Key Details

Neel Nanda MATS 10.0 (Summer 2026)

Admission Procedure + FAQ

Apply here

Due Tues Dec 23rd 11:59pm PT

(Request extension)

TLDR

- Spend **~16 hours (max 20)** working on a **mechanistic interpretability research problem of your choice**, and send me a **write-up + executive summary** of what you learned (+2 extra hours).
    - See **advice**, **details**, and **past examples** in the other tabs. Note especially the list of **common mistakes** and how to choose a **problem I'm interested in**!
        - In particular, I am generally not interested in grokking, toy models, or (most) SAE work
    - For advice on getting started on the research task, how to generally approach research, scoping out a project, etc check out my **comprehensive blog post on how to become a mech interp researcher**, from learning the basics, growing as a researcher, and applying for jobs/professors.
- See my post on **pragmatic interpretability** (or this podcast episode) to learn about my current research approach
- The top ~34 candidates will do a **5 week paid online exploration phase** (**Feb 2** - **March 6**) ending in a 2 week research sprint in pairs.
    - Expect unstructured, self-driven learning
    - **3 weeks part-time, final 2 weeks full-time**
- The ~8 exploration phase candidates with the best sprint projects do the research phase, a **12 week paid** and **in-person** program (**June 1** - **Aug 21**)
    - The typical scholar publishes at least one **co-first author paper** at a top ML venue. I have 1.5 hr/week check-ins with each pair
    - Most exploration phase candidates **will not do the research phase**
    - Note the **3 month gap** before the research phase. You can spend this doing research with me, or take a break until the research phase
- **All backgrounds & experience levels welcome** - I want to work with the most promising people, not just those with the best credentials!
    - I design my process the way I do so that it's meritocratic, and can **identify promising people without experience**.

    - Past scholars include professors, undergrads with no mech interp experience, startup founders, software engineers with no research experience, and researchers with several great mech interp papers

Table of Contents

Key Details

- **Application task**: Spend **~16 hours** (**max 20**) trying to **make research progress** on a mechanistic interpretability problem of your choice**.**
    - **Submit** via this form, due Tues Dec 23rd 11:59pm PT
        - Note: I know some of you are busy and will only have time to do a good application over the holidays, so Dec 23 may be too early. I'll accept **late applications until Jan 2nd**, though can't guarantee I'll get back to you by Jan 12.
            - If you want to submit a late application, fill out this form explaining why you want an extension.
    - Please submit a **write-up** and **executive summary** showing me what **progress you made** and **what you learned** about the problem.
        - I value **communication skill**, don't rush the write up! The time limit has **up to two additional hours** for the executive summary.
        - See examples of successful past write-ups here
    - See **advice** on approaching the application, how to use LLMs for research, recommended resources, and how I evaluate applications
        - You can take as much time as you want beforehand for general learning.
    - My **research interests have changed** a fair bit from some of my prior work, I detail these here, and provide a long list of problems I'm currently excited about here.
    - I'm open to submissions of **existing mech interp work**, but hold these to a higher standard (more info)
    - If you've applied before, see here for a summary of **changes**
- **Key dates**:
    - Applications due **Dec 23**
    - Decisions released **Jan 12**
    - Exploration phase **Feb 2 - March 6** (**5 week online** program for top **~34** candidates)

    - Research phase decisions **March 12**
    - Research phase **June 1 - August 21** (**12 week in-person** program for top **~8** candidates)
- **All experience levels welcome:** I want to work with the most promising people, not those who look best on paper.[1]
    - In MATS 8.0 (my most recent completed cohort), **5 of my 8** scholars had **minimal prior mech interp experience**, but did fantastically - by halfway through the program, some of them had:
        - Helped **understand** **emergent** **misalignment** (i.e. why training a model to write buggy code turns it into a Nazi) and been interviewed about it by **MIT Tech Review**
        - Explored new paradigms for **interpreting** **reasoning** **models**
    - At the other extreme, I've had scholars who **already had multiple great mech interp papers**, like **Arthur Conmy** & **Josh Engels**, who say I still added a fair amount of value.

A reunion of my MATS alumni at NeurIPS 2025 - 20 alums in the same place!

FAQ

Why might you want to apply?

- My core goal is to teach you **how to do great mechanistic interpretability research**.

[1] In my 5 most recent cohorts, I've had 3 independent researchers, 9 ML PhD students/recent PhD grads, 10 undergrads, 3 ML masters students, 5 former software engineers, 1 physics PhD student, 1 ML postdoc, 1 neuroscience postdoc, 4 quant traders, an ML engineer, and 2 former entrepreneurs

- I run the Google DeepMind mechanistic interpretability team and I have a lot of experience supervising research. In the past 3 years, I have **mentored 50 junior researchers** and **supervised 30+ MATS papers**, and 15 top conference papers[2].
- The program often helps scholars get into **mech interp careers**
    - Seven now do interpretability research at **frontier AGI labs**, including Arthur Conmy, who works for me **leading** the **GDM Applied Interpretability Team**.
    - Two alumni **lead research teams** at the UK government's AI Security Institute
- Past scholars also do excellent research in the program itself, even those totally new to mech interp! Some highlights:
    - Showing open source LLMs can be cheaply jailbroken with linear algebra, by ablating the refusal direction
        - This inspired projects at multiple frontier labs, including a Meta paper on fixing it.
    - An ICLR oral using sparse autoencoders to interpret hallucinations, and showing models can "recognise" entities they know facts about.
    - Using interpretability to shape how models generalize without changing any data, preventing emergent misalignment
    - The first paper on transcoders, nine months before Anthropic's well-known papers on transcoders.
    - Work exploring fundamental issues in sparse autoencoders and follow-up work creating state-of-the-art methods that (mostly) fixed them.

Why is this application so much effort?

- I care a lot about being **meritocratic**. This way lets me find the best applicants, not just those who look good on paper. I do my best to assess your potential, not just what you've already done (though it's still super noisy!)
- I've also tried to design this application process so that spending time on it is **useful whatever the outcome** - I don't want to waste 12+ hours of your time!
- I think it's a pretty realistic **simulation of doing research**, especially if you haven't done interpretability research before. Candidates often learn a lot, and are surprised by how much they can get done.
    - I've sometimes heard from unsuccessful applicants that they enjoyed the application so much it convinced them to pursue a research career!
    - If you're not sure if you're interested in doing mech interp or not, I'd encourage you to try applying! I think you'll learn a lot from the application about whether it's a good fit.

What am I looking for in an application?

- My ideal application is one that **teaches me something new**.
    - This looks like identifying an interpretability hypothesis, gathering evidence for and against it, and writing up the evidence and analysis clearly.

---

[2] Note that almost all scholars in recent cohorts have published at least one co-first author conference, and many of the 30 papers are too recent to have finished peer review - list here. But my top priority is to help you do great research, publishing is a bonus.

- I value clear writing, good taste (ie choosing interesting problems and making good decisions), technical skill, truth-seeking, skepticism and pragmatism
- See a much more detailed explanation in this tab, along with past examples

What happens in the program?

- The **top ~34** candidates will do a **5 week online exploration phase** Feb 2 - March 6
    - The **final two weeks** (**full time**) are spent doing a **research sprint** in pairs. Admission to the research phase is largely based on sprint performance.
    - The **first three weeks** (**part time**) are the **preparation phase**. This means preparing for the sprint: self-driven skilling up, doing several day mini research projects with other scholars, going to talks/sessions, reading papers, etc. How you spend your time is up to you
    - More info here
- The **top ~8** candidates from the exploration phase will do a **12 week in-person research phase** in Berkeley June 1 - August 21
    - Scholars work in pairs to write a mech interp paper, with a **1.5 hr/week** check-in from me and some Slack support
    - ~All recent scholars have published this as a co-first author paper at a **top ML venue** (NeurIPS/ICLR/ICML) - see lists of past work below
- Research phase participants often do an optional **3-12 month extension**, to finish their paper and sometimes publish a second.
- All phases include a **stipend**: $4.2K for the 5 week exploration phase, $14.4K for the 12 week research phase. Housing support is provided in the research phase
- See more info at matsprogram.org

What happens if I don't get through to the research phase?

- While unfortunately most exploration phase candidates don't make it to the research phase, I've designed the exploration phase to be **a valuable experience in its own right**, and to teach useful research skills.
    - The median participant rates it as **1.5x-2.5x** the counterfactual use of time.
- In MATS 8.0 & 9.0:
    - At least 7 exploration-phase only scholars **found other MATS mentors** as a result of participating
    - I helped at least **8-10 exploration phase-only** scholars **write papers** based on their sprint projects (1 2 3 4 5)
- Candidates are welcome to try again in the next cohort

Why *shouldn't* I apply?

- Obviously, the application takes a while! If it doesn't sound fun, you probably shouldn't do it.
- The exploration phase of the program is fairly competitive, which some people find very stressful
    - Generally, participants seem to be nice and cooperative, especially since you want to form teams, but the awareness of your chances can be very stressful for some

- Most exploration phase events happen between 5pm-8pm UK time, which works badly for people in Asian time zones. But the events are not necessary for a valuable exploration phase!
- The exploration phase is very self-driven and unstructured - I provide good opportunities, resources, advice, etc and you all have each other as collaborators, but ultimately there's one of me and 30+ of you. You get out what you put in and need to decide how to spend your time. This works great for some, poorly for others
- If you have a full-time job/are otherwise very busy, you may find it difficult to make time for the exploration phase.

How should I choose a problem?

- I'm open to any application that shows strong research skill, but will be more excited about those matching my research interests
- **My research interests have changed a fair bit** from some of my past work - more details below, but in brief I'm now fairly pessimistic about ambitious interpretability (i.e. complete reverse-engineering), and I'm excited about pragmatic approaches with clearer applications to AGI Safety like model biology (studying qualitative high-level properties of models) and applied interpretability (rigorously doing useful things with interp). I'm still interested in basic science, but have a higher bar.
    - Applications that surprise me with something new and cool are fantastic!
- I'm more agnostic about the best techniques, things like sparse autoencoders are a useful tool, but easy to waste effort using when a simpler method is sufficient or better - start by doing the obvious thing!
- I provide a long list of suggested problems here

Can I use LLMs?

- Yes. In fact, I strongly recommend it! **LLMs are a crucial research tool** nowadays, and are especially useful for those getting into a new field.
    - More advice on using LLMs well below
- You're welcome to use them for coding, writing, etc, whatever you want - I want to gauge how well you'll do as a researcher, which includes whatever tools you'd actually use.
    - It is your responsibility to ensure your code and writing are high quality. Well-written write-ups are welcome. Docs that read like LLM slop will be rejected.
- I recommend using **Cursor for coding** (replacing eg VS Code) and using **Claude 4.5 Sonnet** or **Gemini 3 Pro for browser based tasks**
- I've compiled a folder of useful text files for mech interp research, containing a bunch of relevant docs & source code of key libraries, tutorials from ARENA and key libraries, key papers and my relevant blog posts.
    - By default, just **put this 600k token file in Gemini's context window**, which contains the most important documents[3].

[3] It starts with a table of contents explaining what's in it.

Do I need to be based in the US/have US work authorisation

- No and no
- The exploration phase is remote and can be done anywhere
- The research phase is heavily encouraged to be in person, but can be done remotely if need be
- It is an educational program for independent research, not formal employment which makes visas simpler.
    - MATS don't pay you, instead stipends are granted by another organisation, AI Safety Support
    - I do this in my personal time, and it is unrelated to my job at GDM

How does a research supervisor add value?

- My model is that research requires a mix of skills. The day-to-day coding and execution is crucial. But there's also a set of harder-to-learn conceptual skills, collectively called research taste. These skills take a long time to gain because they have poor feedback loops, but they take very little time to use.
- My main role is to lend you my research taste and bootstrap your own. This looks like helping with:
    - High-level Strategy: Choosing a good problem, knowing when to pivot away from a dead end, or prioritizing which of several promising directions to pursue.
    - Experimental Design: Designing a clean experiment to conclusively test a hypothesis, thinking of alternative explanations for your results, or knowing when evidence is strong enough.
- Navigating the Field: I can also give pointers to relevant papers or techniques you might be missing, helping you avoid reinventing the wheel.
- Finally, some people find it very helpful to have a de-facto light-touch manager who provides validation, accountability, and clarity.
- Past scholars have given me the feedback that I'm good at red-teaming, generating ideas, and being motivating and invested in their projects, but that I expect people to be able to work independently and can be fairly blunt with feedback.

If you have any questions that aren't answered by the other tabs of this doc feel free to email me.

Application Task Details

Application Format

Format: An application should consist of a **google doc summarising your key findings** which **begins with an executive summary** and ideally contains a bunch of graphs, and enough detail to follow what you did without needing to read your code. You're encouraged to include code, but it's not required, and I'll only read it as necessary to understand the write-up better. **Remember to let anyone with the link access the doc**!

Executive Summary Format

The first 1-3 pages of the google doc should be an executive summary, which gives the broad strokes of what you did and what you learned. Something at **~1 page** (including graphs) is great, **max 3 pages** and **max 600 words**. Please **include graphs**! Bullet points can work well

I will try to read the executive summary of every application, but won't have capacity to read every application properly, so please make sure it conveys the key info! You roughly want to convey the high-level takeaways/what you think is most interesting about what you've found, and a sketch of what key experiments you ran to validate it.

One good format is to have sections for:

- What problem am I trying to solve? (and a bit on why you think it's interesting)
    - Remember - what you write is always far clearer to yourself than to the reader! Though you can assume it will be read by someone with mech interp research experience
- What are your high-level takeaways? What were the most interesting parts of your project?
- One paragraph and graph per key experiment, giving the gist of what it was, what you found, and why this supports your key takeaways

Can I submit mech interp research I've already done?

- If you've previously done mech interp research (a (co-)first author[4] paper or non-first author but significant contribution paper or high-effort blog post), I'm open to you writing an executive summary for that work, and linking to it, rather than doing the normal application.
    - Please include an **estimate of how many hours the project took you**
    - If other people worked on it with you, please include a description of what you specifically contributed

[4] If you're not a first author but made meaningful research contributions, please outline what they were in your application

    - If it's not obviously a mech interp paper then please explain why you think it shows you have relevant skills.
- I'd prefer a standard application, and I'll judge these more harshly than normal applications (you likely had much more time), but if you otherwise won't have time to apply I'd prefer to get these!
    - If you won't even have time to write an executive summary, I'd still rather get your application than nothing, but have an extremely high bar for those.
- If the previous work was done on your own and in <=20 hours, but not *for* the application, this is obviously fine, and you can just treat it as a normal application project.

Defining the 20+2 hour time limit

- Not counted:
    - General prep (paper reading, tutorials), that you would have done before deciding on a project
        - My fairness principle here is "anything you could reasonably have done to learn mech interp on your own, before thinking of a problem to work on, is totally fine, because more experienced applicants could have done that already"
    - Generic tech set up, like renting and setting up a cloud GPU, that you'd need to do for most projects
    - Breaks
    - Time spent waiting for things to train (assuming you're doing something else during this time, eg training an SAE overnight)
    - Writing your answers to the MATS application form
- I consider any time you spend actively working towards the project goals to be within the 20 hour time limit. This includes (but is not limited to):
    - Writing code for your project
    - Reading papers (chosen because they're relevant to your project)
    - Analysing data/experimental results
    - Thinking and planning time
    - Writing up the google doc
- So the executive summary doesn't get super rushed, you can take another 2 hours for it.
    - I ask that you don't edit the rest of the write-up, and don't write any new experiment code, though you're welcome to write code to make new graphs/visualisations from data you already have, if it'll help present the results better
- You're encouraged to track your time with a tool like Toggl and include a screenshot with the application doc
- If you decide your project is doomed, you're welcome to give up and start a new one, and reset the timer

Advice on good applications

How to produce a good application in 20 hours

I recommend thinking of the application as a mini-research project. My standards are obviously lower than for a full paper, but the best applications look like small, self-contained research investigations. They essentially speedrun the process of identifying an interesting hypothesis, carefully testing it, and then clearly communicating the results. My blog posts on my research process (Explore, Understand, Distill and Key Mindsets) have more detail, but I've summarized the key ideas here.

Research Advice

There are three key phases to a good research project:

1. **Exploration:** The goal here is simply to **gain information and build intuition**. A common mistake is thinking this stage ends once you've picked a problem, e.g. from the list below. In reality, much of a project is spent just figuring out what's going on.
    a. You don't need a clear hypothesis yet. Often the best uses of time are things that expose you to lots of information.
    b. **Get your hands dirty**. Try things like reading your data, giving your model interesting prompts, or seeing what a sparse autoencoder tells you.
    c. This doesn't mean you don't have a plan. It means your plan is to maximize information gain per unit time. Constantly ask yourself: "**Have I learned anything in the last 30 minutes?** Is this direction still fruitful?"
2. **Understanding:** Once you have a hunch about what might be true, your goal is to **convince yourself it's true** with careful experiments.
    a. Keep a running doc with a list of your hypotheses. Alternate between designing an experiment to test one, running it, and analyzing the results.
        - Put key graphs and findings in your doc, as you learn more about hypotheses - you don't want to forget where a key experiment is!
    b. It's crucial to keep track of the kind of claim you are trying to make.
        - Sometimes you want to give an existence proof (e.g., find an example of an interesting phenomenon), where cherry-picking is fine.
        - Other times, you want to argue a method is the right thing to do for a task, which requires comparing to baselines.

    c. **Common mistakes:** Getting too excited and missing simple alternative explanations for your results; running a bunch of experiments that are only vaguely relevant instead of striving for **conclusive evidence**.
3. **Distillation:** This is where you turn your findings into something legible that can convince others. This means writing up your work clearly and honestly.
    a. **This is not an afterthought!** People often neglect the write-up, but it's crucial. **If I don't understand what you did, I will reject your application.**
    b. Given the time limit, you won't achieve the full rigor of a published paper (e.g., large sample sizes, extensive baselines). That's fine! But the principles of providing clear evidence for your claims still apply.
        - Crucially, **avoid relying only on a few cherry-picked qualitative examples**—this is a major red flag.
        - And remember to compare to baselines, if applicable

Writing Advice

It's extremely important to have a good write-up! The advice in my post on writing ML papers may be helpful - obviously, I don't expect a formal paper, but the principles of clear communication apply.

- **Focus on a Narrative.** Don't just dump all your experiments. Structure your write-up around the one or two most interesting, concrete insights you found. What is the key story?
- **Quality over Quantity.** One interesting finding, well-explained and well-supported, is far better than ten superficial experiments.
- **Show Your Work.** Explain *why* you ran an experiment, not just *what* you did. What hypothesis were you testing? What were the possible outcomes? This reveals your thought process.
- **Your Reader Has Zero Context.** The "illusion of transparency" is a huge trap. Things that feel obvious to you will be completely new to your reader. Explain everything from the ground up. Define your terms. Label your graphs clearly.
- **Make Your Executive Summary Count.** It needs to stand on its own and convey the most important takeaways and a sketch of your key evidence. Don't make me hunt for the point or crucial details. Good graphs are a huge plus here.

Guidance on using LLMs

You are actively encouraged to use LLM assistance for your application—I want to gauge how well you'll do at research in practice, so if you'd use it there, use it here! And if you don't use LLMs as part of your research, I think you're probably making a bad decision. In particular, while LLMs struggle to attain expert performance, they're pretty good at beating novices. So if you're trying to get into a new domain, like mechinterp, they can be incredibly helpful, if you know how to use them[5].

[5] For anyone seeing this and thinking of the METR study showing that LLMs slowed people down, I don't think that transfers, though the lessons for what not to do remain! Those were experienced software engineers working in codebases they knew well, i.e. experts, not novices

Here's some advice on using them effectively - I've written this for an audience who have not used LLMs much, but I hope there'll be at least one novel point even for experienced users.:

- **Meta-prompting**: A good system prompt can be extremely helpful. If you don't have strong preferences, or much experience writing your own prompts, I recommend telling an LLM what you want and getting it to write the prompt for you.
    - If the prompt does not do what you want, tell the LLM this, give detailed feedback, and ask it to rewrite the prompt to prevent a repeat.
    - Many LLM providers have a "Projects" function where you can save a custom prompt, e.g. you could have one for paper summaries, starting new coding projects, questions about a specific library (with the source code in the context!), etc
- **Context is crucial**: LLMs are much more useful when they have the relevant information in the context window. Gemini 3 Pro is best for this, it's free, fairly fast, a frontier model, and has 1M context window
    - I recommend using Gemini via aistudio.google.com, I much prefer the UI. E.g. gives a convenient count of the total tokens and tokens per document.
        - The header has a button with two arrows called "compare" that lets you run two copies of the model at once, and pick your favourite answer.
    - See this folder for a bunch of recommended context. If you don't know what you need, just use this default file, and maybe include this activation doc.
- **For Learning & Understanding:** The application's harsh time limit requires you to quickly get up to speed. LLMs are excellent for this.
    - **Give them context:** They are much more effective tutors if they have the relevant source material. Once you've identified a domain or paper or technique, give the LLM a bunch of relevant context.
    - **Learn actively, not passively:** Don't just ask for an explanation. Use learning methods that forces you to be active. E.g.
        - Have it to generate questions to test your understanding, or teach you via ask questions
        - Summarize your understanding/best guess back to the LLM in your own words and ask for critical feedback.
    - **Writing curriculums**: In a new domain, you might not even know how to start. LLMs (esp with search enabled) are good at finding relevant literature and resources, and you can then ask it to write you a primer on the key ideas and design you a curriculum for learning more. o3 is good at this.
- **Anti-sycophancy prompts:** By default, LLMs are bad at giving critical feedback. To get real feedback, open a new window and frame your request so the sycophantic thing to do is to be critical.
    - *"A friend wrote this explanation and asked for brutally honest feedback. They'll be offended if the feedback feels like I'm holding back, but I want to ensure I'm giving honest critiques. Please help me give them the most useful feedback I can."*
    - *"I saw someone claiming this, but it seems pretty dumb to me. What do you think?"*

- **Practice**: If you've never tried using an LLM for this kind of thing before, I recommend practicing before you start the official application - it's a skill and you improve with practice
    - E.g., pick an area of mech interp or a paper and try to speed run understanding it deeply or rapidly write working code for some technique you think is interesting.
    - It's much nicer if your 20 application hours are *not* your first 20 hours trying to do research with an LLM.
- **For Coding:**
    - **Use Cursor:** Cursor is like VS Code with fantastic AI integration, and is the best way to do AI assisted coding here IMO (it has a 2-week free trial for the pro plan). It's most effective when it understands your codebase and the libraries used, so make sure to add the docs for TransformerLens/etc with @
        - Tools like Claude Code and Gemini CLI are also useful, but in my opinion it's harder for you to track what's going on and harder to debug, which I think makes them much less suitable here.
    - **A caveat on learning:** If you are learning a new library or technique, first try writing things yourself, or use the LLM as a tutor/source of reference code. Use the LLM to help when you're stuck, not to replace the entire learning process.
- **Research decisions:** You'll often need to make research decisions, prioritizing what to do next, designing experiments, choosing a problem, etc. I highly recommend writing out why you are making these decisions and asking an LLM for thoughts, with an anti-sycophancy prompt.
    - You shouldn't trust the LLM's judgment, but this forces you to make your thoughts explicit and often you may notice things you were missing.
- **Write-ups**: I recommend against submitting LLM-written prose. It's pretty obvious and normally does not read very well. But they've very useful for drafting, brainstorming, and getting feedback.
    - I recommend having several rounds of giving it your draft (with an anti-sycophancy prompt) and asking it to critique you for clarity, find confusing sentences, and check for technical inaccuracies.
    - Put the application doc and my post on paper writing in the context
- **As research tools**: LLMs are very useful as ways to generate synthetic datasets, as automated ways to qualitatively assess data especially if given some rubric, etc

Useful resources

Please bias towards getting your hands dirty, and focus on writing code, running experiments on the model, and getting feedback from reality. **I recommend spending at most 5 of the 12-20 hours reading papers and tutorials**. You're welcome to do general reading and learning beforehand, so long as it isn't directly making progress on your planned project.

Coding

- If your project involves using a <=9B model, and you want to play around a bunch with the internals, I recommend using TransformerLens.
- Otherwise, I recommend using nnsight as it is more performant and works well on larger models - it's just a nice wrapper around a PyTorch model.
- The ARENA tutorials are fantastic, both as an intro to TransformerLens, and as a practical coding intro to mech interp techniques.
    - If you're new to mech interp and time constrained, prioritise doing the first 3 sections of chapter 1.2 to get the basics
    - The general ML ones are also solid
- You can find concatenated docs, source code, and tutorials of TransformerLens and nnsight, and the concatenated ARENA tutorials, in this folder so you can put them into an LLM.
- If you need an LLM API, I recommend OpenRouter, it lets you access basically every model with same interface.
    - If you want to intervene on the chain of thought of a reasoning model, eg partially filling it and regenerating the rest, a la thought anchors, I recommend Nebius
    - To do initial testing it can be easier via an online chat interface. poe.com is a good way to try out lots of models
- Re what LLM to make the subject of your research:
    - Gemma 3 and Llama 3.3 are solid non-reasoning models.
    - If you want to work with a reasoning model, use Qwen 3. If you need a really good one, try Nemotron 49B.
    - Don't use mixture of experts models unless you need to, they take up a bunch more GPU memory and are a pain.
    - If you want to work with SAEs, use Gemma 2 and Gemma Scope.
- For writing code, use Cursor
    - It also can also deal with a fair amount of the setup costs of a new environment.
- I recommend renting and using your own cloud GPU, rather than toy coding environments like Colab - if you haven't done this before, lean heavily on LLMs for tech support. Instructions
    - To rent GPUs, I recommend runpod.io If cost is a constraint, vast.ai is the cheapest provider[6]

Other resources

- My mech interp reading list for an overview of key papers (a bit out of date, alas)
- Ferrando et al is a good overview of key techniques
    - Key interp techniques to focus on: Direct logit attribution, activation patching, maximum activating dataset examples, linear probes, steering vectors, sparse autoencoders

[6] Unfortunately, we can't practically provide compute funding to applicants. The exploration and research phases have compute budgets though.

    - Key black box techniques: Prompting LLMs, fine-tuning (including LoRAs). Baselines are important!
        - Good test - do you understand why token forcing is effective and hard to fix?
- 3Blue1Brown's ML videos are delightful, and now cover transformers
- My youtube tutorials, especially my intro to transformers, my research streams and my talk on thinking models
    - My talk on different research philosophies may be helpful if you want a better big picture of the field and various disagreements
- Maths fundamentals - main thing you want is strong linear algebra, with some probability, information theory, vector calculus and optimization. ML is pretty conceptually simple tbh
    - The 3Blue1Brown Linear Algebra series is great
    - For the rest, get an LLM to teach you and quiz you on problems with the socratic method.
- For any work with SAEs:
    - The ARENA SAE tutorial (warning: it's very long! You should skip around)
    - Gemma Scope, high quality open weight SAEs on every layer and sublayer of Gemma 2 that my team produced
    - Neuronpedia, an excellent online tool to look up explanations for different latents (aka features) in open source SAEs, run
        - Their Gemma Scope demo is a good place to start if you're new to SAEs
        - They also have an API you can use in a Jupyter notebook to request information about a latent
    - The dictionary learning library: a more hackable and barebones alternative to SAELens. If you want to do anything unusual with SAE training, I recommend using this over SAELens.
- If you want an introduction to mech interp, check out my Machine Learning Street Talk podcast interview or this recent lit review/research agenda[7]
- I wrote my thoughts on theories of change for interpretability helping with AGI Safety in the GDM AGI Safety Approach
- My mech interp glossary (a bit out of date, but still useful)

[7] I don't agree with everything in here, but it's a decent overview/aggregate of many researchers takes

What does a good application look like?

How are applications evaluated?

What do good applications look like?

- **Clarity**: If I understand what you're claiming, what evidence you're providing, and think that evidence supports your conclusion, that instantly puts you in the top 20% of applicants.
    - Show me enough detail so I can follow along: how did you generate your data or choose your prompts, how did you define your metrics, what were your hyperparameters, etc.? This can be concise if done well—bullet points and short code snippets can go a long way.
    - I like bullet points, good graphs, summaries, good structure, and intuitive explanations to get the high-level picture across clearly - more advice here.
- **Good Taste**: You chose an interesting question, and were able to get traction on it, and produce results I find compelling. My favourite kind of application is one where I learn something from it.
    - This doesn't have to be a big, ambitious claim—just any claim that's not immediately obvious without evidence.
    - Originality is a big plus. If I've seen a bunch of applications doing extremely similar things, this is less exciting.
    - Having interests aligned with my research interests is a significant plus.
- **Truth-seeking and Skepticism**: The easiest person to fool is yourself. You constantly questioned your results, looked for alternative explanations, and did sanity checks. Negative or inconclusive results that are well-analysed are much better than a poorly supported positive result. (more advice)
    - The key thing to emphasize is self-awareness and clarity. It's a harsh time limit, so there are going to be holes in your results. It's OK if you show self-awareness of where the holes are, which parts are speculative, what you would investigate next, etc. If you seem overconfident in shaky results, that is not. Make plausible claims over ambitious ones.
    - A subskill here is **attention to detail**: Noticing subtleties and edge cases, and investigating them where appropriate
- **Technical Depth & Practicality**: You demonstrate a good handle on the relevant tools, whether that's coding, experiment design, or specific interpretability methods. You show a willingness to get your hands dirty writing code and running experiments. Your writing and design decisions make it clear that you understand what you're doing and it's well motivated, rather than blindly following a recipe/LLM
    - Useful areas of knowledge: knowledge of mech interp papers and techniques, ability to work with large models on GPUs or train SAEs, fluency with linear

algebra, understanding of transformers, understanding of ML, coding skill, ability to design good interactive interfaces and visualisations, etc.

- **Simplicity**: Being biased towards trying the simple, obvious methods first (or explaining why they were unsuitable). It's easy to get excited by fancy techniques, but they can be a trap. Good applications are pragmatic and focused, not showing off.
    - E.g. in recent work from my team into why models seemingly showed self-preservation, we started with the obvious things of reading the CoT and prompting and, er, it just worked, and we stopped there and wrote up the post.
    - Each piece of complexity in the project should be there for a reason
- **Prioritisation**: You used your time well, and went deep on one or two key insights, rather than being superficial about many things (more advice)
    - A common mistake is getting caught in **rabbit holes** - finding one random anomaly or detail that (in my opinion) isn't very interesting, and spending the whole time zooming on that. Knowing when to pivot where appropriate is impressive
        - If you're totally changing directions (ie, so that your code and findings so far isn't particularly helpful for the new direction), I'm fine with you restarting the 20 hour limit.
    - Another is spreading yourself too thin - doing lots of things superficially, but without enough depth for any one to be interesting
    - Yes, these tips point in opposite directions. Sorry! You need to balance between these two extremes. This is hard and I don't expect anyone to do it perfectly. I recommend setting a timer every hour or two to zoom out and ask if you're making progress or caught up in a rabbit hole.
- **Productivity**: While it's more important to do things well than do them fast, the ideal is both. Some researchers are a lot more productive per unit time than others, and they get a lot more done. (more advice)
    - This isn't about cutting corners - there's a lot of skill to having fast feedback loops, noticing and fixing inefficiency where appropriate, and being able to take action or reflect where appropriate.
- **Show your work**: It's great to see your thought process, understand why you made the decisions you made, etc. This matters most if your results are inconclusive or key parts failed: if you want me through what you tried and why, and what happened, and I think you made reasonable decisions, that's still impressive.
    - The difference between "I got stuck so I gave up" and "I got stuck, so I pivoted or found a new angle, or identified the reason why it didn't work" is huge.
    - Though if you *do* have an interesting finding, please structure the write-up to emphasise it, don't do chronological order!
- **Enthusiasm** & **Curiosity**: Mech interp can be hard, confusing and frustrating, or it can be fascinating, exciting and tantalising. How you feel about it is a big input here, to how good at the research you are and how much fun you have. A core research skill is following your curiosity (and learning the research taste to be curious about productive things!)
    - I know this is easy to fake and hard to judge from an application, so I don't weight it highly here
    - But generally applications that are fun to read get bonus points!

Beyond the application task

I evaluate application tasks according to the criteria above, and by my intuitive sense of "did I learn something interesting from reading this?" A good application task is enough for acceptance, whatever your background.

Beyond this, I do my best to evaluate an application holistically - I want to understand who will be able to do great mech interp research. Naturally, examples of good prior mech interp work are strong evidence here. Beyond that, it's hard to learn too much, but I can get some signal to help with tiebreakers. These *can* be legible credentials, but aren't always. An insightful Arxiv paper is much better evidence than a NeurIPS oral I don't find interesting.

I'm also excited by non-standard credentials. In the application form, I ask: "What are 1-3 pieces of evidence that you'd be able to do good research in the program?" This is your chance to highlight things like:

- Popular open-source projects you've built.
- Startups you've founded.
- Blog posts you're particularly proud of.
- Impactful things you did at work or in class projects.
- Something interesting I didn't think of when writing this list!

If you've done something cool, and you think a reasonable person would update positively on hearing it, please mention it and explain its relevance!

I don't care too much about prior knowledge - if you're good enough to do a decent application task, that's good enough for me. Mech interp is a young field, so it doesn't take that long to learn enough to do original research, especially with modern LLMs and me to help you prioritise. It helps to have experience with mech interp, ML, or maths, especially having good linear algebra intuitions, and basic coding or ML experience, but it's not required.

Note: I may use LLMs to help me with application review, but I will make the final decision on each application. Other MATS mentors will have their own policies.

Common Mistakes

Some common mistakes I see that can really harm an application's chances:

- Skepticism:
    - Not acknowledging limitations in their results (worse, trying to pretend negative results are positive - negative results are fine! Lying about them is not)
        - Related: Trying to hype up their results and make them seem way more interesting than they are. Just be honest! I can tell
    - Not thinking about ways their results could be false and doing sanity checks. A really *positive* sign about an application is when I think of a way the results could be false, then discover you've already checked it!
    - Overcomplicating things - eg having a super complex hypothesis about some phenomena without checking a really simple hypothesis. Or trying a really

high effort method without trying something simple like prompting, reading the chain of thought, or training a linear probe
        - Start with an open mind -
    - Trying to investigate some phenomena without checking if it's really there, e.g. theory of mind in GPT-2
        - Related: Working with a model that's just way too dumb for the task. There's no good reason to use GPT-2 in your application at this point
    - Not looking at your data - read some data points! Talk to your model! If something seems weird, look closer! There's almost always something worthwhile to learn here, but this key step is often neglected (including by professional researchers)
- Problem choice:
    - Choosing an uninteresting problem, eg something both fairly unambitious *and* which isn't anything to do with my research areas of interest, like an incremental improvement to sparse autoencoders, or applying IOI-style circuit finding to a random problem
        - A warning sign is candidates with a particular pet interest. If you're e.g. really excited about medical applications of AI, you're welcome to do a project on this, but there's a good chance you do a project that *only* people interested in medical applications of AI find interesting
    - Choosing a problem that's really far outside my interests, e.g. something entirely theoretical, or which only involves tiny toy models
    - Choosing a problem that doesn't really make sense
    - Choosing a problem that's super ambitious, or conceptually messy, and getting very confused
- Strategy:
    - Realising the project is probably doomed halfway through, and just continuing the project rather than trying to pivot. Knowing when to give up is a key research skill!
        - If you totally change project direction, feel free to reset the 20 hour time limit
- Misc:
    - Poor writing - if I can't understand your summary in the application form / executive summary, I probably won't have time to decipher your research report and figure out if there's something interesting here. Conversely, good communication skills are a big plus. There's a reason I give an extra 2 hours for the write-up!
    - Submitting an entirely LLM written application, about made up experiments (please don't do this…)

Examples of past applications

Here's a bunch of past examples of successful applicants who have kindly offered to have their applications publicly shared. Each has some lightly edited LLM summaries of my notes to give you some idea of what I'm looking for and what I'm thinking about when I review applications.

R1 Distill Diffing (MATS 8.0)

Project: Training a crosscoder to diff a Qwen-Math model and its R1 distill, finding that the R1 distill adopted a more "informal reasoning" style compared to the base model.
Assessment: The project was very productive, and showed good prioritisation and pragmatism by creating a new dataset and using LLMs creatively to find patterns when the primary method failed, though it suffered from a conceptual error in how it defined model-specific latents.
Decision: This is a borderline accept; while the project had a key technical flaw, the strong pragmatism, productivity, and ability to extract an interesting qualitative insight despite setbacks showed strong research potential.
(Note: Despite being a borderline accept, this scholar then made it to the research phase and has been doing great - application processes are really noisy!)

Empathic Machines (MATS 8.0)
Project: Do AIs represent the emotional states of users, and can we causally affect its actions by steering with these? Showed this worked for simple emotions on a toy synthetic dataset.
Assessment: A cute, small idea - well executed, but I expected it to work and the strength of the conclusions are inherently limited by the data quality, so I didn't learn too much from this. Notably well written and presented, this was very easy to understand.
Decision: Borderline accept - though there are strong limitations, they did find some convincing findings, communicated them well, and showed good self-awareness of the limitations and how it might be extended.

What Impacts CoT Faithfulness (MATS 8.0)
Project: The project investigated several factors impacting Chain-of-Thought faithfulness, finding that it was lower for multiple-choice questions than open-ended ones, and providing evidence for the nostalgebraist's self-correction hypothesis.
Assessment: This was a well-prioritized project that showed good taste in choosing an interesting question, built well on existing work, made reasonable decisions, and skeptically tested key assumptions. It was purely behavioural, while most applications were mechanistic, and mechanistic work is slower, so I would have expected more output from a strong application. Writing was difficult to follow, and tended to assume too much context on behalf of the reader.
Decision: On the higher end of borderline accept - they found some insights, on a well chosen question, but communication and volume of output could have been better.

"Wait", backtracking in CoTs of reasoning models *is* intentional (MATS 8.0)
Project: Is backtracking in a reasoning model's chain-of-thought is an intentional behavior? They found through black-box analysis that it is not random, and then used an SAE to identify latent directions correlated with it.
Assessment: The project was very productive and demonstrated strong pragmatism and technical depth by tackling the problem with multiple methods, from large-scale interventions to training an SAE.
Decision: This is on the high end of borderline accept - it's a well-executed and competent investigation that tries many sensible things, but the findings are just confirming a reasonable hypothesis, and it was too broad to have time to go deep on the mechanistic findings and clarify what was happening.
Note: I bumped this up to an accept because I was impressed by the candidates profile, they'd only discovered mech interp a few weeks before, but had done a bunch of self-study demonstrating proactivity and agency and genuine motivation, which suggested high potential. They'd further demonstrated impressive agency with some of their other achievements, like founding a start-up, and side projects making widely used pieces of software (this scholar then made it to the research phase and did great, so this was an accurate prediction!)

R1D1 - Is Reasoning in Language Models Mediated by a Single Direction (MATS 8.0)

Project: The project investigated whether a "reasoning direction" could be identified in a language model's activation space between Llama-3 8B and its R1 distill. It found a direction that could suppress or enhance reasoning.

Assessment: The central idea wasn't super original, but it was a sensible idea executed well and with genuinely interesting results, showing good taste and competence. They showed pragmatism in pivoting after the initial hypothesis failed and communicated this clearly. While the conceptual analysis was a bit limited, the project succeeded in teaching me something new. (Bonus points for a great title)

Decision: Accept. The project is a strong application: it's well-executed, well-scoped, pragmatic, clearly communicated, and taught me something.

SAE Equations (MATS 6.0)

Project: Designing an algorithm to find SAE latents with arithmetic relations, a la king + woman - man = queen. Found some very cool examples

Assessment: Not the most productive of applications, but a nice and tasteful choice of problem, which was well motivated and well executed and found some lovely qualitative results

Decision: Accept - shows good taste, ability to do research, and I learned something new, though more output would have made it stronger.

(Note: While I'm less keen on SAEs these days, I think the style and research skills demonstrated here stand, and this might still have made the cut today)

Recommended Research Problems

How my research interests have changed

My research interests and views have changed a fair bit since early 2024. This means that many of the topics I've done past work on aren't necessarily the topics I'm most excited about supervising future work on, which is often misunderstood by applicants. To help you choose a problem I'm likely to be interested in, see my blog post on a pragmatic vision for interpretability, and this post on ways I think it can help AGI go well

Some other resources:

- My 80,000 Hours podcast interview is a good source on my takes and why they changed (though I've refined them somewhat since then)
- A talk series I gave to my MATS 9.0 scholars about:
  - The big picture of what matters right now in mech interp
  - How I see mech interp helping make AGI safe
  - The story of sparse autoencoder research in mech interp and mistakes I made here, which sparked many of my changes in perspective

If you hear all this and are like, "that sounds really boring, I am no longer interested", then great - we probably wouldn't have been a good match! It's much better to learn that now than later. There's a bunch of other MATS mentors who'll be opening applications soon, hopefully one of them is more aligned with what you're looking for.

Suggested Research Problems

The below are a bunch of recommendations for things I would be excited about. Strong applications often riff off of these ideas - coming up with their own approach, but along similar themes to the below. You should not feel constrained to the problems on this list, but hopefully it can serve as some guidance for the types of questions I'd be excited to see.

Warning: The ideas below have **not** been filtered for "I am confident someone could make progress on this in 20 hours". Pick something where you have some idea of how to get

started (or read around the field a bit and try to generate ideas and a sketch plan before picking a problem)

I broadly categorize my interests into three areas: **Applied Interpretability**, **Model Biology**, and **Basic Science**.

Model Biology

This is about studying the high-level, qualitative properties of a model's behavior, treating it like a biological organism we're trying to understand. This is especially interesting for weird, emergent, or safety-relevant phenomena.

Understanding weird behaviour

- **Deep dives into mysterious behavior:** Take a weird model behavior (e.g., a model seemingly exhibiting self-preservation, blackmailing, or faking alignment) and try to understand what's happening.
    - **Tip**: Start with the smallest model that shows the behavior.
    - Example work from my team at GDM: We analyzed why models seemed to stop you from shutting them down (in a prior demo) and found that the model was actually just confused. We started with the simplest technique of reading the chain of thought to generate hypotheses and testing them causally by varying the prompt, and this just worked. Remember to start simple!
- **Debugging**: Can you take an example of model behaviour where it's behaving in unintended ways, understand what's happening, and ideally fix it? E.g. why it thinks 9.8 > 9.11

Reasoning Models

- **Chain of thought faithfulness**: An obvious safety strategy is “let’s just read the chain of thought and see if it thinks anything suspicious”. Can you find evidence for or against this being effective?
    - Can you learn anything by studying examples of unfaithful chain of thought? Eg from Arcuschin et al or Chen et al
    - Can you create situations where the model does use suspicious reasoning? E.g. by training the CoT to be distorted, or telling it that its chain of thought is being monitored - does it do the bad behaviour but without suspicious reasoning? How complex can the behaviour be?
        - I would love to see someone use synthetic document fine-tuning to train a model to believe that it should have unfaithful chain of thought and see if we can interpret what's happening.
    - Can you design good monitors or metrics for whether the chain of thought is telling us what we think?
    - Can we tell when a CoT was causally important for a model giving its answer?
        - This is a subtle concept to pin down – models may have a causally irrelevant CoT that gives the same answer. And editing the CoT

throws the model off distribution, which may change its answer for unrelated reasons

    - What factors lead to different forms of “unfaithful” CoT? Can we distinguish them?
        - Deliberately rationalizing a preconceived answer (Arcuschin et al., Lindsey et al.)
        - Models changing their answer because of a hint but not admitting it (Chen et al.)
        - Models taking logical shortcuts in maths problems after getting stuck, to claim they’ve achieved a valid “proof” (Arcuschin et al.)
        - Models giving a reasonable chain of thought, but at the last minute “flipping” to a different final answer (Arcuschin et al.)
- **Thought anchors**: In Bogdan et al, my scholars present a paradigm for what mech interp could look like for reasoning models, where we study sentences as our main unit of analysis, and use tools like resampling to understand which sentences are important, and do causal interventions to understand the dependence between pairs of sentences. How can you extend and build on these techniques? Can you find anything interesting by using them? Can you find any weaknesses or limitations?
- **Steganography**: Can models encode information in their chain of thought? Ideally, in a way where the chain of thought is still plausible to us, but even in ways where we know it's encoded but we don't know what it means. Can you train a model to have reasoning we don't understand and use interpretability tools to decipher it? Relevant work
    - Note that this needs to allow the model to do tasks it couldn't do without a chain of thought to be interesting.

Interesting phenomena

- **User models**: Chen et al shows that LLMs form surprisingly accurate and detailed models of the user, eg their gender, age, socioeconomic status, and level of education, and do this from very little information. They can find these with probes, and steer with these to change the model’s actions in weird ways.
    - This is wild! What else can we learn here? What else do models represent about the user? How are these inferred? How else do they shape behaviour?
    - Do LLMs form dynamic models of users for attributes that vary across turns, eg emotion, what the user knows, etc.
        - As a stretch goal, do LLMs ever try to intentionally manipulate these? Eg detect when a user is sad and try to make them happy
- **Out Of Context Reasoning:** Sometimes models generalize much further than expected. Most famously, emergent misalignment, where training a model to write insecure code turns it into a Nazi. What's up with this? Some past work from my scholars suggests this is often downstream of learning a single direction, with hints that it's because the general solution is more efficient. But there's a lot we don't understand - is this the whole story? Why are some solutions easier to learn than others? Do these weird effects come up in any real use cases?
    - A notable example is synthetic document fine-tuning, where training on LLM-generated documents from a world where some false fact is true can get

LLMs to internalize it and act on the consequences of that false belief. What's going on here? Does this really work? How robust is it? Etc.

- **Concept Representations**: How are specific interesting concepts computed and represented?
    - Can we train a truth probe that generalizes well to real situations?
    - What about a deception probe?
    - How is uncertainty represented?
    - Why on earth is there a misalignment direction?
    - How is the awareness of whether or not it is being evaluated represented? Nemotron 49B seems like a good model to study here.
- **Conflicting information**: How do models deal with conflicts between instructions or goals, or their prior knowledge and the context?
- **Model Diffing:** What changes during fine-tuning? Comparing a model before and after a change (e.g., chat-tuning, instruction-tuning, or fine-tuning on fake facts) can be a powerful way to isolate what was learned - see my past scholar's work on diffing chat finetuning for more pointers.
    - Seeing what happens during reasoning fine-tuning could be particularly interesting. Venhoff et al used simple techniques like per token KL divergence to study the high level differences, while Ward et al zoomed in on backtracking, a specific behaviour.

Circuit analysis

- **Attribution graphs**: Are attribution graphs a useful technique for understanding model biology? Can you find anything interesting with the graphs on Neuronpedia? Can you find ways to overcome some of their limitations? Can you find things with them that cannot be found with simpler techniques like guessing and checking?
    - How important is precision? One notable consequence of the attribution graph approach vs, e.g. prompting, is that it can find much more nuanced and detailed hypotheses, like the addition analysis in Lindsey et al. Are there tasks where this precision is important?
- **Baselines**: There's a bunch of simple methods that fundamentally boil down to guessing hypotheses and checking them. Far more effort has gone into fancy techniques like attribution graphs than these. How far can we push them?
    - Linear probes can be highly effective at identifying concepts the model is representing – can we automate and scale the process of testing many linear probes, at all appropriate layers / token positions, for a given task?
    - Scaling the process of reading a model's chain of thought. How can we best analyze and aggregate them to look for unexpected properties, across many prompts? Docent is one interesting approach in this direction.
    - Simply observing model behavior in response to an appropriate mix of prompts can be highly effective to infer mechanistic hypotheses, but there's an art to doing it well. What do best practices here look like? Can they be automated?
- **Automation**: Can we automate the full hypothesis generation + validation loop with LLM agents?
    - Automated hypothesis generation

        - Can LLMs simply guess the high-level casual graph of a task? Can an agent make more headway if we let it iteratively choose diverse prompts and read the output
        - How good are LLMs at interpreting an attribution graph and how good can we make them with the right prompt and scaffold?
    - Automated validation
        - Can we automate the design of probes to test for the presence of predicted features?
        - Can we automate intervention experiments, and synthetic / out-of-distribution inputs, used for hypothesis validation?

Objectively Measuring Interpretability

- **Eliciting Latent Knowledge:** Can we use interpretability to elicit secret knowledge from a model? What techniques work best?
    - In Cywiński et al my scholars taught a model a secret word by training it on descriptions of that word, and then retrieved it with both black and white box techniques. Can you do better? (their models)
    - Can you create more interesting model organisms? Marks et al did a much more ambitious version, involving teaching a model fake facts with synthetic documents, what else can you do with that technique?
- **Understanding-based downstream tasks**: In addition to the above, what other objective tasks are there that test our success at understanding? Movva et al. is another nice example.

Applied Interpretability

I'm excited about a work that finds practical, real-world applications of interpretability, especially for safety. This isn't just using downstream tasks for grounding. The point is to choose a problem that actually matters and show that interpretability helps. I find this an exciting line of work because if we want interpretability to eventually be useful for making AGI safe, figuring out how to do things now seems like important practice.

- **Monitoring:** An extremely important problem in safety is that of monitoring: as a model runs, seeing whether a certain concept is present. The classic technique of probing is extremely cheap and is SOTA for cheap monitoring on frontier models for detecting misuse. What else can we do with probes?
    - How can probes be improved? Can we address cases where traditional probes work less well, like when information is spread across tokens or when there is a long context with lots of room for false positives? Attention head probes in Kantamneni et al are a good starting point.
- **Analyzing Chain of Thought (CoT):** Can you use or analyze a reasoning model's CoT to understand its behaviour, or otherwise achieve some practical uses? Can you steer the behaviour by resampling or editing the CoT?
    - The simplest way is to read or have an LLM read the CoT
    - Bogdan et al may be helpful if you need more powerful techniques
- **Other techniques:** Some other techniques that I think may have promising practical applications.

    - Conditional steering: applying a steering vector only if a probe fires. This lowers the side effects of steering a lot.
    - Training data attribution: A family of methods, including influence functions, to study which data points would have influenced a model to take a particular behavior more. The mathematical claims here are basically bullshit, but I think that being able to associate model behaviors with data points opens interesting use cases like debugging or removing noisy data points or filtering for the best data to finetune on
        - Warning: If you haven't played with TDA before, this may not be practical to work with in 20 hours
    - Abliteration: In refusal is mediated by a single direction my scholars cheaply jailbroke models by removing the refusal direction from the weights. How else can the idea of "abliteration" be applied?

Basic Science

I am generally excited about work that moves forward our understanding of key problems in interpretability. This is less of a focus of mine than it used to be, but I am still excited to supervise such work. Note that I am not particularly interested in work on toy models, algorithmic tasks, or interpretability during training unless there's a great pitch.

- **Understanding Reasoning Models:** What is actually happening inside reasoning models that produce long chains of thought? Can we intervene on their reasoning process?
    - It's surprisingly difficult to edit a model's chain of thoughts, since if you regenerate from that point onwards they will often immediately correct any errors introduced. What's up with this? Can we stop it? If you token force the next sentence, is that enough? Etc.
    - How do models trained with RL compare to those that are distilled from an RL-trained model? E.g., comparing QwQ to an R1 distill.
- **Steering Fine-tuning**: In Casademunt et al my scholars showed that you can control how a model generalises after fine-tuning, with zero change to the data or loss, by ablating concepts we don't want it to use. They used this to mostly fix emergent misalignment. This is really cool! Where else can we apply it?
- **Circuit finding:** What are tools like transcoders or attribution graphs actually telling us about circuits? Are they doing what we think they're doing? What are they missing?
    - How big a deal are the cross-layer connections in cross-layer transcoders? What are they really doing?
- **Basic science of SAEs**:
    - Why are some concepts learned, and not others? How is this affected by the data, SAE size, etc.
        - Scaling Monosemanticity had some awesome preliminary results here, but I've seen no follow-ups

    - How big an improvement *are* Matryoshka SAEs? Should we just switch to using them all the time, or do they have some flaws? What are they really doing?
- **Sanity checking superposition:**
    - Can we find the "true" direction corresponding to a concept? How could we tell if we've succeeded?
    - Can we find a compelling case study of concepts represented in superposition, that couldn't just be made up of a smaller set of orthogonal concepts? How confident can we be that superposition is really a thing?
    - Can we find examples of non-linear representations? (Note: it's insufficient to just find concepts that live in subspaces of greater than one dimension)

Novelty

- **New ideas**: For anyone feeling ambitious, I'm extremely impressed with any application showing ideas and applications of interpretability that are new to me or that I didn't expect to work
    - One of my favorite recent examples was in Casademunt et al, where my scholars showed it was possible to steer finetuning without changing the data.

FAQ (Extended)

FAQ (Extended)

- What's MATS?
    - In brief, it's a program that helps alignment researchers mentor junior researchers without needing to run their own mentoring program. See the MATS website for more information on the program as a whole. Other 10.0 mentors open soon
    - Each mentor has a lot of control over their stream and different streams will have very different experiences, I recommend thinking of it as many different small mentorship programs rather than one big one.
        i. (In particular, most applications are much quicker than mine and I'm the only one with an exploration phase)
    - You're encouraged to apply for as many mentors as you want to. You'll receive all of your offers for the research phase at the same time and can choose between them then.
- What should I expect from the exploration phase?
    - **Structure:** 3 week preparation phase: Feb 2 - Feb 20 and 2 week research sprint: Feb 23 - March 6
    - **Warning**: The exploration phase is *not* a structured course. I am not going to be telling you what to do. I view my role as a facilitator - I try to provide advice, good opportunities and resources, and help you all collaborate and learn from each other.
        i. But realistically, I'm largely running this on my own, there's over 30 of you, I can't really do one-on-one time.
        ii. Past scholars often comment that they are surprised by how unstructured and self-driven it was even, after receiving warnings like this.

- Preparation phase:
    i. Three weeks of education + skilling up, along the lines of this post.
    ii. The main things scholars spend their time on are self-driven learning, like doing coding tutorials and reading papers, and doing mini-projects, 0.5 to 5 day long research projects on their own or with a partner, And then, essentially, as warm-ups to the sprint.
    iii. There will be weekly group check-in calls, self-organised pair programming and collaboration, and you'll be able to ask each other and me questions over Slack
    iv. This is part-time, but some scholars choose to do it full-time. You will not be evaluated on the amount of time spent here, but I expect it to be an advantage in the sprint.
- Example exploration phase content - see last time's schedule:
    i. Talks, like my talk series on the big picture of mech interp and key research areas, or from authors of key papers, or on topics like how nnsight works (a popular mech interp library)
    ii. I do live research on a small mech interp problem while vibe coding and narrating my thought process
    iii. I live write the list of sprint problems and narrate my thought process for how I'm breaking down the space, why I think a problem is interesting, how I'd approach it, etc
    iv. Socials with other participants
        - Both remote, and in-person if there's enough people in the same place! We've had London, NYC, Cambridge (US) and more before
    v. **Note**: There's just one of me and 30+ of you! Unfortunately, this means I don't have capacity for 1-1s and mostly run group events.
- A two week full-time research sprint, where you pick an open problem and try to make progress on it.
    i. I'll mostly judge acceptance to the research phase based on your research sprint output
    ii. You'll do this in a team of two with another scholar (of your choice), though solo projects are possible.
    iii. Each team will give me a presentation at the end of the sprint, and I evaluate who to accept to the research phase.
        - You can request feedback on the project at the end of the presentation.

- What's changed from MATS 9.0?
    - Not much, the application process is largely unchanged
- What's changed from MATS 8.0?
    - I've increased the time limits a bit (now max 20 hours, +2 for executive summary).
    - I've shifted even more away from sparse autoencoder-related projects, and I'm comparatively more excited than I was about model biology and applied interpretability projects - details here
    - There's been a bunch of progress in reasoning model interpretability, and I have a bunch more ideas for projects there.

- Arthur Conmy will not be helping run the exploration phase. But he will have MATS scholars and I may refer scholars who don't make it to the research phase to him and other mentors
- In MATS 8.0, I experimented with encouraging scholars to do mini-projects, basically tiny research sprints, in the first three weeks (on their own or with pairs). This went extremely well and is now a core part of the program.
- LLMs have gotten substantially better over the last six months, and I give a bunch more detailed advice on the best ways to use them.

- What work have past scholars done?
  - Past scholar papers[8]:
    i. Do I Know This Entity? Knowledge Awareness and Hallucinations in Language Models (Javier Ferrando, Oscar Obeso, Senthooran Rajamanoharan, Neel Nanda, ICLR 2025 (Oral))
    ii. Inference-Time Decomposition of Activations (ITDA): A Scalable Approach to Interpreting Large Language Models (Patrick Leask, ICML 2025)
    iii. Scaling sparse feature circuit finding for in-context learning (Dmitrii Kharlapenko, Stepan Shabalin, ICML 2025)
    iv. Learning Multi-Level Features with Matryoshka Sparse Autoencoders (Bart Bussmann, Noa Nabeshima, ICML 2025)
    v. SAEBench: A Comprehensive Benchmark for Sparse Autoencoders in Language Model Interpretability (Adam Karvonen, Can Rager ICML 2025)
    vi. Are Sparse Autoencoders Useful? A Case Study in Sparse Probing (Subhash Kantamneni, Joshua Engels, ICML 2025)
    vii. Sparse Autoencoders Do Not Find Canonical Units of Analysis (Patrick Leask, Bart Bussmann, ICLR 2025)
    viii. Towards Principled Evaluations of Sparse Autoencoders for Interpretability and Control (Aleksandar Makelov, George Lange ICLR 2025)
    ix. Confidence Regulation Neurons in Language Models (Alessandro Stolfo, Ben Wu, NeurIPS 2024)
    x. Transcoders Find Interpretable LLM Feature Circuits (Jacob Dunefsky, Philippe Chlenski, NeurIPS 2024)
    xi. Refusal in Language Models Is Mediated by a Single Direction (Andy Arditi, Oscar Obeso, Aaquib Syed, NeurIPS 2024)
    xii. Explorations of Self-Repair in Language Models (Cody Rushing, ICML 2024)
    xiii. Is This the Subspace You Are Looking for? An Interpretability Illusion for Subspace Activation Patching (Aleksandar Makelov, Georg Lange, ICLR 2024)
    xiv. A Toy Model of Universality: Reverse Engineering How Networks Learn Group Operations (Bilal Chughtai, ICML)
    xv. Finding Neurons in a Haystack: Case Studies with Sparse Probing (Wes Gurnee, TMLR)

[8] Note that some papers required a 1-2 month extension on the program to properly finish off, and some of these papers were done in the extension as a second project

xvi. Interpreting Attention Layer Outputs with Sparse Autoencoders (Connor Kissane, Robert Krzyzanowski, Spotlight, Mechanistic Interpretability Workshop at ICML 2024)
xvii. Linear Representations of Sentiment in Large Language Models (Curt Tigges, Oskar Hollinsworth, BlackboxNLP)
xviii. Copy Suppression: Comprehensively Understanding an Attention Head (Callum McDougall, Arthur Conmy, Cody Rushing, BlackboxNLP)
xix. Training Dynamics of Contextual N-Grams in Language Models (Lucia Quirke, Lovis Heindrich)
xx. Thought Anchors: Which LLM Reasoning Steps Matter? (Paul C. Bogdan, Uzay Macar)
xxi. Understanding Reasoning in Thinking Language Models via Steering Vectors (Constantin Venhoff, Iván Arcuschin)
xxii. How Visual Representations Map to Language Feature Space in Multimodal LLMs (Constantin Venhoff, Ashkan Khakzar)
xxiii. Convergent Linear Representations of Emergent Misalignment (Anna Soligo, Edward Turner)
xxiv. Model Organisms for Emergent Misalignment (Edward Turner, Anna Soligo)
xxv. Towards eliciting latent knowledge from LLMs with mechanistic interpretability (Bartosz Cywiński, Emil Ryd)
xxvi. Overcoming Sparsity Artifacts in Crosscoders to Interpret Chat-Tuning (Julian Minder, Clément Dumas)
xxvii. BatchTopK Sparse Autoencoders (Bart Bussmann, Patrick Leask)
xxviii. Evaluating Sparse Autoencoders on Targeted Concept Erasure Tasks (Adam Karvonen, Can Rager)

- Papers I helped exploration-phase only scholars with:
  i. Internal states before wait modulate reasoning patterns (Dmitrii Troitskii, Koyena Pal - published at EMNLP)
  ii. Towards eliciting latent knowledge from LLMs with mechanistic interpretability (Bartosz Cywiński, Emil Ryd)
  iii. Simple Mechanistic Explanations for Out-Of-Context Reasoning (Atticus Wang, Oliver Clive-Griffin)
  iv. Reasoning-Finetuning Repurposes Latent Representations in Base Models (Jake Ward, Chuqiao Lin)
  v. RelP: Faithful and Efficient Circuit Discovery in Language Models via Relevance Patching (Farnoush Rezaei Jafari)

- Why aren't you giving many examples from MATS 9.0?
  - MATS 10.0 is starting earlier in the cycle than MATS 9.0, so the MATS 9.0 exploration phase ended in early Nov and the research phase runs from Jan to March 2026, so I don't have much data yet!
- What should I do if I want to do mech interp research but am not accepted to the program?
  - Sorry about that! There's a lot more good people who want to do mech interp research than I have capacity to mentor. The advice for how to do a good

application project is the same advice I'd give for doing your first mech interp project in general.

- If I get accepted to your program, is it a big deal if I start and then withdraw?
  - This is fine by me! In the past, about 1 in 6 people doing the training program have withdrawn for various personal reasons. This is fine from my perspective, if in doubt, please apply and just include a note in your application.
  - The point of the exploration phase is to be useful and educational to scholars, not adding value to me - if it's not helpful to you, or a better option comes up, please withdraw! I want you to make the best decision for you.
  - The main constraint is that acceptance to the research phase is based on your pair during the research sprint, so if you withdraw after the start of sprint it may disadvantage your partner
- Will there be a summer 2026 cohort?
  - I can't say for sure, but I plan to do one.
- Is it possible to do the research phase remotely?
  - Yes, this is not recommended (a lot of the value comes from being in Berkeley, having an in-person cohort, learning from each other, networking, etc, and it'll be harder to work with your partner if remote) but if you strongly prefer to be remote this is fine, and won't affect your chances of making the research phase.
  - There may be the option of doing the research phase in London
- There's a long gap between the training and research phase, can I do research in the gap?
  - If accepted to the research phase, you have absolutely no obligation to do research in the gap. But if you're really excited after the exploration phase, don't have other obligations, and want to start working on the research phase project during the gap, I'm excited to work together!
  - The MATS program will not have officially started, so you will be remote. I expect to be able to get you a grant for living expenses and compute.
- Can I do the exploration phase if I have a full-time job?
  - This is fine by me, and participants have done it before, but it's going to be harder for you
    i. You'll need to do the research sprint full-time, but some people have taken leave for it
    ii. The first 3 weeks for the preparation phase are "work whatever hours you want", so you're welcome to just do what you can in evenings and weekends.
  - Obviously, this is a pretty intense schedule, will put you at a disadvantage, and will not work with all employers. Sorry!
  - The research phase is full-time, and cannot be done part-time/on the side. Previous participants who had full-time jobs either took extended unpaid leave, or quit.
- Who owns the intellectual property?
  - Scholars own the IP of their work, not me or MATS.
  - Scholars are strongly encouraged to publish and open source their work, under a permissive license