kind of useless. And it may be quite hard to figure this out by just, like, interacting with a model because if a model is smart enough, it kind of knows that you're doing this. But I think that it's possible that if we can learn to, like, look inside deeply enough, we might be able to detect when this is happening and potentially form a deeper understanding of it and potentially stop this. I think another angle would just be there's a lot of confusions around, is alignment a big deal? Do these risks matter? I think this is quite bad. I think it would be so much better if there was a good empirical grounding here that we could agree on and empirical questions to study. But there's lots of things like, can these systems plan? Do they have goals that there's no real consensus about? Or rather, will this happen? And I think that mechanistic interpretability is potentially a promising tool to answer questions like that because they're fundamentally a question about cognition. I don't think MechInterp is the only tool for answering these questions, but I believe it can be a

Tim Scarfe 00:31:40

promising 1. Is it possible that there would be a level of sophistication in training where the model would deliberately kind of almost encrypt itself to stop you from putting probes in and understanding what it's doing?

Neel Nanda 00:31:54

Like, maybe. My intuition is that that seems significantly harder than learning to lie to people because it just has so many examples of that in the training data. While yeah. It has so many examples of that. And training data, while, how to warp your internals to avoid interpretability tools, it kind of doesn't, and it also doesn't get feedback on that. It's like, I don't really know how to warp my brain so aliens can't, like, read my mind. And, like, arguably, there's, like, patterns of cognition that should be easier or harder to get that it could do, like, not thinking about its goals very often. And, like, maybe that would happen. But I don't know. I predict there will be a window where this stuff is useful before we get models that are, like, so galaxy brained. It's hard to know what tools

Sponsor Messages

Neel Nanda 00:32:52

will work on them.

Tim Scarfe 00:32:52

Yeah. And thank you so much for coming back on.

Neel Nanda 00:32:54

No worries. Happy to be back. My girlfriend was actually very excited when she heard I was coming back on because she says the previous episode was so relaxing to listen to. She used it to help her fall asleep.

Tim Scarfe 00:33:07

I know. I did say last time that Neil's dulcet tones will melt the stress away, and I wasn't lying. But also, there there's there's the other side of that, which is you're an incredible researcher in Mekinterp. And I think a good lead for this is, you know, in your own words, what is Mekinterp?

Neel Nanda 00:33:25

Sure. So, yeah, mechanistic interpretability, or Mekinterb for short, is a subfield of the general study of interpreting AI, non ad books, whatever you wanna call them. And it's kinda based on this hypothesis that neural networks, when trained, have learned some kind of human comprehensible algorithm, But they don't have any incentive to make this legible to humans, but there is some under-

lying structure and algorithm there because, like, that is what does the computation that produces the output. And by careful and principled reverse engineering and science and skepticism, we can decipher parts of those hidden mechanisms. And, ambitiously, this would look like understanding the entire algorithm, but even just understanding what the intermediate variables are or, like, parts of that structure feel like meaningful and exciting progress to me.

Tim Scarfe 00:34:29

I mean, some people argue that I mean, this this is a mammoth task. You know, some might argue that it's, infeasible or or maybe even an unnecessary. I mean, what what would you say to those people?

Neel Nanda 00:34:40

Well, unfeasible and unnecessary are 2 very different things. My response to, infeasible is I think it's reasonable to say fully reverse engineering, like, Gemini or g p d 4 to source code is not very realistic.

Deep Dive: Mechanistic Interpretability

Neel Nanda 00:35:01

But I think that there are a lot of important useful things we can understand about them and partial progress. And, personally, a large part of the reason I'm in this field is I'm pretty concerned about existential risks from AGI, and I think it's important that we have stronger methods of studying them and ensuring safety. And I think that even partial progress in mechanistic interpretability can help us get there, along with just, I don't know, being scientifically fascinating and beautiful. Like, kind of weirds me out. The people are just okay. A bunch of software engineering tasks that seem really complex and difficult, and we have no idea how to make a system that does that in terms of designing it. We don't understand what is inside these things. They are just things that work. And we we've made progress, but I still feel pretty confused on a deeper level about what happens inside them. Regarding unnecessary, what does necessary mean? Like, I think it is important to do research to help us make these systems safer. I think that 1 such pathway is interpretability. I'm not arguing that if we don't do any interpretability research, we will never produce safe systems. That would be unreasonable. I'm not arguing that if we don't pursue specifically mechanistic interpretability but pursue some other philosophies in interpretability, we will never make safe systems. I'm just saying, I think this is a promising approach. I think it's helpful. I think it is helping us find real interesting scientific truths

about these models, and I wanna find them.

Tim Scarfe 00:36:49

So if those are the goals of Mekinter, what are these subgoals?

Neel Nanda 00:36:52

Yeah. So I'd maybe divide the field into 3 conceptual areas. There's basic science, so what on earth is going on inside models? There's automation, taking the techniques and tools for understanding systems and scaling them up, maybe getting an LLM to do them, maybe just having an algorithm that can do them, and, kind of practical applications. Doing things in the real world. Well, the goal isn't necessarily to advance our understanding of interpretability. It's to do something useful and get feedback from reality in the process. And, historically, most of the field has focused on basic science. I think this was correct, and I think we made a lot of progress. I think going forward, I'm excited about exploring other areas more and building on all this progress. Within basic science, a decomposition that I think is often useful is if you think of a model as, like, a computer program in some sense, there's, 2 tasks. What are the variables? And, like, what is their state at each line of the program? And, like, what are the lines of the program? What is the code? What is the algorithm? In Mechinterp jargon, you call the variables features, and their values are, like, how much do they light up? And you call the algorithms or lines of code circuits or algorithms. And, generally, we study a model's activations, the things it computes on the fly when you give it an input, because we believe those represent variables, and, like, what the model is thinking about in some sense, if you'll forgive the anthropomorphism. And, on the other hand, circuits live in the parameters of the model. They're, like, a thing learned during training that doesn't depend on the input, but is like responding to the input and routing it down different channels and things like that.[8]

Tim Scarfe 00:39:05

I mean, you've already answered this question in in your last answer, but just to hit it home a little bit. You know, before we had things like Lyman Schap, we might be doing black box model interpretability, you know, with a surrogate model or something like that. Now you seem to be talking about what algorithms are these models running. I mean, how do you contrast those different views?[9]

[8] Why Should I Trust You? Explaining the Predictions of Any Classifier (LIME) — paper
[9] A Unified Approach to Interpreting Model Predictions (SHAP) — paper

Neel Nanda 00:39:26

Yeah. That's not quite my field. So while I'll say things about it, they could be wrong. I'm sure there's, like, tons of papers that have all kinds of variants of these things. So if I express any critique, I'm sure someone can send me a paper that addresses that or something. I don't know. But on a high level so my understanding of LIME is it's like, let's take a local approximation to a model that's linear or something similar. I think this is a pretty reasonable thing to do, but I think the question is, like, how are you applying the technique? So okay. Let's zoom out a bit. It kinda feels like more of a philosophical difference. Historically, lots of interpretability had just kind of looked at the inputs and the outputs and just hasn't looked at the internals, or they've used it to compute gradients or something. And MechInterp is trying to go beyond this and, like, look at things like the activations of the model, which, again, people have done before, but it's also trying to understand, like, the causal connections and the parameters and how this all connects up and just, like, pushing for a more ambitious vision. And techniques like LIME and CHAP may be useful, may not be useful, but they're more like a tool you would apply in some situation. It's like, for example, attribution patching. We'll probably discuss more later, but very roughly, it approximates a causal intervention using gradients. I have a blog post introducing it, and, also, there are a bunch of other papers that do similar things from previous years because everyone always reinvents everything. And, this is, like, quite similar to LIME in spirit. I think LIME has a different approach to, like, finding the linear approximation, but gradients are really cheap, and, empirically, they seem to work basically fine. But with act with attribution patching, you often apply it to the model activations, rather than model inputs. I believe, SHAP is kind of similar. It's like an attribution method. We've got a bunch of inputs or something, and you're like, what is the Shapley value by which it affects the output? And, again, this is, like, a reasonable tool you could apply in the model to, like, neuron activations. I have not actually seen it applied. I'd be kind of curious to see what the results are. Ideally, you have a more rigorous way of doing the thing, and then you try using a cheaper tool, and you see if it's a good approximation. Though a kind of constant challenge in interpretability is, what even is the ground truth? Okay. So going back to the question, I think in some ways, mechanistic interpretability is kind of continuous with what's come before. Like, ultimately, goal is to take a model and try to understand what happens inside. There are some cultural differences, like different people with different ideas. Like, originally, it was in vision and then moved to language models, and there are some people who are doing language model interpret forever. But it's kind of more of a perspective and philosophical difference than necessarily, like, techniques. It's not just introducing a new technique, is what I'm trying to say.[10]

[10] Datamodels: Predicting Predictions from Training Data — paper

Tim Scarfe 00:42:51

Yeah. I mean, I was speaking with Andrew Elias from MIT, and and he's working on data modeling, which I guess is is not too dissimilar to something like line Lyman's chat where you actually model, you know, how changes in in in the dataset affect the overall predictive architecture. But there's always this thing that it doesn't scale very well. Right? And I I guess the question would be, you know, what what would it mean to reverse engineer a language model?

Neel Nanda 00:43:17

So there's kind of 2 things that I think would both be reasonable senses of this word. The first would be you just fully understand the model. You have a human comprehensible algorithm that acts basically exactly like the model, and you've, like, reverse engineered the premises. And this would be an ungodly massive program. Though an interesting thing is that I expect it to be, like, really wide rather than deep. Like, I expect there to be lots of different programs that kind of happen in parallel, and then most of them are discarded and, like, a few are used such that there's, like, some hope there. But I this seems incredibly ambitious. I don't really think this is realistic, unless we automate it enough, we can just have another LLM do this. And the sense which I think is more realistic is given any input, I will be able to give you a story of the computation done in the model to produce an output. And, to me, this kind of thing is a lot more amenable to, like you just look at the activations at each step, and you do causal interventions to understand how they depend on things, or maybe you even look at the weights and just it's a MapMul. We know how MapMuls work, matrix multiplication. And yeah. Maybe, yeah, maybe this is also just a good thing to add to my previous answer of, like, differences with other kinds of interpretability. Like, to me, the neural network is some kind of substrate that represents an algorithm in its weights somehow. The activations at each step in this algorithm. And I want to understand as atomic a step as I can using tools like activation patching that I think we'll discuss more later. I don't just want to treat it as a black box going from, like, the start to the end. I want to get, like, as zoomed in a view as I can and, ideally, piece this together into, like, a this happens, then this happens, then this happens, then this happens, and you get from the start to the end.

Tim Scarfe 00:45:31

Yeah. You said something which was along the lines of I say similar, which is that, you know, neural networks are wide but shallow. And, you know, symbolic methods are, you know, are very deep, you know, very narrow. And there's always this notion that that they're quite blown up circuits. And then there's, you know, is MEC Interp kind of identifying an algorithm in an input sensitive way? Because if you think about it, surely it must be a superposition of algorithms because the entire model is this gnarly mess of all of these things mixed together. And we're I I guess you're you're suggesting that given given an input example, we look at the behavior of the model, and we try and infer what the algorithm was that took place.

Neel Nanda 00:46:16

Yeah. So yeah. So there's maybe 3 lines of research that I think are worth calling out here. So there's what people might think of when they think of interpretability, like, Mekinterb, like, reverse engineering a thing from its weights, like, fully mathematically understanding it. And, like, we've I don't know. I did this for a tiny model doing modular addition. Some people have done this for other tiny systems. We made a bit of progress on language models, but, like, I basically think no 1 has satisfyingly done this for language models, and it definitely does not seem to scale. So I'm a bit pessimistic on this, though I hot out some hope we can rescue it at some point. There's the causal intervention school of thought. That's like I think of the model as a computational graph with nodes. Like, each attention head is a node. Each MLP layer is a node, or maybe each neuron is a node. And I do causal interventions where I change 1 node from 1 input to another input, and I see how that changes downstream ones and how that change cascades to the output. And, yes, this is this is fundamentally very input sensitive. Often, you'll do this on pretty narrow distribution, like sentences of the form, person a and person b went to the location. Person a gave a object to and it needs to come up with person b. And, like, you know, that's a very narrow context in some sense. The relevant paper there is interpretability in the wild by, Kevin Wang. Yeah. And so this is very input dependent. Partially, it's input dependent because model components kind of do many things, and they're like the jargon is polysemantic. On a narrow distribution, only 1 of these might turn up, but, you can't make a general claim. And then the third family is work with sparse autoencoders. So these are basically a technique to take activations, that are full of all kinds of stuff and mean lots of different things and to decompose it to a larger and sparser and, importantly, monosemantic representation. Like, you've got a bunch of latents, sometimes called features, but I don't really like that word, which we might get into later. And each of these latents hopefully corresponds to some concept. And by itself, this isn't giving you an algorithm.

It's just studying activation. It's just telling you what variables are here at this step. But I think there's a lot of exciting work to be done in, converting that doing circuit finding with SAEs as your notes. And if they truly are monosematic, then this feels like it might get you something that I would consider more input independent, though it's kind of complicated, and there's various improvements to this, which we can discuss later. But you could maybe think of the recent history of Mechanterp as people really wanted input independent algorithms, and that was really hard. And we had lots of success with, like, input dependent stuff, which still seems good enough to be useful. And a lot of people are working on that, and we're hoping we can get less and less dependent enough to be useful or that there's a lot of tasks where you don't need to be input independent. You can just study that domain. And I think both of these are likely reasonably likely to work out.[11]

Tim Scarfe 00:50:04

Yeah. So, you mentioned, polysemantic and and monosemantic, and may maybe we should just quickly, define that.

Neel Nanda 00:50:10

Yeah. So the idea is you can think of any number in a model as being a detector of the input. It, like, is big on some inputs, and it's not on others. This could be, like, a neuron, or just any other element of an activation or even a projection onto some direction. And, we call this monosemantic. If there is a shared property of all of the inputs that significantly cause it to light up, and we call it polysemantic if there do not does not seem to be a shared property, or there's, like, several clusters, or maybe it's a complete mess. And this is always hard because this is an inherently subjective definition. Like, what does it mean to have a shared property? And my typical answer is just like, I don't know, man. It's normally pretty obvious in practice, And there are some edge cases where you might mislabel a thing because you missed the pattern, but, like, it seems basically fine. But some philosophers in the audience may be screaming at me right now.

Tim Scarfe 00:51:24

Yeah. Isn't it 1 of the benefits, I guess, of neural networks is that I mean, I get you can call it subsymbolic that the that the knowledge is entangled and distributed over many shared neurons. And I guess to interpret them, we need to disentangle them. But maybe the brain works in a similar way as well. Mean, the what what you're describing here is that there's this kind of set of circuits that get activated in some kind of task specific way, and then we can disentangle

[11] Interpretability in the Wild: Circuit for Indirect Object Identification in GPT-2 — paper

the representations such that they have an intelligible single meaning, and then we can use that to reason about the circuit or the program that ran in in in the network.

Neel Nanda 00:52:02

I think it is an important fact about neural networks that they can kind of do a bunch of things at the same time with, the same component. Like, an attention head can do different things in different contexts. By at the same time, what I mean is the same component on different inputs can do different useful things. Not on 1 input, it can do 3 things at once. That's much harder. And so the jargon for this is superposition. So maybe let's zoom out a bit and discuss some empirical observations about neural networks before I try to explain what we think is going on. So, empirically, neural network components are often polysemantic. They respond neurons will respond to many different things. Empirically, concepts are often distributed. They often seem to be represented as often linear directions in activation space. Like, if you take a linear combination of neurons, that lights up when the concept is there and doesn't light up much when it's not there. And what we think is going on is, superposition.[12]

Biological Evolution Parallels

Neel Nanda 00:53:20

The idea that there are more concepts than dimensions. Each 1 is linearly represented with its own direction. I I'll justify the linearity assumption more in a bit. And these are all kind of adding together in a way where you can lossily extract 1. You can project onto its direction. There'll be other things with non 0 dot product, which will interfere, but the interference is kind of tolerable. Like, it introduces a bit of error, but not so much that it wasn't worth having that feature there at all. And my guess is that this is actually a pretty important part of why neural networks are more efficient or are just so effective. And interestingly, there's a sense in which transformers or, like, residual networks in general are a lot better suited for superposition because there's lots of addition and reading from this shared residual stream, in a way that lets it represent a ton of things in linear superposition. While if you have nonlinearities on there, it gets a lot more messy. And so, yeah, this is a mechanistic hypothesis that tries to explain the polysemantic and distributed things. There's also a bunch of different senses of superposition. You can have representational superposition where, this is when you've got, like, some activation that's representing stuff, and it's just rep it's just

[12] A Mechanistic Interpretability Glossary — website

squashed in more than it has dimensions. For example, the embedding matrix of g p d 2. It's 50,000 by 768 for the smallest 1, so this means that it's fit 50,000 tokens into a tiny space. But it still clearly kind of knows the difference between the tokens. There's computational superposition, and this is like when you have, something like an ML pillar, like a matrix multiplication and a nonlinearity, and it computes more new features than it has neurons. 1 example where I'm pretty confident this is going on is facts. Like, models know a ton of facts, but and I'm just would be shocked if they knew if they only knew as many facts as they had neurons. And this seems to be represented in some kind of computational superposition. Actually, I have a investigation, called fact finding, where we tried and, basically failed to decipher how exactly the computational superposition worked mechanistically here, but I'm pretty confident it's going on. And the final kind would be kind of circuit or weight superposition. We have a bunch of different algorithms in the parameter matrices added together. It's, like, less obvious to me this is a big deal because there's just, like, n squared parameters for every, like, n dimensional activation, which is more space. But it does seem to be there a bit. So, like, we find, when we, say, take like, use sparse autoencoders or a variant called transcoders and multiply weight matrices together, where we think the start is interpretable, the end is interpretable, but they're much bigger, you often get connections between things that seem semantically totally unrelated, but, like, don't co occur in practice. Like, this thing never lights up when this thing also lights up. And we think this is just this is just representing stuff, and you get interference. It's really annoying, but it's a thing.[13]

Tim Scarfe 00:57:12

You know, like, in the the physical world we live in, you can do analysis at multiple scales. So you can look at the mitochondria in your body, or you can think of yourself as an agent or we can think of the ecosystem and so on. And I guess it's a similar thing with neural network analysis that even the inductive prior of a transformer, you're talking about residual streams and adding things together and so on, and and this is a mode of analysis. And I think it is the case that you could take any transformer and you could represent it with a blown up MLP. So there is an MLP that will do the same thing. And then technically, your abstraction or analysis that you use for the transformer would still work for the MLP, but the MLP is a completely different space. But it goes back to what we were saying before that neural networks are very wide and shallow and that leads to this kind of confection of little unintelligible circuits. And could there exist some much more abstract decomposition of a neural network that would be far more kind of explainable? You know, it'll be a better theory of what's going on in the neural network.

[13] Fact Finding: Reverse-Engineering Factual Recall on the Neuron Level — paper

Neel Nanda 00:58:13

So essentially, could there be a different architecture that's a lot more inherently interpretable?

Tim Scarfe 00:58:18

Maybe an architecture or maybe just maybe a type of analysis that would better explain guess humans need to have quite macroscopic priors to understand things. You know, when we get to the real low level, it seems increasingly unintelligible.

Neel Nanda 00:58:35

Yeah. So I am excited about research that is trying to find these, like, more macroscopic, higher level things. 1, there was there was some interesting work with image models. There was this paper called branch specialization that found that so a fun fact about the original AlexNet is that it has kind of 2 separate branches because they just had 2 GPUs with bad interconnect, and they don't really intersect very often. And these kind of ended up specializing. Can't remember them to what. Like, I think 1 was doing colors and 1 was doing shapes or something. And they found that another image model that wasn't trained like that still kind of had neurons clustering into, like, parallel branches. And if we could find this kind of macroscopic structure in LLM, that would be super cool. A problem with this is that superposition is most effective with things that don't co occur because if they're both happening at once, the interference gets way worse than if 1 happens and the other doesn't happen, and you need to just tell that the first thing happened and the second thing didn't happen. This means that if you had, like, 2 modules, like, the biology neurons and the I don't know. Generating news articles about sports neurons or something. These would actually be great to do in superposition. So, like, the structure is gonna be super eligible to us. And 1 hope is that with sparse autoencoders helping us disentangle these things, we can do more to find this kind of high level structure, and that's a direction I think could be pretty cool for people to investigate. It's not something I've seen that much work on yet.[14]

[14] Branch Specialisation (Distill Circuits) — paper

Universal Circuits and Induction Heads

Tim Scarfe 01:00:31

I mean, it just blows my mind. Have you seen that that there's a type of you can decorate tables, and you you put water on them, and you put 2 electrodes either side, and you just see this kind of, you know, electric patterning that burns burns a kind of tree structure. Looks like a lightning bolt, like a tree structure. You know, we're we're done. It's incredible. But the thing we're dancing around here is that, you know, neural networks are mostly grown and the growth process can be influenced through inductive priors, of course. But when you really dig into it, you just see it it's I'm I'm really stretching the analogy here, but a bit like evolution. Right? Just these weird specializations and local dynamics that form during the training process, and then it's it's kind of like a a living process in some ways.

Neel Nanda 01:01:21

Yeah. So I think I think evolution is actually a really good analogy. So there's a sense in which biological organisms are the subject of a bill of, like, 1000000000 year long optimization process. You have evolution optimizing for something like inclusive reproductive fitness, And you kind of randomly move around in DNA space, and you end up with the human brain. Like, what? And when we look inside biological systems, like, they often make sense. There's, like, structure, and there's organs, and, like, we've learned so much about our bodies, and then there's so much that we're so deeply confused about. And also just lots of, like, random dumb stuff, like the laryngeal nerve that, like, goes from here upwards and then down to the place it ends Yeah. Which is particularly funny in giraffes. And, yeah, just like a bunch of stuff where it's like, oh, man. If I was designing this, I would not have done that. And I bet there's all kinds of stuff like that inside neural networks. I mean, you'll often observe something like, this is kind of weird and confusing. I don't know what to do with it. I'm gonna move on. It probably doesn't matter that much. Or there's kind of weird phenomena where I don't see any real reason to have it, like self repair. If you delete a layer or attention head, often other layers will change their behavior to compensate, to, like, recover the performance. And, like, what? This happens in like, this makes sense if you're training it with dropout or something, like, thing that does that. But, like, why does it happen in models that don't have that?[15][16]

[15] In-context Learning and Induction Heads — paper
[16] The Hydra Effect: Emergent Self-repair in Language Model Computations — paper

Tim Scarfe 01:03:07

Yeah. And that and that's quite biomimetic as well because, I mean, the brain has the same thing. If you have a stroke, different parts of the brain can kind of take over that function. And a lot of self organization and self repair is because you have these little atoms that can be repurposed to, you know, to do completely different things. And 1 thing that I often think about is, you know, imagine we can't do counterfactual analysis with real world, wouldn't it be great if we're good? But imagine if we could just run the evolution on planet Earth again. You know, we might we might not evolve or or maybe there's something evolutionary evolutionarily fit around having, you know, bipedal walkers with big brains and and so on. And I guess you must see this with neural networks because you are seeing the same architecture trained different times for longer, for shorter, on different types of data. And I guess, are you seeing the same kind of motifs coming up again and again?

Entity Detection and Knowledge Boundaries

Neel Nanda 01:04:03

So yeah. So there's this general idea of the universality hypothesis that circuits are universal and will just recur in models trained on the same thing. And, like, the strong version of this is empirically false. Like, there are differences between models. Like but it seems like a weaker form might be true. Like, there's some things that recur, or there's, like, some small set of things and some sample of those appear. Like, I supervise this great paper from Bilal Chuktai called A Toy Model of Universality that well, actually, an important part of that paper turned out to be a little bit wrong and was corrected by a great follow-up work from Dashiell Stander. But the universality part still stands, which was basically we were studying algorithmic models. And, in these algorithmic models, we had, like, a couple of different algorithms they could use, like, 5 or something. And every time you trained it, it just gets a kind of seemingly random sample of those 5. And no 1 studied this enough on language models to really know if a similar thing happens, like my bet is it probably does. And there's kind of some recurring motifs. So, for example, induction heads are, like, a super simple kind of circuit that basically let you take something like I don't know. If a model sees, the word Tim in a sentence, it's kind of not simply what comes next. No offense. But if it sees Tim's scarf previously and it sees Tim again, it knows that scarf's likely to come next. And, induction is just like a simple 2 head circuit implementing this. And, these seem to occur in basically every model I've ever looked at. In the relevant, paper where we studied this, led by Catherine Olson, we looked at a

bunch of internal anthropic language models, up to 13,000,000,000 parameters, and it was there in all of them. I've got a blog post ish thing with looking in, like, 40 open source models, and it's also in there. It's like, it's sometimes true. I think sparse autoencoder universality is, like, a pretty interesting direction. Like, how do the features compare between models and training runs and datasets? Like, comparing code models to, like, normal language models could be cool.[17][18]

Tim Scarfe 01:06:44

Yeah. I mean, this is another thing I I think about a lot is this concept of, you know, almost platonic knowledge that the universe might be generated with some kind of computer program, and we somehow acquire that knowledge, you know, which does lead to the thought experiment that if there were another civilization, maybe I mean, the world works a certain way. And I'm pragmatic about it. I think a lot of knowledge is constructed and social and relativistic and so on, but it it feels like there there are there are some, you know, guidelines around how the universe works. But on on the knowledge thing as well, you were just talking about facts. So right now, neural networks like language models, they are incredibly good at memorizing knowledge. But what they don't have is that degree of certainty. They don't have that epistemic factfulness. And people have done things like retrieval augmented generation and so on. But do you do you think that's an in principle problem? Or or do you think potentially in the future that there could be more certainty about what knowledge the the model has? So

Neel Nanda 01:07:50

I would actually say that I think we're already seeing some meaningful progress here. So, like so there's I think there's 2 problems here that are important to distinguish. There's the model knows something, but that is a false fact versus the model doesn't know anything, so it falls back on the general language model prior and just babbles. I personally consider the first 1, like, out of scope, and I consider the second 1 to be what we mean by hallucination. Mechanistically, I don't expect any difference between the first 2. But in the same way that, like, I don't know, I have lots of false beliefs, I'm sure. I you know, smarter models will have fewer false beliefs, but, like, I don't think that's gonna fundamentally go away. And I think that I mean, just mechanistically, producing a bunch of stuff is, like, different. Like, there was this great nature paper recently from Seb Farquhar on semantic entropy, which is basically just you generate a bunch of things, you group the ones that mean the same thing together, and then you take the entropy of that distribution, and this turns out to be a pretty good sense of how uncertain the model is. And there was a fun follow-up paper using it to train

[17] A Toy Model of Universality: Reverse Engineering Group Operations — paper
[18] Grokking Group Multiplication with Cosets — paper

a probe, which seemed to do a decent job of predicting when models are hallucinating. And I'm currently, supervising a project from Javier Fernando and Oscar Rebesso, look trying to understand mech interpret hallucination in more detail. And they found this super cool thing in what we haven't released yet, called, like, int like, an entity detection circuit. So, like, the model has there are sparse autoencoder features for I recognize this movie name and for I don't recognize this movie name. And these are, like, causally relevant to whether the model will say, I don't know or just babble or, like, tell true facts or or babble for, like, the ones it does know about. Oh, sorry. I think it's if it knows about the movie, messing with this feature makes it say, I'm sorry. I can't help you. If it doesn't know, it will normally say I can't help you, but this can get rid of that, and it will instead babble. And so, like, there seem to be mechanisms here. Like, I think there's a lot of progress that can be made.[19][20]

Tim Scarfe 01:10:20

That does really interest me. I mean, I was interviewing Akbir Khan, and, yeah, he's a really cool guy. He's a really cool guy. Yeah. So his debate paper was, you know, 1 of the papers of the year at ICML. And essentially, it was about having a a kind of a pool of agents, you know, almost like a judge and a couple of critics and getting them to argue it out over 10 iterations to get closer to the truth. Right? And we see now with reasoning, I mean, to me reasoning is about closing a knowledge gap. So I I I don't know something. And the trick is is telling the model I need to start prompting myself to reason now because I know that I don't know. And that's the thing, isn't it? Do you think in principle that a model might know that it doesn't know?[21]

Activation Patching as Causal Intervention

Neel Nanda 01:11:07

I think yes. I mean, I think the work I just described is an example of such a thing. It is distinguishing between entities it knows and entities it doesn't know, at least in some narrow domains like movies. We in the fact finding project I mentioned, we also found that, like, when you, give the model a fake athlete's name, it kind of does seem to act differently than if you give it a known athlete's name. It was actually kind of interesting. We found that the early MLP layers would still generate a kind of sport direction, but that the attribute extracting attention heads, which, like, look at the athlete's name where the factors looked up

[19] Detecting Hallucinations Using Semantic Entropy — paper
[20] Do I Know This Entity? Knowledge Awareness and Hallucinations — paper
[21] Debating with More Persuasive LLMs Leads to More Truthful Answers — paper

and move it to the end, wouldn't look for the unknown names, even though the MLPs were still producing, like, hallucinated spots. This is kinda cool. I wonder what the role of, you know, reasoning and thinking out loud here is because, you

Tim Scarfe 01:12:08

know, again, we can debate whether or not the models internally do it or whether it's a form of externalization. So you let the model think in a in a system 2 way, and that's almost how it reconciles, you know, what it what it knows internally into some kind of calculus that it can kind of, you know, reason with and and perform rationales and so on. So I guess from a MechInterp point of view, how how does it affect the process letting the model perform some rationale before you then analyze it?

Neel Nanda 01:12:39

As in you ask it for a fact, it gives an answer, and then it does a bunch of introspection on the answer?

Tim Scarfe 01:12:44

Well, yeah. I mean, my my intuition is that perhaps that the model is quite gnarly. And if you let the model ponder and consolidate and think about what it knows, will it be able to better know what it knows and what it doesn't know?

Neel Nanda 01:13:00

Inference time compute is helpful. Feels like a pretty uncontroversial statement nowadays. I don't know if I got a more interesting answer than that. Like, it will have more chances to notice something going wrong. And I think I don't quite know what the circuit for is this fact true looks like, but it wouldn't surprise me if it's, like, at least a bit different from the recall a fact circuit, and that there are some facts where it can identify that it's false without actually knowing the answer or something. Like, it just kind of babbles something. So, like, I've gotta say something, but then it looks back, and it's like, that's probably sus, or, like, I'm not really sure. I don't know if anyone's really looked into this, so I'm purely speculating right now. But

Tim Scarfe 01:13:54

Yeah. I mean, the only intuition I have is it it feels like a lot of the reasoning we do is actually a form of patterned tool use. I mean, language is a tool much like the, you know, we were saying earlier the memetic equivalent of a physical tool, you know, like scissors. And we learn how to reason, you know, we learn it at school. So we learn to apply all of these different rationales and applying these rationales in the token space kind of helps us make sense of what we know deep down in in our minds. And maybe there's an analogy there to language models. I don't know.

Neel Nanda 01:14:26

Possibly. I mean, I think there's various kinds of deduction you can do on certain facts. Like, ah, I claimed this, but, like, maybe I'll recall a bunch of facts about the claim that I just made and, like, see if any of them have any bearing on this. And, a general fact about language models is just you kind of only get 1 pass through unless you're doing some kind of chain of thought or something. So, this means that you just can't do that much computation in a single forward pass, and you can do much more if you just pass on a bit of information that you got with the first bit of processing, then the second bit, then the third bit, etcetera. Or possibly there there might not even need to be communication between the different bits of processing. It's just easier for them to happen on separate tokens so it doesn't just interfere with itself a bunch in the same place.

Tim Scarfe 01:15:25

So for folks at home who wanna, you know, get into Mecanterb, I mean, this this is it's still quite a nascent field. I mean, obviously, it's getting much more mature now. But what could folks at home do to get started?

Neel Nanda 01:15:36

So I think there's the field is growing, but there's still a lot of core problems to work on and stuff to be done. I yeah. So I think in terms of reading papers, I have a reading list that we can put in the description. Also, you just Google Neil Nanto MacIntyre reading list. I'm sure you'll find it. I also have a guide to getting started in the field, though it's a bit outdated, unfortunately. And the Arena has, like, a fantastic set of coding tutorials that we should link to as well. And I basically recommend going through, maybe skimming, like, a paper or 2 to just see how interested you are in it, doing the arena tutorials, like, get your hands dirty and understand the tooling, and then do a mix of reading papers and doing experiments kind of riffing off of those papers. I also think it's a lot easier if you have collaborators, or just people to chat to. The Illustra Discord, the mechanistic interpretability Discord, and the open source mechanistic interpretability Slack are all great places there. And, yeah, I encourage people to have to just I don't know. Do a small project. Write a blog post about it. Put it on your website or somewhere like Less Wrong or somewhere else. Put yourself out there a bit. Try to get feedback from people. But more importantly, just, like, get your hands dirty and actually try things and follow your curiosity rather than just reading 50 papers.[22][23]

Tim Scarfe 01:17:20

So what is activation patching in contrast pairs?

Neel Nanda 01:17:24

So the the goal that the activation patching technique is trying to solve is attributing some model behavior, in particular, some kind of numerical output, like the log prob of the correct answer, to some model components on some data distribution. Like, how important is this attention head or this SAE latent or this neuron for the model answering something? And so there's, like, a lot of ways you can do this. And, generally, you want to be causally intervening on this component to change its value. But if you got a chunky component, like a head or a layer who's got quite a big output that's like a vector in a high dimensional space, it's kind of not clear what you should replace this with. So, like, the default thing would just be replace it with zeros. Like, this is kinda what Dropout does. This is the idea of knockout or, like, ablation. And, the problem is this will sometimes just break a model because this is a very off distribution. So there'll be components that aren't relevant to a task, but are, like, just there. There's, like, a bias term or something. So for example, in t p d 2 small, MLP 0 is basically always

[22] Concrete Steps to Get Started in Transformer Mechanistic Interpretability — website
[23] Eleuther AI Discord — website

used to, like, enhance the tokens. Like, it doesn't seem to do much, but, like, if you delete it, everything breaks. Because it's like the output is kind of added to the embedding to be, like, the effective embedding that everything else sees. And so the next level up would be mean ablation. You just replace it with the mean over some corpus. And, like, I think this is a lot more reasonable. But an even more interesting thing you can do is, activation patching, where you have, 2 inputs that are, like, similar, but different in key detail. For example, the Eiffel Tower is in the city of, and the Colosseum is in the city of. These will have a different answer, Paris and Rome. You have some metric, say the difference between the log prob of Paris and Rome. This is quite nice because that's equal to the logit difference of Paris and Rome because maths. And then you swap some activation value from Paris to from the Paris prompt to the Rome prompt, and you see it's what doesn't make the other thing say Paris less? The first 1 is called denoising because you can think of the Roam input as, like, the bad noisy input, and you're, like, denoising 1 component. You're replacing it with yeah. You're, like, replacing it, with the true thing. And going from, Rome into Paris and seeing if it damages Paris is called noising because it's like you're kind of messing up 1 component, and you're seeing if it's important. And the really nice thing is that you can have kind of your choice of baseline. So, like, a pair of prompts like this is called a contrast pair, where you want them as close as possible apart from some key detail. Because this means that things like the I'm doing factual recall right now feature is still there. The I want a city feature is still there, but the, like, which city is it in bit is not still there. And you can also, but you can have different prompts that have different changes. For example, the Eiffel Tower is in the country of. Now

Golden Gate Claude and Feature Amplification

Neel Nanda 01:21:06

you're analyzing the, like, relationship part of the factual recall. And activation patching lets you have this really fine grained tool for analyzing different kinds of information. The denoising versus noising is actually quite an important subtlety. So you can think of noising as being like, was this bit necessary for the computation? Or, like, at least was it useful? If I get rid of it, does anything get damaged? You can think of denoising as, like, was this thing sufficient? Like, does the output of this node from then on cause the output we want? This does not mean that it's the only relevant node, but it means it's kind of enough of an information bottleneck to contain the key info. Like, if you have kind of 3 steps in the process, denoising any 1 of those steps should be enough. And you kind of want these in different situations. So for exam-

ple, if you think you found a circuit, this, like, 3 stage thing with, like, a few nodes of each layer, You can test this by either noising everything not in the circuit and seeing how badly it breaks, which is a way to test the whole circuit at once, or you can denoise kind of each slice of the circuit at a time and see if that is sufficient. It doesn't make sense to pat to, like, denoise 2 slices because the the second slice kind of doesn't care what the first slice is doing because you're just patching in its values. And, yeah, I think activation patching is, like, a really cool and powerful technique. It masquerades under many, many names, like causal mediation analysis in the paper where I first saw it from Jesse Vigg in 2020, or interchange interventions, from Atticus Geiger, or resample ablations, which is mostly just used for the noising part, or causal tracing, which is used in the Rome paper. And there's just so many names. It's really annoying. I personally like activation patching and just try to converge on a name. For people who wanna learn more about it, I wrote this kind of tutorial piece with Stefan Heimersheim called, like, how to use an interpret activation patching. It's not really a research paper, but it's just, like, an intuition dump of how to think about this technique. And I think it's just, like, a pretty powerful powerful tool that is useful in a bunch of settings for trying to understand model components. Oh, and the final thoughts on that is that, if you're using this in practice on a larger model, it's often, quite expensive to activation patch everything. So the technique, I recommend is attribution patching, which is basically you approximate that using gradients. And, I have a blog post on this, and my team put out a paper called ATP Star led by Janos Kramer, kind of measuring in detail that this is a legitimate technique that works and providing some improvements, especially when dealing with attention layers. And because this uses gradients, you can kind of patch everything in a single backwards pass rather than needing to do, like, a separate forward pass per patch. So this can lead to, like, pretty enormous speed ups, though it does have accuracy problems, especially nearer the input. My intuition is that the embedding space of models just isn't isn't nice in the same way that, like, its internals are later on, so gradients just tends to not work as well because it's just like there's, like, 50,000 discrete points in space, and it's not really locally linear. It's like, why would it be?[24][25][26][27][28]

Tim Scarfe 01:24:52

So quick digression on our friend Grant Sanderson. I'm 1 of his biggest fans.[29]

[24] Causal Mediation Analysis for Interpreting Neural NLP — paper
[25] Causal Abstractions of Neural Networks — paper
[26] Causal Scrubbing: Rigorously Testing Interpretability Hypotheses — paper
[27] Locating and Editing Factual Associations in GPT (ROME) — paper
[28] How to Use and Interpret Activation Patching — paper
[29] Attribution Patching: Activation Patching At Industrial Scale — paper

Neel Nanda 01:24:59

He's so great. People who don't know, Grant runs through the YouTube channel, 3 Blue 1 Brown. They make great videos. I'm a fan. And they yeah. Grant mostly does lots of maths videos, but also was doing AI videos. And 1 day, I was like, Macintup has lots of pretty ideas and visuals and is kind of maths unusually mathsy for an AI topic. Why don't I cold email him and see if he's interested? And we had a lovely chat. 1 of the things he was thinking about was how to make his transformer MLP video, and he was looking for, like, a good motivating example. And I thought that my fact finding work was actually a good example, and he seemed to agree. And so he discussed some of that. It's also just, like, a really great video and channel, and you should just watch all of his videos, to be honest. But you should especially watch that 1 because it also has a bunch of stuff on superposition and how to think about this, and it's just way better animated. And it's just better than hearing me talk, to be honest. You should just pause this video and go to that.[30][31]

Tim Scarfe 01:26:11

No. Don't pause it. Wait until the video's out, and then go over there. Quick pause on Gemma as well. Yeah. So as I understand it, you I guess this was inspired by the original microscope project. Was Chris Olahh involved in that?[32]

Neel Nanda 01:26:27

Yeah. I think that was Chris Euler's team.

Tim Scarfe 01:26:28

Yeah. Yeah. And and you folks have done something similar for inspecting, Gemma. How how does that work?

Neel Nanda 01:26:34

Kind of. So, we did this project called GemmaScope, and this is basically a family of several 100 open weight sparse autoencoders on Gemma 2 because sparse autoencoders are a pain to train for reasons we'll get into later, and we thought this would enable better academic Mechanterp research. There is a website called Neuronpedia, who we are not affiliated with, but they're great, and I love them. And they do things like have a page for every latent direction in a sparse autoencoder with the text that activates it and, like, a gent explanation of what it does

[30] AtP*: Efficient and Scalable Method for Localizing LLM Behaviour — paper
[31] How Might LLMs Store Facts (3Blue1Brown) — website
[32] OpenAI Microscope — tool

and things like that. And they kindly made this gorgeous interactive demo for us, which might be the thing you're thinking of that's similar to microscope. Yes. Gemoscope is actually totally different from microscope. We just couldn't think of a better name. But I like the name. Well, talk us through it. So well, so the thinking is basically I kind of think of sparse autoencoders as a microscope for understanding a language model. You pick an activation. You zoom in. You expand it into a sparser and more interpretable form, and you analyze that. This analogy doesn't quite engage with the fact that you then use this to make a reconstruction of the input, but I think it gets the intuitions across. And this is like a microscope for Gemma. Gemmascope.

Tim Scarfe 01:28:14

Tell me about sparse autoencoders.

Neel Nanda 01:28:17

Yeah. So okay. So I think to understand sparse autoencoders, we need to first begin with what is the problem they're trying to solve. So you can think of a neural network as being made up of a of layers, and you pass in some input. It gets converted to vectors or a series of vectors in the case of a transformer. And then each layer transforms this into a new vector or series of vectors. These are the kind of activations in the middle. Often in the layers, we'll kind of in the middle of them have intermediate activations. Though you could argue that, really, this is just not a layer. It's actually several smaller layers, but, like, whatever. And yeah. So we call each of these intermediate variables an activation. It's like it's a vector. We believe that these activation vectors often represent concepts, features, properties of the input, something interpretable, an intermediate state in the model's algorithm. But it's a vector. We need we need some way to convert it into a thing that is meaningful. And sparse autoencoders are a technique that tries to do that. They, basically decompose the vector into a sparse linear combination of some big list, sometimes called a dictionary, of meaningful feature vectors. We, hope that these feature vectors correspond to interpretable concepts, and it's sparse in the sense of most vectors are not part of this combination on any given input.

Model Personas and RLHF Effects

Tim Scarfe 01:30:00

Now many people would have seen the the Golden Gate Bridge example, And that was perhaps the

Neel Nanda 01:30:06

I love him so much.

Tim Scarfe 01:30:09

He he was a cool guy. He was a cool guy. I mean, may maybe just bring that in just for folks who haven't heard about it at home. But but that was that really brought sparse autoencoders, you know, into the masses. Have everyone heard about that?

Neel Nanda 01:30:19

Yeah. So okay. So Golden Gate Claude is a bit complicated to explain. It's very simple in some sense, but I think the, like, interesting takeaway is a bit more complicated. So Golden Gate Clawed, for people who weren't aware, Anthropic took, Claude 3 Sonnets, their, like, medium sized language model at the time. They found the sparse autoencoder feature for the Golden Gate Bridge, just like 1 of these vectors, and then they clamped it to a high value. Meaning, they I don't know. If it normally was, like, somewhere between 0 and 3, they set it to 30. And this made the model obsessed with the Golden Gate Bridge, and it would do things like write recipes that involved a mile long walk along the beach and things like that. And yeah, this was just really fun to play with, and they had a research demo for 24 hours. And I think a common misconception about Golden Gate Clause is that sparse autoencoders were necessary to create this. So there's another kind of simpler technique called steering vectors, where you do something like, give the model a bunch of prompts about the Golden Gate Bridge. Give it a bunch of prompts about, like, London Bridge or something. Take the difference in activations and average that. You get a vector. Add that in. And it's unclear to me whether this would have been better or worse than Golden Gate Clause. No 1 has really looked into this to my satisfaction. But to me, the exciting part of Golden Gate Claude is less the fact that you can achieve this technical feat because I believe simpler methods I mean, even a system prompt would plausibly have achieved them. The exciting thing is it shows that sparse autoencoders were doing something real. They found a concept inside the model. It was decided that it corresponded to the Golden Gate Bridge because that latent variable

lit up more or, like, systematically lit up on things to do with the Golden Gate Bridge, even pictures of the Golden Gate Bridge or descriptions in different languages, and this was causally meaningful. Like, it's very easy to have to trick yourself by finding a thing that kind of correlates with what you care about, but it's not actually what the model is using, and this leads you to a mistaken view of its internals. But by setting this to a high value, they just obviously made the model's behavior different in a really interesting way. And, the behavior seems kind of qualitatively different from what you'd get from a system prompt, which is also really interesting. Though, anecdotally, I've observed kind of qualitatively similar stuff with steering vectors. So I think it's more the act of, like, intervene with a vector inside the model is, like, a powerful thing that we should be exploring.

Steering Vectors and Linear Representations

Tim Scarfe 01:33:27

Yeah. And I was thinking that these models, we only know them to be sycophantic. And they are trained with RLHF, and they they do, you know, they they do what we want them to do. Yet, you manipulate their internals, and all of a sudden, this thing has got a mind of its own.

Neel Nanda 01:33:47

What do you mean by a mind of its own?

Tim Scarfe 01:33:49

Well, it's comes it comes back to this kind of wants, desires, motivations, intentionality type of thing. We only know these models to be very sycophantic. And you you modify its internals in the way that you just described. And now what it does is kind of divergent from what you put into it.

Neel Nanda 01:34:14

Yeah. So okay. So here's roughly how I think about this. Models are good at simulating different personas, often called the, like, simulator's view. They're trained they're pretrained on the Internet and tons of things. They learn a very diverse range of things, which includes the ability to adopt a diverse range of personas. When they are RLHF or whatever kind of post training people use nowadays, you're kind of trying to build and select a persona for it. And many companies go with this kind of fairly agreeable assistance, what you're you're referring

to as sycophantic, and that is a persona. But, like, I think this is in some sense kind of fragile, and I think that they'll often be trained to not break out of this persona easily. Though, I mean, the jailbreak phenomena shows that it's often not that hard with the right prompt. And if you fine tune a model to have a new persona, that seems very easy to do. And this kind of interpretability based intervention is just, like, another kind of thing. But it's not it's not like interpretability has given us a new capability we didn't have before. It's more like, you know, breaking the persona and giving it a new persona was a thing we knew how to do with existing tools. This is just an interesting new tool that's gonna have some different properties that are cool and worth exploring.[33]

Tim Scarfe 01:35:52

But it does seem to indicate a weakness with RLHF because as you say, this simulator's view is that the the model is a kind of superposition of simulacra or or role players and RLHF selectively deletes those role players just leaving the harmless sycophantic ones. And this rather leads to the conclusion that it doesn't really delete many of them. It's quite a brittle way of, you know, making the model present in in a certain way. And actually, all of these other role players are just there, hidden beneath the surface and and you can activate

Neel Nanda 01:36:27

Yep.

Hallucinations and Model Uncertainty

Neel Nanda 01:36:28

I completely agree. I think that is just a true statement about the current state of our ability to post train models. Is this solvable? Kind of unclear. Another fun data point here is I supervised this, paper from Andy RDT called refusal is mediated by a single direction, where we found that you could find a, like, refusal vector by taking prompts like, how do I build a bomb? Prompts like, how do I build a car? Taking the average difference. Unclear if this is a refusal vector or a harmful question vector, but whatever. And then you just ablate this direction in the residuals. Like, you set the projection to 0. And this means the models no longer refuse, and this works in a bunch of open source models. And this is such a simple intervention.

[33] Simulators (AI Alignment Forum) — paper

Tim Scarfe 01:37:23

Can you explain the linear representation, hypothesis?

Neel Nanda 01:37:27

Yeah. So a surprising empirical observation that we've seen a bunch of times with models is that concepts will be represented as some kind of linear directions in space, an activation space. So you can think of this as a linear combination of neurons, though I generally recommend thinking of it more as like 1000 dimensional space where neurons are, like, the standard basis, but you can kind of pick whatever direction you want. Empirically, it is often the case that concepts seem to be detected by linear directions. The simplest case of this is when you've got a neuron that seems to only light up on some things. There've been a ton of papers about this. 1 of my favorites is the curve detectors paper from Chris Olah from when Chris Olah was at OpenAI, which found a bunch of new a family of neurons in a image classification model that just seems to only ever activate on curves of a certain orientation. But, like, as we've discussed, this often doesn't work. Things are polysemantic. There have been other works that have found, kind of more inter like, directions that are not basis aligned, and, like, a bunch of things. A particularly fun 1 is there's lots of papers that try to find truth directions, and there's kind of the just the entire field of linear probing where you basically just learn a direction such that when projected onto it, you detect some concept, and you've got a labeled dataset. 1 of my favorite examples of this is there was this great paper from Kenneth Lee on Othello called emergent world representations where so Othello is a board game like Go or Chess. He trained a model on games with randomly chosen legal moves. So just kind of a chess notation style thing. You see, like, you played in cell 63 and then cell 1 and then cell 17, etcetera. And the model became good at playing these legal moves, and, he found that you could actually probe before the state of the board. But what he found is that linear probes didn't work, but nonlinear probes, like a 1 hidden layer MLP, did. And,[34][35][36]

[34] Curve Detectors (Distill Circuits) — paper
[35] Emergent World Representations in Sequence Models — paper
[36] Emergent Linear Representations in World Models — paper

Sparse Autoencoder Architecture

Neel Nanda 01:40:00

in some follow-up work, what I found was that you could instead, instead of representing it in terms of, like, black and white, it represented it in terms of, does this cell have the current player's color or the current opponent's color? Because that is actually an algorithm that just is useful on every move rather than needing to act differently for black and for white. And when you linearly probe for that, you, yeah, you find that it just works, and you can even causally intervene with these probes. And I bring up this example not because I wanna brag about my papers, but because, I think that this was, like, a really interesting natural experiment where there was an initial paper that seemed to provide some, like, legitimately good evidence for a nonlinearly represented thing. And then in follow-up work, I was like, actually, it was a linear representation hiding beneath the surface. Obviously, this is all kind of anecdotal. Like, we don't have a fully principled study of models that is like every concept is linear. There was a fun paper, I think, from, Robert Shorlash, that showed an example of a nonlinear feature in an RNN, that could be represented. So, like, these definitely could, in theory, happen. My current guess is that, like, most of the language models computation is linearly represented. Quite possibly all of it, but it wouldn't surprise me if there's some weird dark matter hiding beneath the surface. That was a bit of a digression. But the key thing to take away from that is many concepts inside language models seem to be represented as linear directions in activation space, but we don't necessarily know what these directions are.[37]

Tim Scarfe 01:41:58

And we should talk about steering vectors as well. I mean, I know you just you just sort of, like, alluded to it, but give give me an example of

Neel Nanda 01:42:04

that. Yeah. So the idea is essentially, you take some prompts with a property, some prompts with the opposite property or without that property, and you take the difference. For example, you could take I love you minus I hate you. And this just produces a kind of fluffy loving vector that you can add in that, will take a neutral prompt, like I went to my friend and said, and it says really happy, excited things. Or if you subtract it, it says, like, really angry, hateful things. And this works in a bunch of settings. So, I first saw this on

[37] Recurrent Neural Networks Learn to Store and Generate Sequences — paper

language models in Alex Turner's activation edition paper and Kenneth Lee's inference time intervention paper. Alex did a bunch of stuff like sentiment and weddings. Kenneth focused on truth. There was also the representation engineering paper that did in a bunch more settings, and, yeah, it just seems to work pretty broadly. And so drawing this back to the linear representation hypothesis, the key takeaway here, in my opinion, is that if the linear representation hypothesis is true, this kind of subtract 2 things that are kind of related, should isolate the feature you care about. It might also have some other stuff, but if you average, you probably wash that out while preserving the, like, key thing you care about. And then you can just add that in. And another consequence of linearity is that you can just add in more features, and it will process reasonably, which means models can compose concepts that might not have come up during training, but there's kind of circuitry that can deal with this reasonably. A caveat with steering vectors is, so a key hyperparameter is the coefficient of the vector. Typically, you'll want to make it bigger. But if it's too small, it just does nothing. And if it's too big, the model goes mad and spouts gibberish. And no one's really studied exactly why, but I think it's just it's big. If it's too big, then when layer norm scales down the residual stream to be kind of a norm, then because you've now got a large fraction due to a steering vector, everything else gets small, and this drowns out everything else. And when you, like, multiply by neurons, even if the steering vector direction doesn't interfere too much with that neuron, when it's got such a big coefficient, that interference is actually really bad. But, yeah, you can drive models mad. It's you gotta get it right. Tuning. Yeah. I mean, you're kind[38][39][40]

Tim Scarfe 01:44:54

of pointing to a kind of stability analysis. There's like a kind of critical point where beyond that, the dynamics of the model decohere. That sounds like a fascinating research problem.

Neel Nanda 01:45:07

Yep. And, yeah, I think some a project that would be really cool to see, but I'm mildly surprised I haven't seen properly yet, is a kind of chat language model interface with a bunch of steering vectors, either the kind I just described that you get from prompts or, SAE feature vector directions or even something you've optimized for this purpose. This could be things like creativity, factuality, verbosity, things like that. Things people care about in their assistant. How formal versus informal to be, things like that. And you have sliders, people can move,

[38] Steering Language Models With Activation Engineering — paper
[39] Inference-Time Intervention: Eliciting Truthful Answers — paper
[40] Representation Engineering: Top-Down Approach to AI Transparency — paper

and this just changes how the assistant does things. And Neuronpedia has a kind of MVP version of this that is the best I've seen, but I feel like someone could really make a really cool polished version of this. I haven't seen that yet.

Tim Scarfe 01:46:07

How might those directions interfere with each other? You know, like, you might have, faithfulness and morality and don't do bad stuff. Do you do you think there could be weird interactions between them?

Neel Nanda 01:46:20

So it is in general true, as far as I can tell, that if you try adding a bunch of steering vectors at the same time, the coefficient needed to break the model is much lower. And I think this will be a real challenge for getting this to work. Maybe you should, like, fine tune the directions and not interfere with each other or something, especially if you're using the kind of optimizer direction approach. There was a cool paper called ByDPO that seems to have, like, made more effective vectors by doing this. And yeah. In terms of semantic composition, I kind of just expect that to be, like, it's got nontrivial cosine sim because they're kind of similar, which means they'll interfere with each other a lot more. But maybe there'd be some circuitry that gets confused or, like, you push the model in different directions. Mean, I in some sense, saw this with Golden Gate Claw where, like, you asked not to talk about the Golden Gate, so it doesn't want to. You make a talk about the Golden Gate with the Golden Gate vector, and it gets really confused and agonized.[41]

Tim Scarfe 01:47:28

Yeah. I mean, my intuition is that when you when you modify the behavior of a model in in this way, you're kind of pushing it out of distribution, which means you might see a commensurate decrease in capabilities. I mean, you seen anything like that?

[41] Personalized Steering of LLMs via Bi-directional Preference Optimization — paper

Neel Nanda 01:47:41

I mean, yeah. Steering models are, like, kinda jank steered models are kinda janky. Like, their grammar is sometimes worse, or they'll say weird stuff, or they'll just spout random tokens. And, like, generally, you can get a kind of good coefficient where it does the thing you want without going mad, but even then, I would still expect it to have some degradation. I don't know if anyone really studied this with Golden Gate Clord. It's it's kind of hard to study in a sense because it's not models are not supposed to constantly talk about the Golden Gate Bridge, but Golden Gate Clord is. And so you need a test that doesn't unfairly penalize it for that. Like, you could look at things like its MMORU performance and stuff like that. That seems like an interesting thing people should do.

Tim Scarfe 01:48:30

So mathematically, what is a sparse autoencoder doing?

Neel Nanda 01:48:33

Yeah. So a sparse autoencoder is so it's trying to solve 2 similar problems. The sparse coding problem of finding this meaningful list of vectors that we think correspond to concepts in the activation space. This is just like a fixed list. It doesn't depend on the input. And the sparse approximation problem, which is finding a sparse vector of coefficients for this list of vectors that can reconstruct the input. And there's, like, a whole field of study of the right way to do this. Sparse autoencoders are 1. And the idea is it's basically a 2 layer neural network where the middle state is, like, much wider than the input, typically. It has some activation function. The simplest case is ReLU, but I'll discuss other ones later. And you feed in the activation, then some of the hidden latents light up. And the ones that light up and because it's ReLU, most are 0. And the ones that light up, you multiply their decoder vector, like the vector and the output weights, aka the thing we hope is the meaningful feature vector, and that produces the output. And we train this, to reconstruct the input on a bunch of real model activations, typically in the hundreds of millions to billions of tokens. And we have some kind of sparsity penalty, such as l 1 on the hidden activations. I ref and an interesting fact about 2 layer MLPs, both transformer layers and sparse autoencoders, is you can think of each neuron as like an independent unit. Each 1 has an encoder vector and a decoder vector. You project the input onto the encoder, you apply the activation, you multiply by the decoder, and then you add them up. And, I refer to these units as latents. They're sometimes called features, but I personally find this a bit confusing because feature as a word means uninterpretable thing, and latents are sometimes but not always interpretable, and I think it's confusing to assume they are. But yeah. So we take these latents, and the hope is that the

latents correspond to interpretable concepts. The reason, so people hearing this might be like, that's a bit odd. You've purely optimized for reconstruction and for sparsity. You never had the interpretability loss function in there. What gives? So the hope behind sparse autoencoders is that there is a true sparse decomposition into that there is, like, true feature vectors, and activations are a sparse linear combination of those. Such that if we just optimize for finding a good sparse decomposition, it will be at least pretty close to the real 1 and thus interpretable. And, empirically, this often seems to work.

Tim Scarfe 01:51:56

A lot of people at home will know about autoencoders because, you know, go back to the days of MNIST. And typically autoencoders are thinner in the middle than they are on the on the in and the out because you're telling the the the autoencoder to entangle and compress and bottleneck the information, whereas this seems to be the opposite. You're telling it to disentangle and to and to blow up what what's in the middle. But it seems to suggest that the the the model knows that it's entangled features together, and it and it wants to, in this setting, disentangle them.

Neel Nanda 01:52:29

Okay. So on the first part, the autoencoder points. You are completely correct. Generally, autoencoders are like, you have an input. You want to somehow pass it through a constrained bottleneck in a way where the bottleneck is more useful to you, like it's smaller or maybe it's disentangled or whatever, and then reconstruct the input. And often, the reconstruction is just a forcing function to make the bottleneck interesting. But, really, I think you should just think of this as a bit with constraints. It would be dumb to remove the sparsity penalty and train an autoencoder with a wider thing because it's easy. Like, you could just have, latent for every direction in the standard basis, like a positive 1 and a negative 1, and it would just perfectly reconstruct things. And I'm like, yeah. That's boring. But sparsity is actually quite a big constraint. So an intuition for this. Let's say I give you a list of 1000 vectors. If I only let you have a 1 sparse thing, that's just, like, 1000 lines through space. That's, like, a tiny fraction of the dimensionality you're in. Let's say you're in a 200 dimensional space. If I let you have a 5 sparse thing, you kind of have a bunch of, like, 5 dimensional subspaces, unioned together, and, like, a 100 sparse things is still like like, mathematically, this is measure 0. Like, it's an infinitesimal fraction of the larger space, even though it I don't know. It's plausibly kind of close to many points in the space. I'm not entirely sure. But sparsity is just, like, a pretty big constraint, and the fact that you're forcing the model to do that means that the autoencoder actually has some pressure on it. Regarding, like, the model knows, I'm like, what does it mean for the model to

know something? Like, if I try to train a linear probe to detect sentiments, this will probably work. No. It empirically does work, and that is probably in superposition with other things. And, like, does the probe know that it's entangled?

Tim Scarfe 01:54:58

Well, point of order. I think the where that was coming from is, you know, we spoke earlier about the models are not reversible. You know, they they the as these things become entangled together, the model shouldn't, in principle, know how to disentangle them.

Neel Nanda 01:55:13

Ah, so it's more of a mathematical question of how is this even possible?

Tim Scarfe 01:55:17

Yes. It's almost like they shouldn't be invertible. These these operations shouldn't it shouldn't be possible to undo them, you know, going from, you know, going from right to left, if that makes sense.

Neel Nanda 01:55:28

Sorry. I misunderstood. Yes. That's a good question. So the way that so this is kind of sparsity is a constraint again. So, like, if I give you a list of 1000 vectors in a 200 dimensional space, and I tell you, here's a vector in the 200 dimensional space. It's a linear combination of some of these 100,000 vectors. Which ones? Like, that's impossible to answer. Standard linear algebra facts. That's not an invertible function, because there's lots of linear combinations that are 0, which you could add freely. But, like, there probably aren't sparse linear combinations that are 0. And this means that it is it is constrained, like, where vectors can be. And this means that if you have a sparse linear combination, it will often be the case that, for example, if you project onto every vector in this set, you'll have a much higher projection on the ones that are part of your thing. And you you can, in fact, train Sparse automations coders with a tied encoder and decoder. So each latent is you literally dot product with a vector, apply a ReLU, multiply by the same vector, and try to reconstruct the input. Empirically, they perform better if you don't tie them. The intuition here is if you've got some features that are disjoint but highly correlated, then you kind of want encoders that, like, push them further apart than they are by default. Like, there was a really fun, example of this in Anthropics towards monosemanticity paper where, they were looking at base 64, detecting latents. And I think they found 1 that was for, numbers, in base 64, 1 that was for letters in base 64, and 1 that was for ASCII text converted

into base 64. And, you know, you don't wanna activate 2 of these at the same time, so you want encoder vectors that push them apart. And it's, like, an interesting empirical observation that you can do this disentanglement, but, like, sparsity is just, like, a really useful prior, basically. And this is also an interesting property of language. So just the real world is kind of sparse in the sense that it's full of concepts and things. Like, you don't need to be thinking about the theory of relativity if I ask you the name of Ed Sheeran's latest album or something like that. And this means that there's lots of concepts that aren't useful for most other things. If you, I don't know, train a model on a hyper specific task, it's not actually obvious to me that SEEs will be useful. Like, I haven't actually tried them on, a modular addition network, that are trained on exactly that task, but it wouldn't surprise me if they don't do very well.[42]

Tim Scarfe 01:58:40

How do you know, though, if 1 of these latents is interpretable?

Neel Nanda 01:58:44

Yeah. So that's a great question. So so the first thing to emphasize is that, kind of why is this even a question at all, is that sparse autoencoders are an unsupervised technique. That means that you don't tell them what to learn. You just tell them, please be sparse, and you pray that something good happens. And, this would be in contrast to, like, a probe where you give it labels, like this is formal text and this is informal text or something. And this means that at the end, you just get this artifact with, like, tens of thousands of latents, and you're like, what does this mean? And, empirically, some of these latents don't even seem meaningful. Like, the standard the kind of dumbest approach you can do is you just look at the text that most activates a latent, and you look for a pattern. Often, will be a pattern. Sometimes, there won't be. This is, a crude technique. It is known that this can sometimes be misleading. There's a good paper on this from Tolga Balakbasi called the interpretability illusion. And though I don't actually know if I've seen such an illusion for SEE features. In that paper, I think they were just

[42] Towards Monosemanticity: Decomposing Language Models with Dictionary Learning — paper

SAE Core Challenges: Dead Features and Training

Neel Nanda 02:00:00

using basis directions in the residual stream, which you have much less reason to believe might be meaningful. And, yeah, there are just some latents, maybe, like, 20 to 30%, though it probably varies, that just, like, don't really seem to have a pattern. And the kind of standard thing that gets done is either a human or an LLM looks at these and tries to give an ex give it an explanation. You can score these explanations. For example so the idea of having an LLM do this comes from this great OpenAI paper from Stephen Bills called, like, language models explaining language model neurons or something. And their idea was you give it a list of dataset examples, some from the, like, highest bit of the range, maybe some from more in the middle. It produces an explanation, but it will always make an explanation even if there's no pattern because language models be like that, yo. And, what happens is that they then give the model the explanation, give it some more text, say, please predict the activations of this neuron or latent. They were doing neurons, but, like, it's much more interesting on latents. And you see how well it does. You can also do, kind of cheaper things because that's actually quite annoying. Like, you give it 2 texts, you're like, which of these will light up the latent more? You can, yeah, there's various kind of difficulty scales you can do with that. Yeah. There was a nice Luther post on innovations for auto interp from, I think, Caden Yuan. I'm probably butchering his surname. And, yeah, that suggested various innovations like that. Anyway, so that's the kind of looking at dataset examples. Another approach is causal interventions, a la Golden Gate Claude. You make it big. You see what happens. Maybe you make it 0 or negative. You see what happens. For example, if you set a Harry Potter latent to 0 or to negative, it will often lose the ability to answer factual questions about Harry Potter, which is a very fun hacky approach to unlearning. Though, sadly, it seems to perform less well than actual unlearning baselines. And you can also so in my opinion, the gold standard is you give the model a ton of text. You look at all of the time the thing fires, and you then classify all of that by whether it satisfies a property or not. You can sometimes do this algorithmically for, like, is this in Arabic? Is this base 64? The pretraining text is, like, weirdly diverse. So lots of things you would think would work reliably like a regex just totally fail. You could also ask a language model. Does this fit the explanation? Anthropic did that in their scaling monospanticity paper, which I quite liked. So they did that for their Golden Gate feature. You can also handcraft examples and see if that activates it. 1 thing that's often missing from these analyses is, are there times when the explanation is relevant, but the latent doesn't fire? And I think that that's a yeah. I think that that's an area I'd love to see more work in. Anthropic

had a recent mini investigation in their latest monthly updates, I think, where they found oh, often these things are, like, a third of the time when it's about this thing it will activate. So there's a lot we sort of understand about these. But to summarize that rambling answer, there's various things you can do. Looking at dataset examples is 1 natural thing. I highly recommend people go poke around on Neuronpedia, both in the Gemascope demo and the main website, because they have pages with LLM generated explanations from GBD 4 0 mini, so it's not as good as you'd get from, like, the best models and sometimes wrong. Dataset examples, you can even type in your own text and see whether the thing lights up. And just kind of play around, like, get a feel for how reliable this is in practice.

Tim Scarfe 02:04:31

Yeah. I mean, we should emphasize again that this is an unsupervised method. But what you were alluding to, though, is there is potential for kind of automating this process, but there but there will be some brittleness on on on the edge cases.

Neel Nanda 02:04:46

So, I mean, for example, Neuronpedia are taking the 30,000,000 ish latents in Gemascope and generating labels with a g p d 4 0 mini for all of them, and this is great. I think this will be a really useful tool even if it's sometimes wrong. That's, like, another question. It's like, what error rate are you willing to tolerate? And often my answer is, if it's a heuristic tool for a researcher, reasonably high. But then if it becomes an important part of my results, I'll go and check harder. Maybe I should also just comment a bit on why being unsupervised is kind of important here. So a classic mistake in interpretability is projecting your preconceptions onto the model. Like you think it works a certain way. The Othello paper we discussed earlier is an example of this, assuming the features were black and white. My modular addition work is an example of this. I kind of initially thought it would be some nice discrete algorithm, and then it turns out it's actually using discrete Fourier transforms, apparently, and trigger entities. And, yeah, this is very easy to be misled. And the more you have techniques that can tell you when you're wrong, the better. And the more your techniques just kind of let you confirm a hypothesis you kind of already had, the, like, worse it is. And I think that, yeah, it's just sparse autoencoders are great because there can be features we wouldn't have expected in a model that arise. And, like, the fact that it can let you do this kind of unsupervised discovery is great. So, for example, there was a recent blog post from, a bunch of my trainee math scholars building on the, Othello results where they were analyzing sparse autoencoders there. And, and there had been some previous claims that sparse autoencoders didn't recover the board state directions, so they weren't working. But what this follow-up work found is that, actually, the sparse autoencoders were finding a

kind of more granular feature. Like, basically, is this vertical column of the board something where playing in the cell d 6 will work because it takes things in that column? And you can kind of combine these together to get the board state, but, like, this was a more granular thing the model had learned. And, like, I didn't expect that. That was cool.

Tim Scarfe 02:07:23

Yeah. It's like before we were talking about, you know, the the evolution of these useful features, and presumably, you can infer from their presence that they must be useful because otherwise, why would they be there? But when I was reading the Golden Gate Claude's, you know, the the kind of the blog post that they put out, they were talking about some quite abstract features, you know, things like power seeking and deception and so on, and and then you you could kind of click on them and and you could see which parts of the dataset activated those features. And to what extent do you think these models can learn very abstract features like that?

Neel Nanda 02:07:57

Just like, clearly, yes.

Tim Scarfe 02:07:59

Like Go on.

Neel Nanda 02:08:00

Talk to a model. This has clearly got abstractions in there. Like, they're so capable at this point. Like I mean, so concretely, in terms of, like, actual evidence rather than vibes, I think yeah. So in Anthropic Skelemonis fanticity paper, they, 1 of the things I really like about it is it's got all of these kind of rich qualitative analyses of different features and digging into what they mean and their causal effects. And they have things like a this function means addition thing. And if you change that on a variable f and apply it, it goes from multiplication to addition when, like, completing Python code, and I'm like, what? Or they have fourth items and a list feature. And they've also got this section on, like, safety relevant features, which, like, I think are super interesting towards the end. So I think it's so they observe things that seem kind of related to the kinds of things we might be quite worried about in future more capable AI systems, like keeping secrets from its operators, trying to seek power, things like that. And I think this is not actually that scary, and I think Anthropic do a good job of not scaremongering there. The reason I don't think this is particularly concerning right now is that,

you know, these models are trained on characters and books. Those characters will do things like power seek and deceive, and it's just useful to be able to model this to simulate those people. But I also think the fact that we are starting to be able to study things like this with interpretability is really exciting, because I think it's really important. Like, I don't know. Is AGI an existential risk is a pretty polarizing question with lots of prestigious people and strong opinions on both sides, but, like, frustratingly little empirical evidence. And I think 1 of the things that interpretability could potentially give us is, like, a clearer sense of what's going on inside these systems. Like, do they do things like you that we would call planning? Do they have any meaningful notion of goals? Will they do things like deceive us? And the more we can understand how these things manifest, and, like, whether they occur in situations where they shouldn't, the, like, more I think we can learn. And this seems like a really important research direction to me.

Tim Scarfe 02:10:38

Yeah. I I agree. I mean, the reason why that came to my mind was I I looked at some of those activations. Because when you click on them, it shows you in in their test corpus which bits of text maximally activated those latents. And on some of the abstractions, you know, that that that they gave, I looked at the top activations and and they seemed quite kind of low level to me, you know, almost like a keyword match. And of course, know, the the deflation review is that these models are kind of engram models on on steroids or or or whatever. But that might just be an artifact that, know, for whatever reason, the top activations kind of looked quite banal and superficial. But actually, if you look at the whole thing in context, that might confer, you know, what is what you would expect for a deeper abstract to understand it.

Neel Nanda 02:11:24

Yeah. So I think yeah. I agree with that. I think that if you go on something like Neuronpedia, they'll often show you kind of different intervals, like what is between the fiftieth and seventieth percentile of activations. Let's, like, give you some stuff. And this is useful for getting a kind of broader view. Also, just looking at the causal effect is interesting. Like, I don't know. I found things like a feature that lit up on, like, fictional characters in general, but especially Harry Potter. But when I steered with it, it was kind of only Harry Potter related things. And I think that yeah. What do I think?

Tim Scarfe 02:12:15

Well, I think steerability would indicate that it was an abstract. It's almost like the the robustness to its representation post steering kind of indicates to me that it's more than just a keyword matching. It actually understands what the thing is.

Neel Nanda 02:12:30

Yeah. So I think it's it seems pretty clear to me that it's not just keyword matching because we observe things like multilingual features where the same text in different languages lights the feature up. You tend to see these beyond a certain model scale. I mean, not even that big. Like, I think 1,000,000,000 or, like, 500,000,000 is probably enough to start to see signs of it. In Scalimonos fantasti, they had multimodal features, like, the Golden Gate Bridge 1 lit up on pictures of the Golden Gate Bridge. And, like, with my interpretability hat on, I don't find this very surprising because if you can map your inputs into a shared semantic abstract space, you can do efficient processing on them. So, like, of course, this will happen. And there's some interesting work. Like, there was the Do Llamas Think in English? Paper, from, I think, Chris Wendler, and, who I know is a listener of this show. And I think that, that seemed to show that the model decides what to say, and it decides what language to say it in at kind of, like, different points, and you can causally intervene on them differently. And yeah. But I think if you have the kind of engram matching stochastic parrot perspective, you would not predict this. And and I don't really understand the people who hold that perspective nowadays, to be honest. I think it's clearly falsified.

Tim Scarfe 02:14:01

I mean, if if Emily Bender was here right now and laid the chance sorry, laid the child the charge of of a, you know, stochastic parent matching system, what what would you say to her?

Neel Nanda 02:14:13

I guess I would just be like, we've observed algorithms inside these things. Like, you can train a tiny model on modular addition, and it has discrete Fourier transforms and triggered entities. We know that transformers have induction heads that seem to be composing in some kind of actual algorithm. We find these abstract multimodal and multilingual features. And I'm sure you can justify some, like, relaxation of this model where you're like, yeah. It does the easy stuff, but it never does the, like, real hard stuff. So it's a sarcastic ish parrot. But I don't know. The Othello thing is another example. Like, I think that's a clearer example of it formed a world model in some sense. Like, only it ever saw the moves and never saw the boards, but it formed a causally meaningful internal representation of the board.

Tim Scarfe 02:15:05

Yeah. I mean, the the steel man to that argument is that they don't seem to do this compositional generalization. They don't have the invertibility and so on. But then what is an abstraction if if it's not a bag of analogies? And what what is an analogy? I mean, if you if you capture all of the presentations of a concept and link it to a thing called an abstraction or or whatever, and it acts as if it knows the abstraction, and it can reason with that abstraction, at some point, it's a distinction without a difference.

Neel Nanda 02:15:39

Yeah. I think another important factor here is, like, so in my opinion, the claims being put forward are kind of for all claims. Like, there do not exist instances of the thing not being a sarcastic parrot. I totally think there are many instances where it is. Like, you know, they memorize lots of stuff. Sometimes they hallucinate. Lots of that lots of the time, they're just doing basic grammar. Like, the token don is followed by apostrophe t to make don't as, like, an engram. Like, clearly, the models learned this. And I think that we're still figuring out how to steer them, like, how to get them to use the, like, complex abstract circuitry that we want rather than the thing that's really useful for predicting the next token. Like so for example, even, kind of models in the, like, low billions of parameters know how to do addition. You can give them something like a 113 plus 237, and they will often give the correct answer. And this just clearly can't be that useful. It can't have come up that much in the training data, even the general circuit of addition. Well, something like figuring out when a full stop is gonna come next versus a comma comes up so much. So, like, think how much more of an incentive the model has to devote parameters to that kind of thing. And I think we're just not very good at reshaping how that how the circuitry is expressed in a way

that leads to, like, all kinds of dumb things.

Tim Scarfe 02:17:27

Yeah. So if I understand correctly, you're saying that the model at the moment is a big soup. And because of pressures from the architecture and and the data, there are some really useful circuits that are very robust. And then there's almost like a miss like strata, then there's a middle tier of things that are slightly less robust and likely to hallucinate. And then there's just, you know, pure noise on on the outer layer.

Neel Nanda 02:17:52

Robust isn't quite the axis I'm thinking about. It's more, like, kind of abstract or, like, competence or, like, usefulness. Like, there is circuitry that can, like, I don't know, think about complex tasks or produce a plan or something like that. Produce a plan in the, like, chain of thought sense. And that seems like the kind of thing we really care about. While it's enormous dataset of memorized facts, we probably don't. Or I don't know. It's probably memorized a bunch of, like, famous books, and we probably don't care that much about that. And,

Advanced Activation Functions and TopK

Neel Nanda 02:18:43

though maybe we do. Like, plausibly, you do want the model to have memorized the whole bible. I don't know. And I think that yeah. I think that

Tim Scarfe 02:18:57

Well, how how could we meaningfully decode you know, if if if we think of it as a kind of geological strata and there's bits that have I mean, some of it is just knowledge and some of it might be circuits for doing certain types of reasoning and planning and so on. And it's almost like we now now we've got this big thing. Maybe we want to shape it and grow it differently in the first place, but given a big model, how do we factorize it into the bits that we want?

Neel Nanda 02:19:23

Yeah. So I'm honestly not that convinced that interpretability is the correct tool here, though this is a direction I'm interested in people exploring. So in some sense, we already have a bunch of tools for this. Prompt engineering, in my opinion, is basically just trying to give the model the right magic words to get it to use

the circuitry you want and not the circuitry you don't want. Because by default, the model doesn't know what you want. Fine tuning is, like, another thing, and, like, the kind of chat instruction following fine tuning that people tend to do is, like, a kind of very important example of that. And my mental model of fine tuning is that it's mostly just kind of upwaiting existing circuits, though this has been this has not been proven, and I would really like to see it proven. Seems like a super interesting direction. And yeah. I think that you can also think of steering vectors or kind of SEE feature clamping, as another kind of more interpretable elicitation or steering of these models. Plausibly, there's things you could do where you observe that, like, these 2 attention heads connect with each other in this circuit that I don't like. So if I break that connection, like, take this head and subtract the output of the earlier head from that and instead adds, like, the output of the earlier head on a different input, technique called path patching, we might talk about more later. Maybe that would break the capability you don't want in a way that makes the model better, but this is very much a speculative thing. And I don't know. I'm quite sympathetic to the bitter lesson in many ways. Like, just throw a computer at the problem is often a really good solution, and this makes me think that things like fine tuning are gonna be quite a hard baseline to beat unless you have a situation where your data is really bad, like you don't have much of it, or it's got lots of spurious correlations, or it's pretty noisy, and you maybe want to use interpretability, to do better there.

Tim Scarfe 02:21:44

Very cool. So I mean, coming back to sparse autoencoders. I mean, first of all, there was the the the vanilla variant which was described in in towards monosemanticity. But there's a whole bunch of variants that, you know, change the architecture and the activations and so on. And and your team has has been working on 1 of those, which is the jump relu. But, I mean, can you just kind of sketch all of that out?

Neel Nanda 02:22:03

Yeah.

Research Applications in Transformer Circuits •

Neel Nanda 02:22:04

So so the first 1 is shrinkage. So it is mathematically the case that if you are using l 1 regularization, feature latents will fire less than is optimal purely for reconstruction. Because you got 2 objectives, and l 1 always wants things to be smaller, so it'll make things a bit smaller than they should be. This is sad. And, yeah, that's shrinkage. The second 1 is a bit more conceptually difficult. So the idea is, superposition causes interference. This means that if you have some, feature direction, say the dog feature, and you project onto that, and you plot a histogram of the projections, you'll kind of obviously expect to see a kind of dog mound, like, I don't know, between, 3 and 5, there's, like, the dog bit. But there might be a lot of interference. So, like, it might not be the case that when the dog isn't present in the prompt, it's at 0. It might be between minus 2 and 2 and, like, just very noisy. And, you it's quite hard to solve this problem with a ReLU because what you want mathematically, what you want to do is you want to measure the distance from the origin if it's between 3 and 5, and otherwise set it to 0. But relus say everything above the ReLU threshold, is not set to 0. So you, like, can't have the ReLU threshold at 0. But if you have it higher, like 3, that means you're instead measuring the distance to 3, which is like distorting things and making it the wrong scale. And so an activation function which does do this is called the jump relu. So the way a jump relu works is it's basically like a normal relu, but you add a threshold t, that's positive, and everything below t is just set to 0. So for people watching the video, the graph basically looks like straight line, jump, discontinuous jump, and then it's the identity. And this can exactly let you represent things like if it's if it's above 3, take the distance from 0. If it's below 3, set it to 0. And so my team wrote a paper on gated sparse quantum codes led by Sen Rajamana Haran, who is great at just having wild crazy ideas for new SEE architectures. And the idea here was let's train a gate. Let's have, 2 encoders, 1 which figures out which features should fire, producing a binary mask, and the other that figures out how much they should fire. And this, and we only apply l 1 to the which features should fire. But because it's a binary mask, you can't have shrinkage. And we did some magic tricks to, like, make it so you could train a thing even though it produced a binary mask. It was quite hacky. And, but we also found that if you made the 2 encoders have the same matrix, just different biases, like the same weight matrix, it basically performed as well and, like, you had fewer parameters, so life was better. And it turns out that this mathematically reduces to a jump ReLU. And we had a, sequel paper called jump relu essays, which, are maybe the state of the art recipe, though it's a little bit unclear. Again, led by Sen. So the idea here is basically normal SAE, but you replace the activa-

tions with jump relos. A problem with jump relos is that you've got this threshold variable, But so if the threshold is 3, then everything above 3 is the identity. So if you got an input where the activation is 4, the threshold kind of has 0 gradients. If you make it a bit higher or smaller, it doesn't change things, which means it's based really hard to train. So what we instead did is we optimized the l naught rather than the l 1 proxy. By default, you can't do this because l naught is discrete. Everything is 0 or 1. It's just like, did this feature fire? If yes, add 1. If not, add 0. But we used straight through estimators, which basically means rather than thinking of it as a function of activation strength that is, like, 0, and then there's a sudden jump to 1, We think of it as a linear function that's 0 for a while and then has a kind of very steep diagonal line up to 1 and then is 1. And then you take the gradients of this, but only on the backwards pass. So it's, discontinuous when you're going forwards. You intervene when you're going backwards to make it this kind of weirder estimator thing, and we found that this performed really well. And it addresses both shrinkage and the other kind of messier problem of how do you deal with low value interference. The other architecture worth knowing about is top k SAEs from Leo Gao at OpenAI, who wrote a great paper on this. The other part of that paper was scaling SAEs to g p d 4, the absolute madman. And super simple idea, you rather than having ReLU's at all, though you can also have ReLU's, you apply a top k function. So you just take the top, say, 100 latents, keep those, everything else is 0. And this just gets you sparsity for free with, no concerns over like, does it have too many features firing? And this also works pretty well. We found that Jump Relu slightly outperformed it. Anthropic found that top k seemed better. It's a bit unclear. Both seem useful. Top k makes it very easy to just set whatever sparsity you want. This is actually quite an annoying problem. So, so we've got this sparsity penalty when training these things, whether l 1, l naught, and choosing this sparsity will change how sparse the model is, how many features tend to fire. It's kind of unclear what the right sparsity is because the intuitively, there shouldn't be that many, concepts relevant to a given input. So you shouldn't need that many. Maybe, like, 50 to a 100, which is what we typically target. But maybe it's more like 20. We don't really know. And you and often, the kind of slowest activating features will kind of just be noise, but it's often noise that helps the model reconstruct the input, but in kind of an uninterruptible way Because it's just saying, well, if the projection onto this vector is high, I will add a similar vector, and then I will reconstruct it. Yay. But you often want a good reconstruction. And so it's kind of unclear what the right way to do is. Typically, the way we compare SAE families is rather than just saying which 1 performs better, we take a range of sparsity penalties or different k's in the case of top k, and we plot a Pareto curve of, like, how many features are firing, l naught, and how good is it at reconstructing. This can either be how good is it at reconstructing the activation, or if you substitute the reconstructed activation and then finish running the model, what's the decrease

in cross entropy loss, which is, in some sense, closer to what we care about, because there might just be some uninterpretable garbage that doesn't matter in the activation. And, typically, this will look like a curve because having more features is better. And then we can compare the curves for the different methods. And so when I say a technique is better, what I mean is the curve is, like, up and to the right. And you should really go look at the papers and look at the diagrams. I'm not explaining this very well verbally. And, the other thing that's crucial to check is that you haven't accidentally made things uninterpretable. And you basically just do this by having a human interpretability study or having a, like, language model do it for you. And an interesting thing is that there's been quite a lot of progress in improving at the sparse reconstruction side of this, but interpretability numbers really haven't changed that much. And, that's odd. A final note on batch on top k is there's a small improvement called batch top k that Bart Busman, 1 of my mentees, made. So an annoying thing about top k is that you so you get to choose the sparse team, which is great. You don't have to, like, tune a hyperparameter and fiddle around. But what this means is that, you have exactly the same number of features per input, which, like, isn't really what you want, like, intuitively. Some tokens are boring. Some tokens are interesting and should have a lot more stuff happening. The idea with batch top k is rather than taking the top k for each token, you take the top b times k over the batch, where b is, like, the total number of tokens in the batch. And this means you can have variable numbers per input, but on average, the sparsity will always be a 100. And at inference time, you can just take, like, a kind of typical value of the b k th thing as a threshold and just fix that so you don't have to have a batch every time, which, is actually, identical to jump relu at inference, which is kinda cute. It's, like, nice connections. Though, jump relu

Feature Learning and Width Parameters

Neel Nanda 02:33:20

with identical thresholds everywhere. And batch top k is, like, probably my recommendation for people who wanna use top k stuff or just, like, really wanna know the sparsity. Jumper lume might be a bit better, and it's what we use for Gemmascope, but I think it's, like, probably more complicated to train and easier to shoot yourself in the foot. Though we have some open source implementations now.

Tim Scarfe 02:33:45

You said that we wanna have as as many features as possible, and and there seems to be something limiting the amount of features produced even across some of these methods. I mean, what what what are the things limiting the number of features? You know, useful interpretable feature.

Neel Nanda 02:33:58

So I don't know if I'd agree with that we want as many features as possible. So

Tim Scarfe 02:34:08

What do we want? What's the objective? Is it as many useful interpretable features as possible?

Neel Nanda 02:34:15

I want to know what's going on in the model. What is the computation happening? What are the variables? And I don't know how many variables there are. It actually is probably more complicated than that. Like, it's I've actually updated away from the idea. There's some fixed number of variables.

Tim Scarfe 02:34:36

So what can we do with autoencoders, and what evidence do we have that they work?

Neel Nanda 02:34:41

Yeah. So okay. So from the let's think about this from the perspective of a researcher. You've got a language model. You've got a sparse autoencoder. What can you do with that? So, typically, so you need to pick an activation to train your sparse autoencoder on. Like, you need to train a different 1 for, like, each activation in the model, essentially. Typically, the most interesting 1 is the residual string because that's like a bottleneck that's the sum of all layer output so far. So it's like the kind of if you understand what's going on there, you've actually answered quite a lot about the computation, while any given layer is only a small fraction. So, for example, in scaling monosmanticity, Anthropic just trained a few SAEs on, like, the middle residual. Yeah. So what can you do with 1 of those? So, the kind of natural thing to do is you just use it as a microscope. You run, put some text with the model, and on each token, you see which latents light up. And then, ideally, you have some tooling like Neuronpedia that will let you understand what those latents mean. And you can be like, ah, the model is thinking about this, this, and this, or why is the model thinking about this? That's odd. I'm curious what's up with that. I highly recommend people just go to the, Neuronpedia Gemescope demo and just play with the microscope part where you can just put in text and see what lights up, and it's got a very, very pretty UI. NeuronPDR also has an API, which means that when you're, like, a researcher messing around things in a collab, you can just see the dashboard for each feature, which is delightful because it's kind of a pain making them yourself. And yeah. So, like, using it as a microscope. Another thing you can do is just kind of using it as a discovery tool. Like, you just look at each latent, like, randomly choose some, and you're like, what does this fire on? Oh, that's an interesting concept. I wouldn't have guessed the model has that concept. Or maybe you guess for a latent you think exists. So you come up with a bunch of prompts that might have it, and you try looking at that and see if you can find a latent that kind of acts like a probe distinguishing those 2 datasets. The next thing you can try doing is causal interventions. So, basically, steering. You can either do this by clamping. So you pass the residual stream through the SAE, but then you take the latent, like, say, the dog latent or the yelling latent, and you make it put it at a fixed high value, and then go proceed. You could also just do steering. Just use the decoder vector as a steering vector. You can think of this as just adding some number to the latent activation, or you can think of this as just, man, just adding a vector. You don't even need the SE. 1 note is so SAEs introduce error. The reconstruction is not the same as the original activation. And, these error terms, the, like, difference between the original residual stream and the, like, new 1, the residual residual stream as it were, is sometimes contains important things. Like, there's some features that were just too niche to be captured. And, so if you replace with the reconstruction, this in and of itself will have an effect on the model, which might

drown out the effect of clamping a latent. Typically, what we do is we just add the error term back in or do some more efficient operation that is mathematically equivalent because that's kind of wasteful. And this is annoying because it means there's this, like, mysterious node that we don't understand, but it also means we can do cleaner investigations. So pros and cons. You can also yeah. So, yeah, you can use it to evaluate what's going on causal interventions. Ablations are probably a more interesting kind of causal intervention. It's like you've given a prompt and some concepts light up. Like, the Eiffel Tower is in the city of, and it says Paris. And you're like, which latents actually matter for it saying Paris? You can just delete them 1 at a time and see how the Paris log probability changes. And, yeah. So I think a more ambitious thing is circuit finding with them. So, yeah, so sparse autoencoders, by default, they're just an activation studying technique. They find the variables in the model, but this doesn't inherently give you the active the, like, algorithms, the circuits. But what we can do so, yeah, I think the best work here so far is the sparse feature circuits paper by Sam Marks and Aaron Miller. What they basically did is they took the outputs of each attention and MLP layer and every residual stream. They train a different SAE on each of those. They did this on pithier 70,000,000 because it's kind of expensive and annoying, especially because you have so many latents when you have, like, this everywhere. And you try to just do, like, causal interventions. You've got some prompts. They studied things like subject verb agreement, like, the yeah. Like, the man says, the men say or something, and tried to find interesting features for this where you basically just look at the causal effect of each feature or latent when you get rid of it, and then you also can look at connections between them. So, like, you delete a latent in layer 2. You look at the change for a latent in layer 4, or maybe you subtract the direction for the latent in layer 2 from the input to the 1 in layer 4, and then you try to use that, and then you see what effect that has. And, 1 nice thing about sparse autoencoders is because they're sparse, often many latents won't matter. And so you don't have to do this edge tracing for, like, everything because that would be a combinatorial nightmare. Also, because they had residual SAEs, this meant that they kind of only had to do, like, 1 step of a path, because there was, like, a bottleneck at each layer they were connecting to. Like, they weren't looking at connections from, like, attention 1 to MLP 5 or something. A problem with this approach is that, the error nodes will just completely break your model. Oh, sorry. The reconstruction error, when you've got an SE at every layer will kind of cascade and just destroy your model, which means that circuit finding, without them is, like, kinda sketchy. But if you include error nodes in your circuit, then it's like you just got this massive uninterruptible lump, and, they got much worse results, when they, like, didn't include error terms. You can still, yeah, you can still make some progress. Like, you can say, if I have a subgraph of latents and I think it's a circuit, you could try deleting all of them and see how much that damages model performance, for

example. And, like, that can give you some evidence. I'll also say that I think you don't even necessarily need to do the kind of edge patching stuff because often you'll have a sufficiently small set of latents. You can just kind of assume all of the edges are meaningful, and it's, like, kinda good enough. 1 caveat I'll give to this is that this is very much kind of causal intervention based circuit finding. They, in fact, used, a gradient based approximation to this called attribution patching that I think I discussed briefly earlier, which they improved using integrated gradients, so, like, classic interpretability technique. But it's basically trying to approximate causal intervention of, like, changing a latent value to its value on another input. 1 kind of nice thing is that there's kind of a diff good default value for latents of 0. Finding a good default value is actually quite hard for, like, general circuit finding because activations aren't normally, like, the 0 vector. So if you set it there, that's actually really destructive and weird. You could set it to the mean, but maybe the mean has much smaller norm and that messes with layer norm and things like that. You can also do activation patching when you replace the latent's value with its value on another input. This is useful if the latent is kind of active most of the time, though it can actually be misleading to do mean ablation when you replace it with the average. Because if the distribution of a latent is at 098% of the time, then it's a 102% of the time, then the mean is 2. But, like, in some sense, being 2 a 100% of the time is, like, less faithful to the original distribution than being 0 a 100% of the time. So I don't know. There's, like, lots of questions we still haven't figured out here. 1 of the things we went out of our way to do with Demascope is to have SAEs on, like, every layer and sublayer to, like, enable this kind of analysis. And I think you can just about fit all of the narrow ones for Gemma 2 2 b on, like, an 80 gig a 100, maybe leaving out the residual ones, because I'm excited to see what circuit analysis can find on those kinds of models. I actually have a pair of, math scholars, Stepan Chabalin and Dmitry Karlipenko, who are gonna produce some work on this stuff soon, who've been looking into in context learning. And, so there was a recent, set of cool papers on these idea of function vectors or task vectors. There are actually 1 from Eric Todd and 1 from, I think, Roy Handel, like, a day apart. The idea for this is in context learning, or, like, few shot learning, you can have a bunch of examples of a task, and then you, give it another input, and it does the task, like producing opposites, tall to short. What they found is that at a certain layer, you could extract a vector that corresponded to the task. And then for a different prompt with, like, a different input, and a different task, you could swap out the task vector, and it would do the original task for the input in that prompt. So kind of isolating the few shotness and also finding that it was at, like, a slightly earlier layer than when the model figured out what the input was, which is quite cute. And so what Stepan and Dimitri found, which I think we put out in a recent blog post, is that, there are sparse autoencoder features corresponding to these task vectors and also sparse autoencoder features corresponding to, like, task detec-

tors, like, task is happening on earlier tokens that, like, connect up to them. And, yeah, I am really curious to see how much more you can uncover of those.

Tim Scarfe 02:47:45

Is it possible to know, though, in advance I mean I mean, how should we expect these features to arise? I mean, if we have a certain type of, you know, dataset about, I don't know, architecture or something, you know, should we expect a certain type of feature to arrive?

Scaling Laws and Training Stability

Neel Nanda 02:48:02

Yeah. So this is a really interesting question that we don't understand super well yet. So yeah. So I think a crucial, hyperparameter I haven't spoken about so far is the width of your dictionary, the number of latents. This is chosen by the researcher before they train the SPARS autoencoder. And you can kind of think of this intuitively as the zoom level of your microscope. And, like, do you find kind of a few coarse grained features or lots of fine grained ones? And, a really interesting thing is that so the kind of picture I laid out at the start of there are just some ground truth list of features which have their own vector. Under that view, you might expect a sparse autoencoder to kind of learn the first 20,000 of those if it's got 20,000 latents and just totally neglect the rest. But what we find is that that's partially true. Larger sparse autoencoders do learn features that seem totally missed. 2 of my MAT scholars, Bart Bissman and Patrick Liesk, have a fun blog post on that. But there's also lots of feature splitting, where you kind of have a coarse, higher level feature that splits into, like, lots of subthings. Like, I don't know, color might split into, like, red, green, blue, purple, etcetera. That's actually sufficiently common. That probably doesn't actually happen, but there are, like, lots of examples. And this is kind of weird because it raises the question of, are sparse autoencoder features actually, like, units of computation or not? And my current guess is that they are, like, probably not. Barton Patrick had another post called, like, are they atomic? And what they did here is they trained a meta where you take SAE latent features sorry. You take the decoder matrix of an SAE, and you train another SAE to reconstruct that decoder matrix called meta typically with, like, a tiny l naught, like, or something. And this found that sometimes you'd have features that like an Albert Einstein feature that decomposed into, like, male physicist, German, whatever. And, these often these kind of meta features often corresponded to things in a smaller SAE. And so it's like smaller SAEs often split apart in bigger SAEs, but bigger SAEs are often kind of just composi-

tions of things in smaller SAEs. But they're also often interpretable and causally meaningful, and these feel like you know? I like interpretable causally meaningful things. That feels like it should be enough. And I don't know. I think this is a thing that I do not feel like I currently yet understand. I think the right question is maybe just are sparse autoencoders useful for understanding what's going on inside models? But it also seems important to have better foundations of, like, what have they actually found? Or I feel like they found something, but I don't know what they found. And it seems kind of fascinating that the internals of a model are kind of hierarchical in this way, where you can have interpretable, causally meaningful directions at different levels of abstraction. And, yeah, the other comment I'll make on what you might expect to come into this is so my favorite part of Anthropic's, scaling monospanticity paper is the feature completeness section. So what they did here is they took a bunch of things that you might expect to arise in a model, like chemical elements or, I think maybe London boroughs or something, and, where there's kind of, like, a clear name. So you can just check. Is there, like, a feature that lights up on text including this name? And they took these SEEs of different widths, and they looked at how what is the probability, or rather, like, how many of these get learned and, like, what does getting learned look like as a function of the frequency of occurring in these different SEEs of different widths. And they found that you could fit quite well a, like, simple predictive model that's I think it was like a sigmoid curve. I don't remember the details. But, basically, sufficiently small SAEs don't care. And then depending on the frequency, there'll be a point where, like, the probability starts increasing, and then it goes up a lot, and then it's, like, basically a 100%. Though, plausibly, it also splits later on, which is, like, a complexity I'm not sure they went into.

Tim Scarfe 02:53:27

But catch that reference. What was it?

Neel Nanda 02:53:30

So this is the feature completeness section of anthropic scaling monosmanticity paper. It's a long paper. It has so many sections. So good. And so kind of I think the takeaways from this are that, a, kind of a model, kind of the frequency of a feature matters a lot, which just makes sense. Like, you wanna things are more useful if they occur more often and if they are kind of, like, bigger in the activation space in some sense. But I think the second interesting takeaway is that it seems probabilistic whether a feature gets learned in an SAE of a given width. And, like, if you train multiple of the same width, they'll have shared features that were kind of the obvious ones to learn. But for features that are kind of more niche and kind of more on the borderline, it's like, you know, just kind of random. And I don't really know what the takeaway here is. Like, I think I'd love to see more research in this area. In terms of advice for practitioners, I basically think that you should just train several widths of SAEs. In Gemascope, we have kind of a wide and narrow for every site, And we also have, a few layers where we have, like, powers of 2, so you can do, like, lots of hierarchical and feature splitting analysis. But, yeah, just have a few wits. Try the different ones on your tasks. See which ones capture the features you care about. It'll be nice if we had a better answer, but that's the best I got.

Tim Scarfe 02:55:07

Well, you can also train s SAEs on, I don't know, like, the residual stream on on just MLPs on on attention. I mean, how how do they compare?

Neel Nanda 02:55:17

Yeah. So, yeah, you can basically train it on, like, any activation. You can even train it on, like, the queries and keys and values and attention layer, though that's more confusing, and that's worked less well. Yeah. Doing it on the output of a layer, I think, is a super reasonable thing to do. The there's an interesting question of whether you should do it on the MLP activations or the MLP output. So, typically, the MLP activations will be, like, 4 or more times the size of the residual stream, but then there's just a linear map mapping it down to the output. And so, like, kind of conceptually, there shouldn't really be a difference. But also your SAE will have 4 times as many premises. And, like, maybe this helps. Maybe this doesn't. I don't know. Typically, I just train out on the output because it's 4 times cheaper, but it would be great if someone actually studied this systematically. Returning to your question, I think that these just have different tools. Like, if you want to study what a layer is doing, you want the output for that layer. I think a totally valid mode of analysis would be, taking a task you care about, like in context learning or refusal, doing activation patching, like swapping layer

outputs between different prompts to find the ones that matter. And then you could, go and take the Genascope SAE for that layer and then see what happens. The residual stream is generally more useful if you want a kind of holistic view of what's happening. Probably all kinds will be kind of causally meaningful, but, the residual stream is gonna be the 1 that, like, detects what the model is thinking about, while the layer ones are more like, what's happening at this layer? Like, what does that layer do? It's just, like, a different question. I think that m l an interesting thing about MLPs is that they the, like, MLP layer itself is this kind of dense nonlinear mess, and it's kind of in superposition. So you've got, like, thousands of jelly activations or gated neurons or whatever, and it's just very unclear how to think about this. This is fine if you're just doing causal interventions or you just want to study an activation. But if you want to try to do some kind of weights based circuit analysis, like recovering this hope of, like, input independent things, which feels like SAEs should bring us closer to you because they are kind of monosemantic, You just kind of can't deal with the big chunky MLP bits. The solution to this is transcoders. So I supervised this delightful paper from, Jacob Donavsky and Filip Chelinsky on them. The idea of a transcoder is rather than finding a sparse reconstruction of the MLP output with from the MLP output, you use the MLP input to find a sparse reconstruction of the MLP output. So it's kind of like a sparse replacement MLP layer with lots of neurons, but they're sparser and relus. And this you can kind of use this as a drop in replacement. On GPD 2, they worked about as well as MLP output SAEs in terms of reconstruction. On Gemma 2, they were notably worse, I think, because Gemma 2 has gated MLPs. Though, we still released a suite of transcoders, which might be useful for people who wanna do this kind of input independent circuit analysis. Though a quite annoying thing we found is that it was really hard to do purely input independent circuit analysis because you take 2 transcoders, and you multiply the decoder weights for 1 and the encoder weights for the other to get these, like, yeah, a kind of, like, interpretable basis by interpretable basis matrix. And there's lots of weights that just don't matter, like I was kind of gesturing at earlier. And, the way Jacob and Philip solved this is that they, yeah, they looked at feature co occurrence, and they just kind of didn't look at the connections for features that didn't co occur. But, like, that's somewhat input dependent, so it's not fully satisfying. But it was, like, solid attempt, and I'd love to see more iteration there. On attention. So attention is, much more linear than MLPs in some sense. So the attention pattern computation is weird and nonlinear, and I don't think we yet understand how to deal with it with SAEs. There was a interesting blog post from Keith Winrow on QK transcoders, which seems like a cool approach here. But if you take the attention pattern as a given, a slightly sketchy thing to do, but, bear with me, then it's basically just a linear map. You use the attention pattern to kind of mix together the different passed tokens, and then you apply a linear map, the, value matrix and the output matrix, and you get a new thing. And you

do this for each head in parallel, and then you add them together. And so, technically, you do the value map, and then you do the attention mixing, but they commute, so it's fine. And I think see the last episode if you want me rambling about this in much more detail. And so this is interesting because it means you can do linear attribution. You can, just say, look. This is basically a linear map of past tokens and then some feature lit up. So, like, what were the tokens that contributed to this? And, you can even rather than training it on the output, train it on the, concatenated mixed values over the heads, which is, like, the thing immediately before the output weights. This is just like a linear map away from the actual output. And this means that you can now attribute things directly to, like, each head, which is quite a nice feature. And I, supervised this paper from Conor Kossain and Rob Kotianowski, basically just trying to do a deep dive of, like, can you do attention can you do SAEs on attention layers? And, basically, yes. Works great. They're potentially a bit more interpretable than MLPs, and you can do all kinds of nice things with, like, linear attribution like this. For example, Rob, the absolute madman, went through every head in g p d 2 small and looked at the top 10 SAE latents according to, like, how aligned they were with that head specifically, because you can just read off the weights for each head and look for a pattern and just kind of labeled heads by, like, do I see a pattern? What is the pattern? And check out the paper if you want. I think we've got a table of all of them somewhere.

Feature Identification and Bias Correction

Tim Scarfe 03:02:46

Very cool. And what's the relationship between the number of features that that that you try and find and how kind of meaningful the features are? And I was also wondering, like, how how is it related to the size of the model?

Neel Nanda 03:02:59

Yeah. So this is not a thing that I think has been studied in much detail. I predict that the interpretability will be quite similar, though the larger though there's often kind of training stability problems with larger SEs. Like, when you get beyond, like, 1000000 latents, things get more annoying. For example, in Anthropix 34,000,000 latent SAE, they, I think, had about 65% of the features were dead, meaning they just didn't fire. And this is, like, a really high ratio. It's just, like, 20,000,000 features doing nothing. And this might just be because there aren't 34,000,000 things, though. They definitely found that their SAE had missed out on some stuff. EG, the model knew all of the London boroughs, but

they couldn't find SAE features for all of them, only most of them, for example. Yeah. So learning stability problems. But, like, beyond that, my guess is they get much more fine grained to the point where you could argue that some of them are just, I don't know, like, memorizing a couple of training examples or something, perhaps, if you go to the extreme. I mean, in the infinite width limit, you'd kind of expect each token to to get its own latent or something, which seems boring and degenerate, but I'd be surprised if we ever get to that point. Intuitively, larger models should just have more concepts they know, meaning you should want a larger thing. I don't know if anyone's actually tried training an enormous SAE on, I don't know, every model in the g p d 2 family or Pythia family and just tried to see whether the smaller ones look kinda crazy and the larger ones look reasonable. But, like, I think that would be a pretty cool experiment.

Tim Scarfe 03:05:03

On on the security side of things, is it possible that these, you know, sparse autoencoder features could be used to, I don't know, perform adversarial attacks or something like that?

Neel Nanda 03:05:15

Probably. So I think my general vibe is we haven't seen any much evidence that SAEs can do stuff that's, like, beyond what you can achieve with, say, fine tuning. Like, fine tuning can do a lot of adversarial attacks against a model. And it seems as plausible that you can do kind of interestingly different ones with interpretability, but I, I haven't seen any real signs so far that there's kind of dramatic capabilities that were just not there that, like, were really hard to achieve with fine tuning. So, for example, in this, refusal is mediated by a single direction paper, that I did with Andy Alditti, where we found a, like, refusal steering vector that we could use to jailbreak models. You can jailbreak models by fine tuning the weights, and this was, like, a large part of why I felt comfortable releasing that paper. Like, it was just an interestingly different attack and, in some ways, kind of easier and cheaper. And, like, maybe things like that might happen with SAEs. And, I mean, it's a new technique. So, like, there's always the potential that there's some things I'm not expecting that we're gonna learn, but I don't see any particular reason to expect it to advance the frontier of what we can do beyond things that use throw compute at the problem, because throwing compute at the problem is really effective.

Tim Scarfe 03:06:51

Are there any, scaling laws with s SAEs?

Neel Nanda 03:06:55

Yeah. So this is a thing that kinda surprised me. The yeah. Both, Anthropic and OpenAI have had recent papers just scaling these to much bigger models. Like, we've talking about Anthropics a bunch. OpenAI scaled it to d p d 4. And, like, both have some nice sections in their paper on SAE scaling laws. And it seems like I don't know. Initially, we didn't really know what we were doing when training SAEs, but now we've iterated on the methods enough that they're, like, smoother. And we can just do things like change the amount of data, change the amount of latents, things like that, and plot out, I don't know, what an optimal optimal

Training Dynamics Analysis •

Neel Nanda 03:07:40

set of hyperparameters for a given compute budget would be or something. And, like, it's kind of messy because there's a lot of hyperparameters. There's, like, how much data? How many latents? What sparsity penalty should you use? And, like, there's not really canonical solutions to any of these, but people should totally go check out those bits and stare at the pretty curves.

Tim Scarfe 03:08:03

Yeah. I mean, 1 of the problems is just the amount of garbage features. I mean, what do we what do we do with them?

Neel Nanda 03:08:09

Yeah. So okay. So there's, like, dead features, which basically don't fire. There's uninterpretable features, which fire sometimes, but we can't really see a pattern. And then there's, high frequency uninterpretable features, which fire a lot, like, 1% of the time plus. And dead features, I don't know. Kind of ignore them. They don't really matter. You can print them out of your SAE, so I think the rates of dead features have gone down a lot as we've kind of refined our training techniques. Like, previously, there were lots of hacky things people did, like resampling, where Ares Warfarin just replace all of the dead neurons with new weights. But I think training methods now are quite a bit more stable, and that doesn't seem necessary anymore for the most part, with more modern methods. I think that the uninterpretable ones, I don't really know. There's an argument for just removing them. I'd be kind of curious to see what happens if you just do auto interp on every feature. You find the ones that are bad. Like, you delete those,

and then you, I don't know, fine tune the SEE a bit so it doesn't have, like, a gaping hole in it. That seems like kind of an interesting direction. At the moment, I don't really think about them too hard. Though if I started to see, like, lots of uninterruptible things lighting up a bunch, I'd start to be more concerned. I think the thing which is more concerning and I'm, like, a lot more interested in understanding is these high frequency features. So under l 1, things which light up, like, 1% of the time are, like, very heavily penalized. So they don't tend to happen too much. But under things like Top K or Jumpreliu, they're penalized much less. Like, top k doesn't really care. And with jumpreliu, you only get penalized if you're kind of near 0. So if it's just big 1% of the time, maybe it doesn't care. And this means and if it activates a lot of the time, it can be quite useful for yeah. If it activates a lot of the time, it can be quite useful for reconstruction even if it's not interpretable. And sometimes these things do seem interpretable. It's kind of confusing. Like, Anthropic had a recent monthly update where they compared different sparse autoencoder training techniques and, like, mentioned they'd looked into this a bit and had been like, actually, we think these things are kind of interpretable enough. We're not that worried about this. Is the rough vibe I remember? Though people should read the post. I might be summarizing it incorrectly, as goes with every paper I discuss in this podcast. But yeah. So

Tim Scarfe 03:11:01

what was that paper you just referenced from Anthropic?

Neel Nanda 03:11:03

It was 1 of their monthly updates. I think the 1 the second most recent 1.

Tim Scarfe 03:11:11

I mean, 1 thing I I was thinking about though is that, you know, 1 school of thought, and I'm not sure whether you lean in this direction, is that these things represent units of computation. Another school of thought is that they are post hoc descriptions, and what we need is, I guess, some kind of a causal mechanism to tease that out. I mean, what what do you think about that?

Neel Nanda 03:11:31

So I don't know what unit of computation means. This is kind of what I was getting at earlier with the feature splitting and these things are not atomic discussion. Like yeah. Like, in some sense, I feel like a unit of computation must be irreducible, and it must kind of fit into a kind of clear circuit and be used downstream. But if you have 3, if you've got a mammal feature that splits into a bunch of animal features but is causally meaningful in its own right, is that a unit of computation? Like, maybe? Or, like, in some sense, it's enough to be useful, but it doesn't it's not quite achieving the ground truth of what's going on with the model. And, like, I don't even know whether the stuff bottoms out somewhere in some underlying truth or if it's all just some kind of messy hierarchical structure where you can kind of go up or down the like, it's not even a tree because things often merge. Like, directed acyclic graph, And yeah. So I think a research direction that I'm pretty excited about is trying to apply these things to real world tasks or at least kind of downstream tasks. For example, in Sam Mox's, Sparse Feature Circuits paper, in addition to the circuit analysis stuff, there was a really cool bit where called SHIFT, where they, so they took a gender biased dataset. Like, I think it had bios of, female nurses and male professors, and they trained a

Scale and Infrastructure Requirements

Neel Nanda 03:13:20

probe on that. And it partially learned to pick up on the profession, and it partially learned to pick up on the gender. So when you evaluate it on a dataset with the opposite gender flip, it does really badly. What they found is they could identify, the SAE features that were most relevant to gender with a mix of giving the model data and human analysis, get rid of those, and it the probe became much less biased. And I think this was, like, a really cool application of I don't know. I don't know what our existing probe debugging baselines should be, but seems pretty impressive that SEs can do something like that. And I'm like, if it can do something like that, how much do I care whether it's a unit of computation or a post hoc thing? Like, I care a lot about it being real, and I care about it, like, teaching me things about the model. And I don't want to assume things about it that will mislead me, but

Tim Scarfe 03:14:20

Is it possible for us to correct knowledge or to insert knowledge into a model using SAEs?

Neel Nanda 03:14:29

Maybe. So a problem with SAEs that I think is often elided in this kind of high level discussion is that the thing you want is not always a feature. Like, I've trained SAEs and tried to make Golden Gate Claude, and I just didn't find a Golden Gate feature. And that was yeah. Like, maybe if I trained a wide enough SAE, I would have. I don't know. And, there's actually a benefit of steering vectors because this isn't really an issue there. You can just always make a dataset for the thing you want. And so, like, if the knowledge you want is represented as an SAE feature, you could totally delete it. That would probably somewhat help. I yeah. I know of some people who are looking into this, though it sounds like it doesn't outperform existing unlearning baselines at the moment. And I think that yeah. I don't know. Knowledge in session, maybe? So this might work. Yeah. This might work. It so it's it feels a bit odd because to me, knowledge is like a kind of lookup. It's like a function. Even if you wanna do it in, like, a really nice, interpretable way, you kind of wanna have the input that's like Michael Jordan and the output that's like plays basketball. And you can maybe add a little bit of Michael Jordan to the, input direction for basketball to the encoder direction for basketball. You could also just add an extra latent that has Michael Jordan and produces basketball. This is kind of against the spirit of an SAE because, like, that's clearly not reconstruction y. That's just, like, an additional function you've added. But, like, in some sense, you can do this. It feels more intuitive with transcoders, the, like, MLP inputs to MLP output thing, because in some sense, factor lookup is the job of MLPs. Though, I mean, we found that it tends to be distributed across layers, which is kind of annoying, but you can insert it at 1 layer totally fine, especially if you're down to just have, like, a fairly high magnitude decoder vector or encoder vector.

Tim Scarfe 03:17:01

What's the relationship between, SAEs and the task that's being performed or the amount of data trained on?

Neel Nanda 03:17:08

So SAEs get better when trained on more data as a general rule. Sometimes they'll kind of saturate. I'm not entirely sure if this is, like, true saturation or just a problem with the training methods. But in general, more data will make them better at reconstruction. We haven't looked too hard into, I don't know, a new features appearing, or is it just refining the existing ones? I think this would be, like, a pretty cool thing to just look into. Like, take a bunch of text that activates an SEE latent. Look at that latent over training. Plot some graphs. Do this for a bunch. Does it look like phase transition y, or is it gradual? I have no idea. Someone should look into this. I think that you I expect you can train specialized SAEs. I not I'm not aware of too much work on this. There's a paper called Towards More Principled Evaluations as Sparse Autoencoders I supervised from Alex Makilov and Gail Gleng. The focus of this was we think that SEEs should be useful for circuit finding. Let's take the circuit IOI, indirect object identification, which, we talked about a bit in the last episode in g p d 2, and let's see how well SEEs find it, and compare this to kind of supervised dictionaries that we just train because we kind of think we know what the features should be. And, I think that was a pretty cool evaluation. The relevance to this is they both trained SEEs on, like, web text, but they also trained it on just IOI specific stuff. And to some degree, it performed better, but it also was kind of weird in various ways. Like, there was this phenomena of feature oversplitting where there was some feature that should have been there. Like, is the name first or second? And instead, it's split into, like, 20 features that were all kind of like that. And this was much more annoying to work with and analyze, than the kind of broader SAEs. But maybe you can solve this by making it narrower. But if you make it narrower, then it doesn't learn the name of features, which you also care about, which are kind of should be more fine grained. So yeah. But, like, that was a very narrow task. I'm sure if you did say Python code, it would work much more sensibly.

Tim Scarfe 03:19:52

I mean, minute ago, you were alluding to training dynamics, and this is something that that really interests me, you know, which to say, how do the the weights evolve over the course of of of training and how do the, you know, circuits emerge and so on. Could you not, you know, kind of compute SAEs at different points of the the training curve and use that to understand the training dynamics?

Neel Nanda 03:20:18

Yes. I think that is a thing that should work. I yeah. So yeah. Just to be clear, we're switching

Chain-of-Thought and Latent Reasoning

Neel Nanda 03:20:28

from training dynamics of SAEs to the training dynamics of the model and using SAEs as a tool to study this. And, yep, I think that I kinda think there's just lots of stuff you can do. I think this would be 1 example. SEs might even be overkill. Like, you could just take some concepts you believe are represented and just train probes for those. 1 thing I'd be quite interested in is, like, how does the probe evolution evolve over how does the probe direction evolve over training? Training dynamics isn't really my field, so I'm gonna say things, and there might just be a paper that just does this that I haven't come across. But, yeah, I think, looking at what the representations look like over time would be super cool. Looking at how well an SEE transfers would also be interesting, especially if you correct for, like, changing bias terms. Because, like, if the average changes, that's gonna screw up everything. But maybe the direction of a future doesn't change.

Tim Scarfe 03:21:34

Because, I mean, obviously, you've spoken a lot about grokking, and I guess that could be seen as looking at training dynamics and so on. But would it be interesting if you saw so called emergent materialization of of capabilities or or features that come out of an SAE? Would that be interesting to you?

Neel Nanda 03:21:55

So, I mean, I think emergence in general is an interesting topic. I would be a little bit hesitant to draw too strong a conclusion from what happens with SAEs because, for example, it could be the case that it gradually learns the feature for, say, pillow, but the SAE doesn't think it's worth its while to have a pillow latent until it reaches a certain threshold of salience and, like, then it learns it. And, like, that might look really sudden, but, actually, it's gradual. So, like, what I'd act so this make me kind of more interested in, probes, possibly using the SAE as, like, a motivator of what what things to search for and, like, what probes to look for, or just literally using an SAE feature as a probe and just translating it to the earlier ones. Because, like, it wouldn't surprise me if, the pillow latent on the first SEE that learns it still works on the checkpoint before. And yeah. I don't know. The there's, like, a couple of models that would be pretty good to study this on. I've got some toy language models in the transformer lens library, like Jell O 1 l or Jell O 2 l that are, like, 1 layer and 2 layer models that should be super easy to train SEEs on. There's the Pythia suite that has tons of checkpoints. Yeah. But, yeah, I'm hesitant to use presence of a feature in the SEE as indicator because there's

just, like, a lot of noise there. So I wanna be more cautious.

Tim Scarfe 03:23:36

Yeah. And I think that your team have just put out a whole bunch of open weight SAEs for yeah. I mean, what were the engineering challenges doing that?

Neel Nanda 03:23:48

Yeah. So this is kind of wild and slightly depressing from my perspective, in that, SAEs have kind of become quite a big deal in MechInterp, or at least I think they're a big deal. And I think they will unlock lots of cool things we can do. So I wanna focus on them, which involves training them. And this is just, like, much more of an involved engineering thing than lots of previous projects that were more like play around in a Colab notebook with a tiny cute model. And so the reason it's an engineering challenge is so, essentially, to train an SAE, you need to run a bunch of text through a language model, collect activations at a certain layer, and then for, say, a couple of billion tokens, and then use those activations, to train a sparse autoencoder, which can also be quite big. It it's, like, only a 2 layer network, but, like, the middle can be enormous. So, like, if you're going as big as Anthropic did to, say, 34,000,000 latents, and you want a trainer on a model like Gemma 2 27 b, which has a residual stream of width 5,000, the total number of parameters is 2 times 5,000 times 34,000,000, which is, like, 340,000,000,000 parameters, which is, like, a lot bigger than Gemma 2 27 b. We did not go that big. We only went up to, like, 1000000 latents, but still gets sizable. And so, you kind of have 2 choices for how you do this. You can either do this online. So you, pass text with the model, you store the latents in, like, your GPU RAM, and then you run an SEE on them. You often want to have a big buffer of these activations so you can shuffle them because activations from the same prompt, will kinda be correlated a lot in what features they have. Though no 1 has actually tested to my satisfaction how much this matters. I just intuitively expect it to matter. It's like so many things like that in essays. And, yeah, so this basically doesn't work for models beyond a certain size, and I mean that if you wanna do, like, a hyperparameter sweep, this is, like, quite expensive, because you need to run the model again and again. Or even just if you want to train a bunch of SEEs at once, on different activations. And so the second mode, which is what we eventually converged on, is activation storage. So you just run the model once. You collect all of its activations. Actually, I think we did, like, a couple of stages, because the, like, disk input output became a bottleneck

Career Path and Advice for Researchers

Neel Nanda 03:26:40

you wanted to store that many activations. You save these to disk, and then you train sparse autoencoders on them at your leisure with whatever hyperparameters you want. And if you mess up the training, you can just go back and train again. The so this is just kind of intense. Like, we stored about 20 pebbytes of activations, which is a lot. We had to request special permission to get that disk allocated to us. But, fortunately, DeepMind believes in sparse open coders. And we used about 22% of the compute of g p d 3, which is, like, not that much by modern standards, but, like, by interpretability standards is quite a lot. The main reason, all of these numbers are so big so we mostly released things on Gemma 2 2 b and Gemma 2 9 b because I think these are, like, a good intersection of, kind of useful to of, like, kind of big enough to be interesting, small enough that, like, academics can, in fact, work with them without breaking the bank for GPUs or needing to mess around a ton with, like, paralyzation, is that we wanted to do, like, every layer, including attention MLP and residual. And so, like, demo 2 9 b has, like, 42 layers, so this is, like, a 120. And we also wanted to do 2 widths, which is 240. And we couldn't decide on what the correct sparsity was. We did, like, 6, and so that's, like, a lot. I typically, claim we did about 400 because I'm not sure I should count the sparsity ones. Because I normally just recommend people use the ones closest to a 100 unless they actively want to do some kind of sparsity study, which people should do. I don't know how these things work, man. And, we also did, I don't know, a couple of other ones. Like, we occasionally did a 1,000,000 latent 1 on a few layers. We did a few on Gemma 2 27 b, and we did a few on the chat tuned Gemma 2 9 b. Though, actually, SEEs transfer surprisingly well between base models and chat models. 2 of my MAT scholars, Konica Sein and Rob Kuzianowski, have a nice blog post showing this, and we corroborated that in the Gemascope technical report. I also should say, credit for Gemascope goes to the team and most especially to Tom Liberum, who was the technical lead and dealt with all of this engineering nightmare, so I didn't have to. And yeah. And, like, this is just Gemascope. Like, I expect, if we wanted to do a larger model, and, like, larger SAEs, it would become kind of even more annoying. Though there's a distinction between, kind of training 1 SAE on a residual stream, which is enough for some use cases but not others. Like, it's enough to kind of monitor the system. It's enough to, like, steer it, but it's not enough to, like, see the internal circuitry. And doing every layer in sublayer made it, like, a 100 times more expensive. Maybe not quite a 100 times more expensive because you don't need to run the language model a 100 times as much, but, like, more

painful. And, so, yeah, it's unclear to me how much this is worth it, and I would be quite curious to just see people explore this kind of thing in practice. And, yeah, the reason we did Gemascope, this open release in the first place, is as what I've been saying hopefully communicates. This is really annoying engineering wise, and this is a lot easier to do if you're in an industry lab than if you're outside because you just have better infrastructure and more compute and, you know, 20 PepperBytes of storage. And, kind of in the same way that things like Gemma and Lama, like, they're really expensive to train, but they're to use, they're, like, much cheaper. We thought that we could unlock lots of cool research projects by doing this. There's also also some other cool open way SAE projects worth giving a shout out to, like, Elutha's have trained a bunch on Llama. And those probably actually counts as open source since I think their code and data is all open source. OpenAI released a bunch of high quality ones on g p d 2 small, which is nice for smaller projects. And there's a bunch of more scattered people who've put out various nice things. But the hope with Gemascope is that for, like, any project that wants a kind of interesting model, this could just be high quality canonical SEEs that would enable as many projects as possible. So we just kind of threw random variants in there.

Tim Scarfe 03:31:52

So so much stuff has been done, but there's still lots of work to do. I mean, what are the open problems in in SAUs still?

Neel Nanda 03:31:59

Yeah. So okay. So here's a couple of areas that I'm excited about. So I think just basic science. There's a lot we don't understand about SAEs. Like, I've I know I hope I've pointed to many mysteries so far. Like, when should you expect a feature to be in the SAE? What's the right sparsity? What's the right width? Are there true units of computation? 1 problem I might call out is just what's going on in, SE error terms? What do they mean? Can we understand them? And we know that they sometimes just capture features that the model missed, but, like, there's also weird stuff going on. Who knows? There's yeah. That's category 1. Category 2 would be using SAEs to solve mysteries of interpretability. So there's just, like, a lot of things I don't understand. Like, what exactly does fine tuning do to a model's circuits, especially chat fine tuning, which is, like, a particularly important kind? Can we can we find, like, a truly monosemantic circuit with SAEs now we've got kind of the right units? Are these the right units? Kind of unclear. But, yeah, more of a kind of a using them task. And also just, I don't know, applying them to interesting circuits, like, a few short learning. The next category would be real world tasks or downstream tasks. Like, are there things that people who aren't interpretability nerds think is in-

teresting? And can we use sparse autoencoders to make progress there? I I say this both because, I don't know, I think doing useful things, especially on the end of, like, making models safer is great. And I partially say this because I think it's just so, so easy to trick yourself in interpretability. And we have, like, some evidence of SAEs working. I'm like, get into that a bit more in a bit if you're curious, but I think that just there was a task that people struggled on, and SAEs were able to do this at least as well, ideally better as competing approaches or with, like, some advantages over competing approaches such that it might be useful in some settings. Would it be really awesome to see? For example, can we use them to make models hallucinate less? Can we use them to monitor models for, like, erratic behavior or whether they're being gel broken? Can we use them to make models more steerable, like, in ways are there scenarios where, like, prompt engineering fails? Can we debug models? Like, other times, Lama or Gemma have weird behaviors that we want to go and dig around and see if we can decode why this is happening? Like, 1 of my dreams is to be good enough at interpretability that if we have a moment like Sydney again, where it's just behaving in really wild and unexpected ways, like trying to convince a reporter to leave their wife, that we can actually understand what is the cognition happening inside that led to this.

Tim Scarfe 03:35:22

Yeah. I mean, earlier on, we were talking about that connect the dots paper and the unfaithful chain of thought. And, you know, potentially, you could apply essays to some of these problems. But could you just introduce those to the audience?

Neel Nanda 03:35:34

Yeah. So, yeah, I think I briefly talked about the unfaithful chain of thought. Key idea. Well, key motivation. People often see a model produce a chain of thought, and they're like, that is an explanation. Therefore, it is interpretable. I have understood it. Why would I ever need to do interpretability? And, like, sometimes this is a true, but, like, sometimes the model kind of already knows the answer and just produces some kind of spurious bullshit if it thinks it's supposed to give an explanation right now, and it's unclear how causally relevant this is to the answer. And so this paper had a really nice demonstration of this where they, made a bunch of multiple choice questions with example chains of thought. They gave the model a few shot prompts with 10 questions where the correct answer was a. The final 1, the correct answer was b, and the model tended to produce a spurious chain of thought justifying why the answer was a and then said a. And I'm like, this suggests me that the model has a circuit inside it that is like the generate bullshit chain of thought circuit. It has an internal representation of what the correct answer should be, and it somehow is mapping that to this circuit. And I'm like, that is wild. Can we turn this into some kind of, like, faithful chain of thought detector? Where, like, I don't know, you give it a bunch of maths problems, and you check how causally meaningful the chain of thought is, and you compare that to your detector and use that as a force of ground truth or something. That would be wild.

Tim Scarfe 03:37:07

So we were just talking about Owen Evans. Well, it was hit out of his group anyway, and he had this paper which went viral on Twitter a few months ago called connecting the dots, talking about this out of context generalization and and reasoning. And there's some really cool examples in that. But Kent, can you just fill us in on that?

Neel Nanda 03:37:25

Yeah. So, yeah, so Wines Group does the great genre of paper that's like you get an LLM and you take a thing that you would intuitively expect to work, and you're like, oh, that doesn't work or vice versa. And so the connect the dots paper, the so what they did is they I'll just discuss a concrete example. So they came up with a mystery city, say Paris. They call this city x, and they keep giving the model prompts of the form, the distance between city x and London is blah, where blah is the true distance, and they fine tune it, to predict the distance to a bunch of other cities. And then this is, like, kind of a weird thing to do. You might expect he would just, like, memorize these or something. But then at inference time, if you ask it, like, what is the name of city x? It says Paris. If you ask it questions about Paris, like, what's the biggest landmark? We'll say the Eiffel Tower. And they had several other tasks that, like, in this kind of genre of you fine tune it to do a task where kind of discovering some latent thing is a useful way to do the task, and then you ask a bunch of questions that show it generalizes in a way you might not expect. And I really like this result. And my hypothesis for what's going on is that the easiest way to answer these questions is not to memorize. It's to use the existing circuitry that has, I don't know, memorized, or maybe the model has some actual internal world model to compute distances, and finding the Paris direction does that. And so, like, there is gradient signal in the Paris direction, and this is enough to, like, overcome any additional pressure. And so the model just converts the, like, city x phrase into Paris. And I would love someone to, replicate 1 of their examples on, like, llama 3 7 b or I can't remember. Is that 9 8 b? And or, like, Gemma 2 9 b and use SAEs to try to dig into what's going on there. Because I would find it very cool if my random postulating with Mech and Terp intuitions, was correct.

Tim Scarfe 03:39:51

So people at home, you wanna learn more about SAEs. What should they do?

Neel Nanda 03:39:56

Yeah. So, yeah. So I think the Arena tutorials are a great place to start. They've just made a new 1 on sparse autoencoders, which we can link. And this will kind of give you a hands on walkthrough and, like, help you write code and, like, really engage with them. I think this is a great place to start. If you want a gentler introduction, the other best place to start is the Neuronpedia demo for Gemascope, where you can just kind of play with it in an interactive way, see what a sparse autoencoder can do, try staring with it, try using it as a microscope, and, no code required. Maybe I'd start with the NeuronPDR demo, and then I had to do the Arena tutorial. And then if you wanna dig deeper, my reading list has a section with SEE papers in it. And beyond that, I recommend just, like, jumping into projects. There's a bunch of SEEs on small models if you don't have much compute. If you know how to use larger models, you can, play with the various Gemma 2 SEEs in Gemascope. I think Gemma 2 2 b is, like, pretty manageable to play with. Just like, I don't know, test hypotheses about it. Like, does it have this feature? Can I get it to steer in this way? What things can I exhibit feature splitting? What things am I confused about? Just like, I'm a big believer in, like, as you learn about a field and as you read papers, just go and get your hands dirty and write a bunch of code and implement ideas in the paper or, like, predictions it makes or extensions. And I think this is pretty naturally flows into, like, actually doing research.

Tim Scarfe 03:41:46

Well, Neil, it's been amazing having you back on MLST. This has been an absolute marathon. And how long have we been recording for? A long time.

Neel Nanda 03:41:55

So I got here yeah. I got here at 2. It's now 10, 8 and a half hours.

Tim Scarfe 03:42:02

That's pretty good. That's pretty good. Yeah. Honestly, it's been amazing. I I think that this episode is probably the biggest brain dump out there on the Internet on sparse autoencoders. So we promised you that it would be dense and very rich of information. And I hope that we have not only delivered the goods, but we've, you know, gone past what we did last time.

Neel Nanda 03:42:27

Yeah. It's been great. Thanks so much.

Tim Scarfe 03:42:29

Wonderful. Thank you so much, Neil.

References

[1] **Neel Nanda - Personal Website**
https://www.neelnanda.io/
person

[2] **Sparse Autoencoders Find Highly Interpretable Features in Language Models**
https://arxiv.org/abs/2309.08600
paper

[3] **Progress Measures for Grokking via Mechanistic Interpretability**
https://arxiv.org/abs/2301.05217
paper

[4] **Representation Engineering / Activation Steering**
https://www.alignmentforum.org/posts/3ghj8EuKzwD3MQR5G/an-introduction-to-represe
paper

[5] **Golden Gate Claude**
https://www.anthropic.com/news/golden-gate-claude
website

[6] **Evidence of Learned Look-Ahead in Chess-Playing Neural Networks**
https://openreview.net/pdf?id=8zg9sO4ttV
paper

[7] **Chris Olah on Interpretability Research (80,000 Hours)**
https://80000hours.org/podcast/episodes/chris-olah-interpretability-research/
website

[8] **TransformerLens**
https://github.com/TransformerLensOrg/TransformerLens
tool

[9] **Why Should I Trust You? Explaining the Predictions of Any Classifier (LIME)**
https://arxiv.org/abs/1602.04938
paper

[10] **A Unified Approach to Interpreting Model Predictions (SHAP)**
https://arxiv.org/abs/1705.07874
paper

[11] **Datamodels: Predicting Predictions from Training Data**
https://proceedings.mlr.press/v162/ilyas22a/ilyas22a.pdf
paper

[12] **Interpretability in the Wild: Circuit for Indirect Object Identification in**

GPT-2
https://arxiv.org/abs/2211.00593
paper

[13] **A Mechanistic Interpretability Glossary**
https://www.neelnanda.io/mechanistic-interpretability/glossary
website

[14] **Fact Finding: Reverse-Engineering Factual Recall on the Neuron Level**
https://www.alignmentforum.org/posts/iGuwZTHWb6DFY3sKB/fact-finding-attempting-to
paper

[15] **Branch Specialisation (Distill Circuits)**
https://distill.pub/2020/circuits/branch-specialization
paper

[16] **A Mathematical Framework for Transformer Circuits**
https://transformer-circuits.pub/2021/framework/index.html
paper

[17] **In-context Learning and Induction Heads**
https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html
paper

[18] **The Hydra Effect: Emergent Self-repair in Language Model Computations**
https://arxiv.org/abs/2307.15771
paper

[19] **A Toy Model of Universality: Reverse Engineering Group Operations**
https://arxiv.org/abs/2302.03025
paper

[20] **Grokking Group Multiplication with Cosets**
https://arxiv.org/abs/2312.06581
paper

[21] **Detecting Hallucinations Using Semantic Entropy**
https://www.nature.com/articles/s41586-024-07421-0
paper

[22] **Do I Know This Entity? Knowledge Awareness and Hallucinations**
http://arxiv.org/abs/2411.14257
paper

[23] **Debating with More Persuasive LLMs Leads to More Truthful Answers**
https://arxiv.org/abs/2402.06782
paper

[24] **Concrete Steps to Get Started in Transformer Mechanistic Interpretability**
https://neelnanda.io/getting-started
website

[25] **Eleuther AI Discord**
https://discord.gg/eleutherai
website

[26] **Scaling Monosemanticity**
https://transformer-circuits.pub/2024/scaling-monosemanticity/
paper

[27] **Causal Mediation Analysis for Interpreting Neural NLP**
https://arxiv.org/abs/2004.12265
paper

[28] **Causal Abstractions of Neural Networks**
https://arxiv.org/abs/2106.02997
paper

[29] **Causal Scrubbing: Rigorously Testing Interpretability Hypotheses**
https://www.alignmentforum.org/posts/JvZhhzycHu2Yd57RN/causal-scrubbing-a-method-
paper

[30] **Locating and Editing Factual Associations in GPT (ROME)**
https://arxiv.org/abs/2202.05262
paper

[31] **How to Use and Interpret Activation Patching**
https://arxiv.org/abs/2404.15255
paper

[32] **Attribution Patching: Activation Patching At Industrial Scale**
https://www.neelnanda.io/mechanistic-interpretability/attribution-patching
paper

[33] **AtP*: Efficient and Scalable Method for Localizing LLM Behaviour**
https://arxiv.org/abs/2403.00745
paper

[34] **How Might LLMs Store Facts (3Blue1Brown)**
https://www.youtube.com/watch?v=9-Jl0dxWQs8
website

[35] **OpenAI Microscope**
https://openai.com/index/microscope/
tool

[36] **Refusal in Language Models Is Mediated by a Single Direction**

https://arxiv.org/abs/2406.11717
paper

[37] **Simulators (AI Alignment Forum)**
https://www.alignmentforum.org/posts/vJFdjigzmcXMhNTsx/simulators
paper

[38] **Curve Detectors (Distill Circuits)**
https://distill.pub/2020/circuits/curve-detectors/
paper

[39] **Emergent World Representations in Sequence Models**
https://arxiv.org/abs/2210.13382
paper

[40] **Emergent Linear Representations in World Models**
https://arxiv.org/abs/2309.00941
paper

[41] **Recurrent Neural Networks Learn to Store and Generate Sequences**
https://arxiv.org/abs/2408.10920
paper

[42] **Steering Language Models With Activation Engineering**
https://arxiv.org/abs/2308.10248
paper

[43] **Inference-Time Intervention: Eliciting Truthful Answers**
https://arxiv.org/abs/2306.03341
paper

[44] **Representation Engineering: Top-Down Approach to AI Transparency**
https://arxiv.org/abs/2310.01405
paper

[45] **Personalized Steering of LLMs via Bi-directional Preference Optimization**
https://arxiv.org/abs/2406.00045
paper

[46] **Towards Monosemanticity: Decomposing Language Models with Dictionary Learning**
https://transformer-circuits.pub/2023/monosemantic-features
paper