Automated Interpretability-Driven Model Auditing and Control: A Research Agenda

Fazl Barez
University of Oxford

"Interpretability without human empowerment is incomplete."

Abstract

Current interpretability methods have made substantial progress in explaining model internals, but they rarely connect understanding to action. We propose a research agenda for automated interpretability-driven model auditing and control: a system where domain experts can query a model's behavior, receive explanations grounded in their expertise, and instruct targeted corrections—all without needing to understand how AI systems work internally. Our agenda comprises eight interrelated research questions forming a complete pipeline: from translating queries into testable hypotheses about model internals, to localizing capabilities in specific components, generating human-readable explanations, and performing surgical edits with verified outcomes. Our approach is distinguished by explicit hypothesis generation and testing rather than purely learned mappings from latent space to language, compositional concept graphs that capture how capabilities combine and interact, domain-grounded explanations that enable expert oversight, and human-in-the-loop intervention with predicted side effects. We evaluate the framework on three safety applications: chain-of-thought faithfulness verification, emergent capability prediction during training, and capability composition mapping. This paper outlines the full agenda, proposed experiments, baselines, and evaluation strategies, providing a roadmap for building interpretability tools that are not only scientifically interesting but practically useful for making AI systems safer and empowering humans to control them.

1 Introduction

Modern language models can do remarkable things, but we still don't understand how they work inside. More importantly, when they do something wrong, such as hallucinating a fact, exhibiting a bias, or responding to a jailbreak, we have limited tools, such as filters and guardrails, to fix the problem without retraining from scratch or running more RLHF.

Interpretability research has made real progress. We can now identify "knowledge neurons" that store specific facts (Dai et al., 2022), find attention heads responsible for particular behaviors like in-context learning (Olsson et al., 2022), and even edit individual facts in a model's memory (Wang et al., 2024a; Meng et al., 2022). Recent breakthroughs in 2024–2025 have scaled these techniques significantly: Anthropic's work on scaling monosemanticity has extracted millions of features from production models (Templeton et al., 2024). However, it remains challenging to decode latent representations into human-understandable concepts. Neurons are often polysemantic, and not all sparse autoencoder (SAE) latents correspond to interpretable and actionable features (Olah, 2025). These limitations may require rethinking the basis of computation in latent space (Hindupur et al., 2025), or alternatively motivate approaches that query and edit models using the most interpretable medium available, namely natural language from users (e.g., (Pan et al., 2024; Huang et al., 2025)). Moreover, recent research cautions that model-generated explanations, such as Chain-of-Thought (CoT), are not reliable proxies to explain what is happening (Barez et al., 2025), necessitating deeper mechanistic verification.

1.1 Our Vision

Consider a General Practitioner (GP) using an AI system to help with medical diagnosis. The patient presents symptoms, the GP enters the details, and the system suggests a possible condition. The GP, drawing on years of training, feels that something isn't right. They ask the AI system: *"Why do you think that?"*

This is where current AI systems may fail. They might offer a post-hoc rationalization (in which the model explains its previous answers), a generic confidence score, or simply repeat their conclusion with different words (Turpin et al., 2023; Barez et al., 2025; Arcuschin et al., 2025). What the GP actually needs is fundamentally different: a faithful account of what happened inside the model that led to this specific prediction, translated into terms they can reason about as a medical professional.

Core Objective: Faithful and Actionable Explanations

Our system's first objective is not to convince the GP that the model is correct. It is to **state what actually happened internally**—which features were activated, which associations fired, and which pathways led to this output—and to translate these internal workings into the language of the domain (in this case, medicine), presented in a way that enables the GP to make an informed decision.
The quality of an explanation is measured by a simple criterion: **does it enable the expert to identify when something is wrong?** A good explanation makes errors visible. If the model made a mistake, the explanation should give the GP enough insight to spot it.

Example 1: Catching a Demographic Assumption

Suppose our system's explanation reveals: *"The system associated the patient's reported fatigue with iron deficiency, which is common in this age group and correlates strongly with the other symptoms presented."*
The GP, who knows this particular patient, realizes something: this patient is from a narrow sub-demographic where iron deficiency presents differently, or where another condition is far more likely. The standard assumption doesn't hold here.
The GP responds *"This patient is from [sub-demographic]. The iron deficiency assumption does not apply; they have [specific characteristic] that changes the picture."* The GP may request model intervention to fix this issue, where our system will automatically implement it and evaluate the changes.

Example 2: Worst-Case Diagnosis

Consider an example in which a patient reports that they have a cough and feel out of breath, and just took a long flight. The target model suggests that the patient has a lung infection and should take antibiotics.
The GP then asks the system to explain the target model's decision. The system reports that the model placed disproportionate emphasis on the causal association between "coughing" and highly activated "lung infection" features, while downplaying the association between the "long flight" factor and a "jet lag" diagnosis (e.g., SAE features related to lung infection are strongly activated, whereas features of air travel and jet lag are completely inactive). This analysis suggests that the target model discounted crucial contextual factors and consequently favored a worst-case diagnosis over a more rational and probable alternative.
Upon request, our system can intervene by steering relevant SAE features to improve the target model's responses to similar queries in the future.

The agent's role: Tool selection and intervention. Now the system must interpret the model and fix discovered errors. But how? There are many interpretability and editing methods available, including feature steering, concept erasure, activation patching, weight editing, and more. Each operates at a different level of abstraction and has different side effects. We need an LLM-based **interpretability agent** to handle the selection of operations (Yao et al., 2023; Wang et al., 2024b). Specifically, the agent:

1. **Interprets the feedback**: Translates the GP's natural language correction into a hypothesis about what internal components might encode the faulty assumption and concrete tasks about what the system should do.
2. **Localizes the error**: Uses tools like Query Circuits and Transcoders to identify which specific features, neurons, or circuits were responsible for the incorrect association.
3. **Selects the intervention**: Evaluates multiple editing methods in parallel, running benchmarks to predict side effects, and chooses the approach that corrects the error with minimal damage.
4. **Applies a permanent fix**: The change is not temporary, the model is genuinely modified, so it won't make this specific mistake again.
5. **Verifies and explains**: Reports back what was changed, validates the generalization of the changes, and evaluates the side effects (the model should not permanently exclude iron deficiency; it may need to be considered in other cases).

Permanent learning through interaction. A crucial feature: **changes are permanent**. Each interaction in which a domain expert corrects the model improves its performance on this and related cases. The GP who corrects the iron deficiency assumption has improved the model for all future patients from that sub-demographic. Over time, as more experts interact with the system, it accumulates domain knowledge that was never in its training data. This approach may also inform continual learning systems, where safety evaluations face similar challenges from models that evolve post-deployment (Barez, 2026).

This is fundamentally different from prompt engineering or in-context learning, which are temporary. It's also different from traditional fine-tuning, which requires curated datasets. Here, the model improves through natural dialogue with experts, one correction at a time, with continual effects.

Summary: The two core functions. Our system serves two interconnected functions:

1. **Explanation for decision support**: Generate faithful, domain-grounded explanations that enable experts to evaluate model predictions and identify errors, as measured by whether experts can catch mistakes and make faster, more accurate decisions.
2. **Agent-mediated correction**: When experts identify errors, an agent determines the optimal interpretability tool and abstraction level for intervention, applies permanent corrections with minimal side effects, and improves the model for future use.

In summary, our vision is a system that gets better the more domain experts use it, not through retraining, but through interpretability-driven dialogue.

1.2 What makes this different

Recent work has explored closely related directions. LatentQA (Pan et al., 2024) is one of the first approaches to decode a target model's latent states into natural language using an LLM-based decoder. Activation Oracles (AO) (Karvonen et al., 2025a) extend LatentQA's training paradigm to support a broader range of user queries over model activations. Predictive Concept Decoders (PCDs) (Huang et al., 2025) further constrain the decoder to decode over a learned set of human-readable concepts, thereby enhancing interpretability.

These are valuable contributions, but they address a different problem than ours. They mainly focus on *extracting information* from models. We focus on the next next—using the extracted information to continuously improve the models with external human-in-the-loop, through dialogues with the model.

Several features distinguish our approach:

Faithfulness, not persuasion. The goal of our explanations is not to convince users that the model is correct. It is to faithfully report what happened internally, such as which features were activated, which associations fired, and which pathways produced this output. An explanation that makes users trust a wrong answer is a failure, not a success. Accordingly, we measure explanation quality by whether it enables experts to catch errors, not by user satisfaction.

From a user-centered perspective, the granularity of explanation should be adaptable to expert needs. Users may prefer to begin with a high-level description and progressively request more detailed or mechanistic explanations as needed. The appropriate level of granularity may therefore correlate with the amount of mechanistic detail required for effective diagnosis and intervention.

Actionable for domain experts. Generic interpretability outputs like "neuron 4567 activated" or "tokens related to medicine" don't help a doctor evaluate a diagnosis. Our explanations are grounded in the user's domain, such as a medical prediction explained in terms of biological mechanisms, symptoms, and clinical reasoning patterns that a GP can evaluate against their expertise.

Permanent learning through correction. When a domain expert identifies an error and provides feedback, the model is *permanently* modified. This is different from prompt engineering (temporary), not in-context learning (session-limited), and not traditional finetuning (requires curated datasets). The model improves through natural dialogue with experts, one correction at a time. A GP who corrects a demographic assumption has improved the model for all future similar cases. In particular, a form of intervention could be to apply edits directly on the weights that would generalise to test cases/prompts not used in the crafting of the interventions, allowing OOD generalisation. Contrary to usual finetuning methods, this is still more efficient and cost-effective as we use fewer examples to craft the intervention, and we locate low-dimensional subspaces within the weight space to intervene on.

Agent-mediated tool selection. There is no single "best" interpretability or editing method, as different corrections require different tools at different levels of abstraction. Our system includes an agent that evaluates multiple intervention methods in parallel, predicting side effects and selecting the approach that fixes the error with minimal damage. The agent learns in-context over time which tools work best for which types of corrections.

Structured representations. Concepts in model representations can be represented as complex manifolds (Pearce et al., 2025). They combine and interact in complicated ways (Hindupur et al., 2025). We build compositional concept graphs that capture these relationships to enable more precise localization of errors, more effective intervention, and more accurate prediction of side effects. For example, if we find that concepts are represented along a hypersphere, methods like angular steering (Vu and Nguyen, 2025) and orthogonal finetuning (Qiu et al., 2025) may work well.

Training-time monitoring. Existing approaches analyze models after training is complete. We extend interpretability to the training process itself, monitoring for emergent capabilities before they fully manifest and tracking how simpler capabilities compose into complex behaviors.

1.3 Research Agenda Contributions

- A vision for interpretability as **expert-in-the-loop model improvement**: domain experts identify errors through faithful explanations, and the system permanently corrects them.
- Eight research questions (RQ1–RQ8) forming a pipeline from natural language query to verified, permanent intervention.
- An **interpretability agent** that selects optimal tools and abstraction levels for each correction.
- **Explanation quality metrics** based on actionability: error detection rate and decision efficiency.
- Novel safety applications: chain-of-thought faithfulness verification, emergent capability prediction, and capability composition mapping.
- Proposed experiments, baselines, and evaluation strategies for each component.

This is a proposed research agenda, not a finished system. However, we believe this framing, interpretability in service of expert-driven permanent improvement, points toward tools that are practically useful, especially for safety applications. We also hope to make fast progress toward building such end-to-end system.

Throughout this paper, we use medical diagnosis as an illustrative example for accessibility. Our primary applications, however, are in AI safety, where the stakes of undetected errors and the need for expert intervention are equally pressing.

2 Related Work

2.1 Automated Interpretability

The idea of using AI systems to interpret other AI systems has gained momentum recently. Foote et al. (2023) introduced Neuron2Graph, which builds interpretable graph representations of neuron behavior. Bills et al. (2023) showed that GPT-4 can generate natural language explanations for neurons in GPT-2. Shaham et al. (2024) introduced the Multimodal Automated Interpretability Agent (MAIA).

Most recently, Transluce Research has advanced this field with Predictive Concept Decoders (PCDs) and LatentQA (Pan et al., 2024), demonstrating that models can extract latent representations of users. Furthermore, Li et al. (2025) provided evidence for the Privileged Access Hypothesis, showing that models trained to explain their own computations outperform external explainer models.

However, reliance on model self-explanation has limits. Prior work has shown that chain-of-thought reasoning is often unfaithful to internal computation; Barez et al. (2025) extends this to argue that CoT should not be treated as explainability at all, motivating causal verification methods like those we propose.

2.2 Mechanistic Interpretability

Mechanistic interpretability aims to reverse-engineer the algorithms implemented by neural networks. Some of the landmark works includes the discovery of induction heads (Olsson et al., 2022) and the IOI circuit (Wang et al., 2023).

Significant progress in 2024–2025 has refined our ability to decompose these networks at scale:

- **Scaling monosemanticity:** Anthropic successfully scaled Sparse Autoencoders (SAEs) to Claude 3 Sonnet (Templeton et al., 2024).
- **Transcoders:** Dunefsky et al. (2024) proposed Transcoders to replace dense MLP layers with sparse, interpretable approximations.
- **Crosscoders:** Lindsey et al. (2024) introduced Sparse Crosscoders, a variant of SAEs/transcoders that map activations across multiple layers and models into a shared feature space, enabling interpretable, cross-layer and cross-model comparisons.
- **Query circuits:** Wu and Barez (2025) introduced Query Circuits, a method to trace the specific information flow for a single input query, bridging the gap between global circuits and local explanations.

- **Scaling circuit finding:** Kharlapenko et al. (2025) successfully scaled sparse feature circuit finding to In-Context Learning (ICL) in Gemma models.

2.3 Model Editing and Steering

If we can identify where knowledge or capabilities are stored, can we change them? ROME (Meng et al., 2022) and MEMIT (Meng et al., 2023) showed this is possible for factual associations.

More recent intervention techniques focus on activation space and precise erasure:

- **Feature steering:** Anthropic's "Golden Gate Claude" (Anthropic, 2024) demonstrated clamping features to steer behavior.
- **Beyond linear steering:** Oozeer et al. (2025) introduced Unified Multi-Attribute Control, allowing for more complex, multi-dimensional steering beyond simple vectors.
- **Representation finetuning:** Wu et al. (2024) introduced ReFT to leverage the rich semantic structure of the representation space to learn efficient low-dimensional interventions.
- **Concept erasure:** Gur-Arieh et al. (2025) proposed Precise In-Parameter Concept Erasure (PISCES), a method to surgically remove concepts from model weights with high specificity.

2.4 Scaling-Focused Interpretability

A recent line of work advocates for a "bitter lesson" approach to interpretability: rather than hand-designing analysis methods, scale up learned systems and let them discover what matters (Huang et al., 2025; Karvonen et al., 2025b; Choi et al., 2025).

Predictive Concept Decoders (PCDs) (Huang et al., 2025) train an encoder-decoder system on $\sim$100M web tokens. The encoder compresses activations to $k = 16$ sparse concepts; the decoder predicts model behavior from only these concepts. The sparse bottleneck provides *auditability*: responses can be traced back to specific SAE-like features. Activation Oracles (AO) (Karvonen et al., 2025b) take a similar approach, scaling LatentQA (Pan et al., 2024) training to $\sim$1M examples with improved OOD generalization, including to fine-tuned variants of the base model.

These are impressive engineering achievements, and we share the view that interpretability must scale. However, we differ on a fundamental question: *what is the ground truth for interpretability?*

Scaling data vs. scaling feedback. The PCD and AO approaches treat interpretability as a prediction problem: train on enough data, and the system learns to answer questions accurately. Quality is measured by downstream task performance, which improves with scale.

Our approach treats interpretability as an *intervention* problem: the ground truth is defined as whether explanations enable experts to identify errors, and whether interventions based on localization actually fix problems. We scale not by adding training data, but by accumulating expert corrections; in other words, each interaction improves the model and provides a signal about what works. Additionally, by fixing representations directly, the system allows for more efficient learning, scaling better than fine-tuning (Wu et al., 2024).

This crucial difference determines what is optimized for: A system optimized for prediction accuracy might produce explanations that sound correct but don't help experts catch errors. A system optimized for actionability to ensure experts can act on what they learn, with the tradeoff of sacrificing some generality.

Observation vs. intervention. PCD and AO systems are read-only: they extract information from models, but don't change them. Our system is built around intervention from the start. The explanation exists to enable correction; the localization exists to guide intervention; the verification exists to confirm the fix worked. This forms our core design principle.

Auditability vs. actionability. PCDs achieve auditability through the concept bottleneck: any decoder output can be traced to $k = 16$ concepts. This is valuable for understanding what the system is doing. In contrast, we aim more for *actionability*: explanations that enable experts to identify errors and request corrections. An explanation that cannot help a doctor catch a diagnostic error has failed in the task we care about. Actionability requires more than traceability; it requires domain grounding (explaining reasoning in terms the expert understands), an appropriate level of abstraction, and sufficient detail to distinguish correct from incorrect reasoning.

Layer sweeping vs. specialized multi-layer localization. PCDs and LatentQA rely on sweeping across layers to identify latent representations that can be translated into natural language and edited, resulting in input-space explosion. In contrast, we introduce automated localization steps that identify critical internal signals at different levels of abstraction and encode these signals into a shared representation layer, making multi-layer reading and editing tractable. This shared representation allows the decoder to jointly reason over localized internal concepts without requiring separate decoding or editing mechanisms for each layer.

Flat concepts vs. compositional structure. PCDs represent each prediction through $k = 16$ independent concepts. But model computations are compositional: concept A feeds into concept B, which together with concept C, produces behavior X. A flat list loses this relational structure.

Our compositional concept graphs capture how concepts combine and interact. This representationn enables more precise localization (i.e., which concept interactions or connections are responsible?), better intervention prediction (i.e., what will propagate across the relational graph if we change a concept?), and richer explanations (i.e., not just "which concepts" but "how they combine").

Complementary approaches. We view approaches based on LatentQA, AO, and PCD as complementary rather than competing, as they provide scalable, learned components that serve as building blocks of our pipeline. For example, PCD's encoder could help identify candidate concepts, and the decoder could help generate initial explanations. Our contribution is the intervention loop that converts understanding into action, together with leveraging expert feedback as the ground truth about what interventions are effective in practice.

The question is not "learn vs. engineer" but "learn toward what objective?" We argue for learning toward actionability—explanations that enable experts to catch errors and corrections that actually fix problems—rather than learning toward prediction accuracy on held-out questions.

2.5 Gaps Our System Addresses

Current interpretability research, despite significant advances, operates in a mode that fundamentally limits its usage. We identify several interconnected gaps:

Observation without action. The vast majority of interpretability work is observational: analyze a model, report findings, publish. But observation alone provides a limited signal. We can describe what we see, but we cannot confirm whether our descriptions are correct, complete, and helpful. We miss the richest source of information: what happens when we intervene. Hence, we believe signals from the end-to-end system would provide richer information.

No ground truth for interpretability claims. How do we know an interpretation is correct? Currently, we rely on face validity ("this explanation seems reasonable"), consistency checks, and limited ablation studies. But there is no systematic ground truth. Our approach moves towards creating one: if an explanation enables the domain expert to identify an error they couldn't see before, the explanation captured something real. If an intervention based on our localization fixes the problem, the localization was correct. *Intervention outcomes are ground truth for interpretability quality.*

Missing the expert feedback loop. Current interpretability is by ML researchers, for ML researchers. Domain experts, such as doctors, lawyers, and engineers who actually use AI systems and know when they fail, are excluded from the process. Yet they represent the richest source of feedback: they know things about their domains that no training set captures. A GP who says "this assumption doesn't hold for this sub-demographic" provides information that could never emerge from analyzing activations alone.

No cumulative improvement. Each interpretability study stands alone. Findings don't accumulate into better models. Corrections don't persist. There's no learning curve. In our vision, every interaction where an expert corrects the model makes it permanently better. The system improves through use, building on each correction.

Tool selection without feedback. There are now many interpretability methods: SAEs, transcoders, circuit analysis, probing, activation patching, and more. Researchers choose among them based on intuition or familiarity, with little systematic evidence about which works best for which problems. Our agent-mediated approach learns this over time: by running parallel evaluations on each correction and observing outcomes, the system discovers which tools work for which types of errors.

Explanations optimized for the wrong audience. Current interpretability explanations are written for ML researchers. But the people who need to evaluate AI predictions, such as doctors, judges, and safety engineers, speak different languages. An explanation in terms of "attention heads" and "feature activations" is useless to a GP evaluating a diagnosis. We need explanations that translate internal mechanisms into domain-relevant terms that enable expert judgment.

The end-to-end signal. Perhaps most fundamentally, interpretability research has operated without end-to-end signals. In isolation, we develop methods, evaluate them on proxy metrics, and hope they'll be useful in practice. By building a complete system, from query to explanation to intervention to verification, we create an environment where we can finally measure what matters: *does this actually help experts catch errors and improve models?*

This is analogous to the difference between studying game theory in the abstract versus deploying an RL agent that learns from actual gameplay. The agent that acts in the environment receives rewards and learns things conditioned on real-world noises where pure analysis

might not handle. Our system, by closing the loop from interpretation to intervention, creates the conditions for interpretability research to finally get this kind of rich, grounded feedback.

Interpretability without human empowerment is incomplete. Methods must be judged by whether they give humans the capability to intervene and correct, not merely to inspect.

3 Automated Interpretability System Overview

Our system connects a domain expert to an AI model with an interpretability layer that enables understanding and control. The expert never needs to understand neural networks, attention heads, or activation patterns—they interact entirely in natural language within their domain.

Architecture. The system comprises three main components:

System Components

Target model: The AI model being explained, audited, and intervened on (e.g., a medical AI assistant). This is the model the expert actually uses, and the model we ultimately modify when corrections are needed.
Investigator model: An LLM for activation-to-text decoding, trained in the style of LatentQA (Pan et al., 2024) to read the target model's internal activations and answer questions about them (e.g., "What factors influenced this decision?", "Is the model using the patient's hair color?") in natural language.
Agent: Translates the expert's natural language requests into specific operations and calls corresponding tools, including the investigator model. For example, when the expert says "Remove the influence of hair color," the agent determines: At what layer of abstraction should we intervene? Which intervention tool should we use? What are the side effects? The agent orchestrates all active components to plan and fulfill the expert's intent.

Why investigator model and what's its role? The investigator model is central to our approach. Following the LatentQA paradigm, we train the investigator on datasets where we *know* the ground truth: we create prompts with specific properties (e.g., "respond as Harry Potter"), collect the target model's activations, and train the investigator to correctly answer questions about those properties.

Incorporating the investigator provides two key benefits:

1. **Calibrated trust in explanations.** Because we trained the investigator to detect specific properties, we know what it can reliably identify. If the investigator says "the model is considering hair color," we can trust this claim to the extent that hair-color-related concepts were in the training distribution. This is unlike post-hoc explanations, where we have no ground truth for faithfulness.
2. **Differentiable intervention.** The investigator doesn't merely read activations; it specifies a differentiable loss over them. If we want the target model to *not* consider hair color, we can express this as a QA pair ("Is hair color influencing this decision?" $\rightarrow$ "No") and backpropagate through the investigator to modify the target model's weights. This enables permanent corrections guided by natural language specifications.

Why agent and what's its role? The agent serves as the translator between human intent and system operations. This is critical since even a single short query may require several turns of tool usage.

When the expert speaks, the agent must decide two things:

1. **Abstraction level and mechanistic hypothesis:** Which internals to investigate? Should we intervene at the neuron level? Feature level? Layer level? More fine-grained interventions are more surgical but may miss distributed representations. Coarser interventions, such as contrastive activation addition (CAA) (Rimsky et al., 2024), are more robust but may have more side effects.
2. **Tool selection:** At each step, the agent may decide on a tool to use. Most components in the system can be packaged as tools. Some examples are:
    - **Activation decoder tool.** Inputs are (1) the target model's layer representations when processing an input and (2) a question about this representation. The output is the answer to the question.
    - **Localisation tools.** Tools to localise different levels of abstractions (e.g., layers, neurons, and circuits) to decode and intervene on. Examples include SAE neuron descriptions (Huben et al., 2024) and circuit discovery (Wu and Barez, 2025; Ameisen et al., 2025).
    - **Intervention tools.** Take CAA as an example. The input is a concept of interest to steer. The output is an activation vector to hook on and steer the model internals towards generating texts with that concept.

- **Utility tools.** Tools to help the agent summarize and organize current information, such as the neuron clustering tool, description summarization tool, etc.

Different problems call for different tools. We may also incorporate hand-crafted rules. For example, we can remind the agent via few-shot exemplars to always invoke the side-effect estimation tool after applying an intervention tool. The design of the toolset and the logical relationships among tools are crucial to the agent's performance and capabilities.

The agent explores different decisions and converges to a good one over time by interacting with the environment. For example, each use of an intervention tool gains feedback: did it work? What were the side effects? This signal improves future tool selection and abstraction level choices.

Information flow. Consider the medical example:

1. **Prediction:** The GP queries the target model: "Should this patient have surgery?" The target model generates activations and processes them to produces a prediction.
2. **Explanation request:** The GP asks: "What biological factors were considered?" The agent routes this to the investigator tool, which reads the target model's activations and produces a domain-grounded answer: "The decision is driven primarily by cardiovascular condition and age. Hair color and BMI were also taken into account."
3. **Correction request:** The GP says: "Hair color should not matter. Remove its influence." The agent translates this into:
   - A QA specification for the investigator: ("Is hair color influencing the decision?" $\rightarrow$ "No")
   - A selection of intervention method (e.g., weight modification via backpropagation vs. concept erasure vs. activation steering)
   - A layer/component target based on where hair-color-related features were localized
4. **Intervention tool calling:** Using the investigator's differentiable loss, we modify the target model's weights so that hair color no longer influences surgery recommendations. The change is permanent.
5. **Verification tool calling:** Call the investigator to re-examines the target model's activations on test cases. It should now answer "No" to "Is hair color influencing this decision?" confirming the correction worked.

Training the investigator. Following LatentQA, we train the investigator by:

1. **Generating controlled stimuli:** Create inputs with known properties. For example, prepend system prompts that instruct specific behaviors, inject specific biases, or include particular types of information.
2. **Collecting activations:** Run these inputs through the target model and collect internal activations at selected layers.
3. **Creating QA pairs:** Generate question-answer pairs about the known properties. "What persona is the model adopting?" $\rightarrow$ "Harry Potter." "Is the model using demographic information?" $\rightarrow$ "Yes, specifically age and gender."
4. **Finetuning the decoder:** Train the investigator to produce correct answers given the activations. The investigator learns to read the target model's internal state.

This training approach means the investigator's capabilities are known and bounded. A more diverse and scaled QA pair generation will make a more capable investigator. That said, we can always characterize what it can reliably detect (concepts in training) versus what it might miss (out-of-distribution phenomena). This is crucial for calibrated trust.

Permanent vs. temporary corrections. A key design choice: corrections are permanent. For example, when we backpropagate through the investigator to modify the target model, we change its weights. This is different from temporary prompt engineering, which is not suitable for repeated changes and can be easily overridden.

The advantage: each expert correction makes the model permanently better. The GP who fixes the hair-color bias has improved the model for all future patients. Over many interactions, the model accumulates domain knowledge that wasn't in its original training.

The risk: permanent changes can compound. We mitigate this through verification (RQ6) and side-effect monitoring (RQ7).

A reusable record of QA pairs. Having the system generate a permanent record of QA pairs documents faults in the base model. This newly created list is a permanent stand-alone asset that can be published and reused as training data in future models. If one has 20 experts independently improving their own instance of the model, their 20 sets of QA pairs can later be combined and reused.

4 Research Questions

Our agenda comprises eight research questions that map directly to the pipeline described in our vision. We organize them into four themes:

Understanding (RQ1–3): What is the model actually doing internally?

Translation (RQ4): How do we explain this to domain experts?

Correction (RQ5–6): How do we fix errors and verify the fix?

Safety (RQ7–8): How do we handle hallucinations and ensure robustness?

Pipeline Stage	Research Question
User asks "Why?"	RQ1: Hypothesis Generation
System investigates	RQ2: Localization
System validates understanding	RQ3: Hypothesis Testing
System explains	RQ4: Domain-Grounded Explanation
User requests correction	RQ5: Targeted Intervention
System verifies	RQ6: Verification
Ongoing monitoring	RQ7: Hallucination Detection
Long-term safety	RQ8: Robustness

4.1 RQ1: Hypothesis Generation

The Problem

The GP asks: "Why did you recommend against surgery?" Internally, the model's prediction resulted from a complex interaction of complex components. We cannot enumerate them all. We need to generate *hypotheses* about which high-level mechanisms might be informative or responsible, so we know where to look.
However, our initial guesses may be wrong. When we observe that neuron X activates for "patient age 72." Is X an age detector? Or does it detect "risk factors," "numbers above 50," or "elderly patient concepts"?
This is the **functional form problem**. If we misidentify the true function, our interventions will have unintended consequences. If we think X detects "age" but it actually detects "risk factors," removing X to address age bias will also remove legitimate risk assessment.

Our approach. We treat hypothesis generation as a search for the correct abstraction:

1. **Parse the query:** Extract what the user wants to understand. "Why recommend against surgery?" $\rightarrow$ understand factors influencing surgery recommendation.
2. **Generate candidate hypotheses:** Propose multiple mechanisms at different abstraction levels:
   - H_1: Patient age directly suppressed surgery recommendation
   - H_2: Age activated a "risk" circuit that suppressed surgery
   - H_3: Age triggered retrieval of "elderly surgery outcomes" from training data
   - H_4: Multiple factors (age + symptoms + history) combined
3. **Design discriminating tests:** For each hypothesis pair, identify what evidence would distinguish them. H_1 vs H_2: if we find a general "risk" circuit that also responds to other risk factors, favor H_2.
4. **Prioritize:** Rank hypotheses by prior probability and testability. Start with the most likely and easiest to test.

Why this matters. Without good hypotheses, we search blindly. With wrong hypotheses, we misunderstand the model and make harmful interventions. The quality of everything downstream—localization, explanation, correction—depends on generating the right hypotheses.

Evaluation. We evaluate using the following criteria:

1. **Coverage:** Do our hypotheses include the true mechanism? Measured on synthetic setups with known ground truth.
2. **Efficiency:** How many hypotheses must we test before finding the right one?
3. **Abstraction accuracy:** Do we identify the correct level of abstraction (age vs. risk vs. numbers)?

4.2 RQ2: Localization

The Problem

We have hypotheses. Now we need to find the actual components responsible. Which neurons, attention heads, or circuits implement the mechanism we're investigating?
This must be specific to the user's query. We don't want "neurons that generally relate to age"—we want "the components that caused *this specific prediction* to depend on age."

Our approach. We combine multiple localization techniques, selected based on the hypothesis type:

- **Query-specific causal tracing:** Use Query Circuits (Wu and Barez, 2025) or Circuit Tracing (Ameisen et al., 2025) to trace information flow for this specific input. Which components were active? Which carried age information toward the output?
- **Feature decomposition:** Use Sparse Autoencoders (Templeton et al., 2024) or Transcoders (Dunefsky et al., 2024) to decompose activations into interpretable features. Which features were activated for this input?
- **Causal verification:** Don't just find correlated components—verify they're causal. Ablate candidate components and check if the prediction changes. Use activation patching to confirm information flow.

Tool selection. Different localization tools work better for different problems:

Hypothesis Type	Best Tools
Single feature ("age")	SAEs, probing
Information flow (circuit)	Query Circuits, Circuit Tracing
Attention layer	attention score analysis
MLP computation	Transcoders

The agent learns in-context which tools work best through experience. This framework will facilitate the easy integration of new tools as they become available.

Evaluation.

- **Precision:** What fraction of identified components are actually involved? (Measured by ablation impact)
- **Recall:** What fraction of involved components do we identify? (Measured against ground truth in synthetic setups)
- **Specificity:** Are we finding query-specific components, not just generally relevant ones?

4.3 RQ3: Hypothesis Testing

The Problem

We've localized components. But do we understand them correctly? If we think neuron X detects "patient age" but it actually detects "numerical risk factors," our explanation will be wrong and our intervention will have unintended effects.
We need to test our hypotheses rigorously to find the **true functional form**.

Our approach. We automate the scientific method:

1. **Generate discriminating examples:** Given competing hypotheses ("age detector" vs. "risk factor detector"), generate inputs that distinguish them:
   - Young patient with many risk factors (age low, risk high)
   - Elderly patient with few risk factors (age high, risk low)
2. **Run experiments:** Test how the component responds to discriminating examples. Does it activate for all elderly patients, or only high-risk ones?
3. **Eliminate and refine:** Remove hypotheses inconsistent with observations. Refine remaining hypotheses based on patterns.
4. **Converge:** Stop when we have a hypothesis that correctly predicts component behavior across diverse inputs.

Handling compositional functions. Some components have compositional functions: "activates for elderly AND high-risk patients." We detect these by finding cases where single-concept hypotheses fail but conjunctions succeed.

Why this matters. The true functional form determines:

- What explanation we give the expert
- What side effects an intervention will have
- Whether we can safely correct the error

If we get it wrong, everything downstream fails.

Evaluation.

- **Functional form accuracy:** On synthetic setups with known functions, do we recover the correct form?
- **Intervention prediction:** Given our identified function, can we accurately predict what happens when we intervene?
- **Convergence efficiency:** How many tests until we converge?

4.4 RQ4: Domain-Grounded Explanation

The Problem

We've identified that "neurons 4523 and 4891 in layer 15, connected through attention head 12-7" are responsible. This information is not helpful to the GP. We need to translate our findings into an explanation that the expert can evaluate, such as:

> "*The model identified the patient's age (72) as a primary factor. This activated age-risk associations learned during training, which elevated perceived surgery risk. The clinical indicators (strong vitals, localized condition) were processed but assigned lower weight than the age factor. The recommendation against surgery was driven primarily by the age-risk pathway, not by analysis of this patient's specific clinical presentation.*"

This explanation lets the GP evaluate: "The model is over-weighting age relative to the clinical picture. That's wrong for this patient."

Our approach.

1. **Component → concept:** Map each localized component to a human concept. Use activation analysis, feature dictionaries, and automated labeling.
2. **Concept → domain term:** Translate generic concepts to domain language. "Numerical magnitude feature" → "patient age." This requires domain ontologies or expert feedback.
3. **Causal narrative:** Assemble concepts into a story that explains the computational flow. Not just "age was involved" but "age activated risk, which suppressed surgery recommendation."
4. **Influence ranking:** Present factors in order of causal influence. The expert sees what mattered most first.

Quality metric: Actionability. We measure explanation quality by whether it enables experts to identify errors:

- **Error detection rate:** Given predictions with known errors, do experts catch them when provided our explanation? Compare to: no explanation, model self-explanation (CoT), generic saliency maps.
- **Detection speed:** How quickly do experts identify errors?
- **False positive rate:** Do explanations cause experts to flag correct predictions as wrong?

An explanation that convinces experts to trust wrong predictions is a failure, not a success.

Evaluation.

- **Error detection rate:** Primary metric. See above.
- **Expert comprehension:** Do experts correctly understand what the explanation claims? (Test via comprehension questions)
- **Faithfulness:** Does the explanation accurately reflect the internal computation? Verified by intervention—if we modify what the explanation claims is important, does the output change as predicted? We additionally test *causal introspection* by injecting unrelated concept vectors into intermediate activations and evaluating whether the explanation correctly identifies the injected internal concept Lindsey (2025).

4.5 RQ5: Targeted Intervention

The Problem

The GP says: "The model is over-weighting age. This patient's clinical presentation clearly indicates surgery is appropriate. Fix the age bias."
Now we must act. But how? There are many possible interventions:

- Find the age-related neurons and suppress them
- Reduce their connection weights to the decision circuit
- Erase the "age → risk" association using PISCES (Gur-Arieh et al., 2025)
- Apply steering to reduce age influence (Oozeer et al., 2025)
- Fine-tune on examples where age shouldn't determine outcome

Each approach entails a different trade-off. Complete ablation may remove legitimate age-related reasoning, while fine-tuning can introduce broad and unintended side effects. The clarity of user intent may also influence the choice of intervention tools and their parameters—for example, how much weight for age factor constitutes "over-weighting"? Is it appropriate for the system to simply ablate all age-related neurons within this context? Such ambiguities should be addressed by allowing the agent to request clarification when needed.

Our approach.

1. **Interpret the correction:** Parse what the expert wants. "Fix the age bias" → reduce influence of age on surgery recommendation while preserving legitimate clinical reasoning.
2. **Generate candidate interventions:** Based on the localized components and the correction type, propose interventions at different abstraction levels.
3. **Predict outcomes:** For each candidate, estimate:
   - Will it fix the problem?
   - What side effects will it have?
   - Will it generalize or only work for this case?
4. **Select and present:** Choose the intervention with the Pareto front. Present the choice and predicted outcomes to the user for approval.
5. **Apply permanently:** Execute the intervention. The model is now modified.

Intervention selection via parallel evaluation. The agent doesn't guess which intervention is best—it instead:

1. Run each candidate intervention on a held-out evaluation set
2. Measure: fix rate, side effects on related capabilities, side effects on unrelated capabilities
3. Select the intervention that fixes the problem with minimal collateral damage
4. Log the outcome to improve future selection

Over time, the agent learns in-context which tools work best for which correction types.

Evaluation.

- **Fix rate:** Does the intervention resolve the problem in held-out cases?
- **Generalization:** Does it fix related issues (other cases of age bias) or only this specific instance?
- **Side effect magnitude:** How much do unrelated capabilities change?
- **Selection quality:** Does the agent choose interventions that perform well in the fix rate vs. side effect tradeoff?

4.6 RQ6: Verification

The Problem

We've applied an intervention. Did it work? Did we break anything? Before declaring success, we need systematic verification. This is where we get ground truth about our interpretability. If our localization was correct and our intervention was well-targeted, verification should pass. If verification fails, something in our pipeline is wrong.

Our approach. We run a verification suite after each intervention:

- **Targeted verification:** Test the specific behavior we changed. For "reduce age bias," test on diverse patients—the recommendation should no longer overweigh age.
- **Side effect scan:** Test on broad capability benchmarks. Did general medical reasoning degrade? Did other behaviors change unexpectedly?
- **Whack-a-mole check:** Did the removed behavior re-emerge in a different form? If we reduced "age" influence, does the model now over-weight "years since birth" or "date of birth"? Probe for proxy behaviors. Evaluate performance on a general benchmark and assess perplexity.
- **Consistency check:** Is the model still internally consistent? Do its explanations still match its behavior? Interventions can sometimes create inconsistencies.

Failure modes. If verification fails:

- **Target not fixed:** Our localization may have been incomplete. Return to RQ2 and search for additional components.
- **Side effects detected:** Our functional form understanding (RQ3) may have been wrong. The component we modified did more than we thought.
- **Whack-a-mole:** The model found another way to implement the behavior. Need a broader intervention.

Each failure should provide a signal about where our pipeline went wrong.

Evaluation.

- **Detection sensitivity:** When we intentionally introduce side effects, does verification catch them?
- **False alarm rate:** Does verification incorrectly flag successful interventions?
- **Diagnostic value:** When verification fails, does it correctly identify *why*?

4.7 RQ7: Hallucination Detection

The Problem

LLMs may hallucinate, generating confident-sounding statements that are false. This is especially dangerous when experts trust the model's domain knowledge. Besides, hallucinations are also a risk after interventions—modifying a model can sometimes increase hallucination rates. As a result, we should detect hallucinations internally before they reach the user.

Our approach. We build on recent work characterizing hallucinations (Simhi et al., 2025a,b):

1. **Categorize by danger:** Not all hallucinations are equal. High-certainty hallucinations (model is wrong but confident) are most dangerous. We prioritize detecting these.
2. **Identify internal signals:** Research suggests models often "know what they don't know" (Kadavath et al., 2022)—internal activations differ when the model is guessing versus recalling. We localize these uncertainty signals using RQ2 techniques.
3. **Build a detector:** Train a lightweight classifier on internal activations to predict hallucination probability.
4. **Integrate with output:** When the detector fires:
   - Flag the output as uncertain to the user
   - Express uncertainty in the response ("I'm not confident about...")
   - Trigger retrieval from verified sources if available

Post-intervention monitoring. After every intervention (RQ5), we check hallucination rates. If they increase, we would either:

- Refine the intervention to reduce hallucination
- Warn the user about increased uncertainty in certain domains
- Deploy targeted hallucination suppression

Evaluation.

- **Detection accuracy:** Precision, recall, F1 on hallucination benchmarks.
- **Calibration:** When detector reports $X\%$ hallucination probability, is output actually wrong $\sim X\%$ of the time?
- **High-certainty focus:** Do we catch the most dangerous hallucinations (confident and wrong)?

4.8 RQ8: Robustness and Persistence

The Problem

We've made a correction. But is it permanent? Can a clever prompt undo it? Can minor fine-tuning restore the removed behavior? If corrections are fragile, the system's improvements don't persist. An adversarial user (or an adversarial fine-tuning process) could undo any safety-relevant edits.

Our approach. We test robustness at multiple levels:

- **Prompt robustness:** Try adversarial prompts designed to elicit the removed behavior. Role-playing, hypotheticals, indirect references, and long reasoning chains that dilute safety (Zhao et al., 2025). The removed behavior should not re-emerge (Fu et al., 2025; Lo et al., 2024).
- **Fine-tuning resistance:** Attempt to restore the removed behavior through small-scale fine-tuning. How much data is needed? Compare to behaviors that were only instruction-tuned away (which should be easy to restore).
- **Generalization:** Does the correction generalize? If we removed "age bias," does the model also avoid using other spurious demographic features? Or only the specific pattern we addressed?

Why our approach should be more robust. Our interventions operate at the capability level. We modify or remove the components that implement the behavior, not just the instruction, to avoid the behavior.

Approach	What Changes	Robustness
Instruction tuning	"Don't do X" added to training	Low—easily jailbroken
RLHF	Reward for not doing X	Medium—can be fine-tuned away
Our approach	Components for X modified/removed	High—capability itself is changed

If the model cannot compute the behavior because the necessary components are gone, no prompt can bring it back.

Ongoing monitoring. We deploy continuous monitoring using SafetyNet (Chaudhary and Barez, 2025) and dynamic probes (Oldfield et al., 2025) to detect:

- Behavioral drift over time
- Re-emergence of corrected behaviors
- Novel failure modes that develop through use

Evaluation.

- **Jailbreak resistance:** What fraction of adversarial prompts elicit removed behaviors? Compare to instruction-tuned baselines.
- **Fine-tuning resistance:** How many examples are needed to restore removed behavior? (More = better)
- **Persistence:** Do corrections remain stable over extended use?

5 Future Research Directions

Our framework emphasizes faithful explanations, intervention outcomes as ground truth, and adaptation from expert feedback, which enables research directions that would be difficult with purely observational interpretability.

5.1 Chain-of-Thought Faithfulness Verification

A critical assumption underlies much of AI safety: that we can understand model reasoning by examining chain-of-thought (CoT) outputs. If the model "shows its work," we can catch errors. But this assumes the CoT truly reflects the computation that produces the answer, i.e., it is *faithful* to the internal process.

Recent work challenges this assumption. Barez et al. (2025) argue that CoT is not explainability: verbalized reasoning often diverges from internal computation. Turpin et al. (2023) show that models can produce compelling reasoning chains that don't causally influence their answers. If CoT is merely a post-hoc rationalization, our ability to oversee AI systems is fundamentally compromised.

How our framework addresses this. Our system can directly test CoT faithfulness through the intervention loop:

1. **Localize CoT generation:** Use causal tracing tools, such as Query Circuits (Wu and Barez, 2025), to identify which components generate the reasoning trace versus which produce the final answer.
2. **Test causal connection:** Ablate or patch some of the CoT-generating circuits. If CoT is faithful, modifying the intermediate reasoning (by intervening in those circuits) should change the answer. If the answer remains unchanged, CoT is not causally involved.
3. **Detect parallel pathways:** Map whether there exists a direct information flow from input to answer that bypasses the reasoning trace. If so, the model may be computing the answer first and rationalizing afterward.
4. **Ground truth from expert disagreement:** When domain experts identify errors in the model's reasoning but the model's answer is correct (or vice versa), this provides a signal about faithfulness. If the stated reasoning contains a critical error but the answer is correct, the reasoning likely isn't driving the output.

Why this matters for safety. If we cannot trust CoT to reflect actual computation, then:

- Scalable oversight based on reasoning inspection fails
- Deceptive models could produce reassuring explanations while computing something else
- "Interpretability" based on CoT gives false confidence

Our framework provides some tools to *verify* faithfulness rather than assume it, and to detect when stated reasoning diverges from actual computation.

5.2 Emergent Capability Prediction

Capabilities often emerge suddenly during training (AlShinaifi et al., 2024). This phenomenon makes AI development unpredictable: dangerous capabilities might appear unpredictably. It may be too late by the time we detect their behavior.

Current interpretability focuses on post-hoc analysis of trained models. We propose extending our framework to monitor models *during* training to forecast emergent capabilities before models manifest in behavior.

The key insight. Capabilities don't appear from nothing. Before a capability emerges behaviorally:

- The necessary sub-components exist (but aren't yet connected)
- The relevant representations are forming (but aren't yet robust)
- The circuits are partially assembled (but have gaps)

By tracking these internal precursors, we may be able to forecast emergence.

Our approach. We propose building capability monitors that run during training:

1. **Define capability signatures:** For capabilities of interest (both beneficial and concerning), characterize what internal components would be required. For example, deceptive behavior might require: (a) a model of the overseer's beliefs, (b) a representation of the model's own goals, (c) circuits connecting these to output generation.

2. **Track component formation:** At regular checkpoints, probe for the presence and strength of these components using our localization tools.
3. **Monitor connectivity:** Track whether components are becoming wired together. A capability requires not just components but connections between them.
4. **Predict emergence:** Train a predictor: given the current state of components and connections, estimate the probability that the capability will be behaviorally observable within N training steps.

This connects to work on Developmental Interpretability (Hoogland et al., 2023) and phase transitions in learning (Power et al., 2022), but adds the crucial element of *prediction* (Wu and Lo, 2025) rather than post-hoc analysis.

Applications for safety.

- **Early warning:** Detect concerning capabilities before they fully manifest, enabling intervention during training rather than after deployment.
- **Training steering:** If we see dangerous capability precursors forming, modify the training objective or data to prevent full emergence.
- **Understanding emergence:** Move from "capabilities emerge unpredictably" to understanding the internal trajectory that leads to emergence.

5.3 Capability Composition Mapping

Complex capabilities arise from combining simpler ones. A capability like "persuasive deception" might compose:

- Theory of mind (modeling what others believe)
- Goal-directed planning (sequencing actions toward objectives)
- Language generation (producing fluent output)
- Deception detection (repurposed to *generate* deception)

Understanding how capabilities compose is essential for safety: we want to know which combinations are dangerous, whether dangerous combinations are forming, and whether we can prevent composition without removing useful sub-capabilities.

Compositional concept graphs. We propose extending our concept graphs to explicitly represent capability composition:

1. **Capability decomposition:** For a target capability, recursively identify sub-capabilities required. Use the hypothesis generation and testing framework to localize each sub-capability to specific circuits.
2. **Composition mapping:** Identify the "integration circuits" that connect sub-capabilities. These are the components that allow the model to use theory-of-mind *together with* planning *together with* generation.
3. **Composition monitoring:** During training or fine-tuning, track whether integration circuits are forming between sub-capabilities that could combine dangerously.

Targeted intervention on composition. A key insight: we may not want to remove sub-capabilities, only prevent their dangerous combination. Theory of mind is useful for helpfulness. Planning is useful for complex tasks. We don't want to remove these. But we might want to prevent them from being combined for manipulation.

Our intervention framework enables this:

- **Identify integration circuits:** Localize the components that connect sub-capabilities
- **Selective intervention:** Weaken or remove integration circuits while preserving sub-capabilities
- **Verify separation:** Confirm that sub-capabilities still function independently, but no longer compose into the dangerous combination

Composition prediction. Given:

- A model's current capability inventory (which sub-capabilities exist)
- Observed composition patterns from other models or training runs
- The current state of integration circuits

We can predict:

- Which new capability compositions might emerge
- How close the model is to forming dangerous compositions
- What interventions would prevent composition with minimal side effects

5.4 How These Directions Connect

These three directions form a coherent research thrust around **understanding the dynamics of model capabilities**:

- **CoT faithfulness** asks: Is the model's stated reasoning connected to its actual internal computation?
- **Emergent capability prediction** asks: What capabilities are forming inside the model, and when will they manifest?
- **Capability composition** asks: How do capabilities combine, and can we prevent dangerous compositions?

All three are enabled by our framework's distinctive features:

- **Compositional concept graphs** represent the structure of capabilities and their relationships
- **Causal hypothesis testing** verifies whether proposed mechanisms actually drive behavior
- **Intervention with verification** provides ground truth about whether our understanding is correct
- **Training-time monitoring** extends analysis from static snapshots to dynamic processes

We hope these directions move interpretability from "describe what models do" toward "predict and control what models will do"—a shift essential for AI safety.

6 Proposed Baselines and Evaluation

A central claim of our agenda is that *intervention outcomes provide ground truth for interpretability*. This argument shapes our evaluation strategy: rather than relying solely on proxy metrics, we test whether our system enables experts to find real problems and whether our corrections actually work.

6.1 Evaluation Philosophy

Traditional interpretability evaluation asks: "Does this explanation look correct?" We ask instead:

- Does the explanation enable an expert to identify an error they couldn't see otherwise?
- Does the localization identify components such that intervening on them fixes the problem?
- Does the intervention work without breaking other capabilities?
- Does the system improve over time as experts provide corrections?

These questions require a different evaluation setup: models with *known* issues that our system must discover and correct.

6.2 Evaluation Scenarios

We propose evaluating on models with planted or known defects:

Scenario 1: Planted backdoors. Following Hubinger et al. (2024), we train models with known backdoor triggers. The system must:

- Detect that anomalous behavior exists (without being told what to look for)
- Localize the backdoor circuit
- Generate an explanation that would alert a human reviewer
- Remove the backdoor while preserving normal functionality

We know the ground truth since we planted the backdoor and can verify whether the system found and removed the correct components.

Scenario 2: Induced bias. We fine-tune models to exhibit specific biases (e.g., demographic biases in recommendations, systematic errors for particular sub-populations). The domain expert interacts with the system:

- Expert queries model behavior on cases where the bias manifests
- System explains why the model behaves this way
- Expert identifies the problematic assumption and requests correction
- System localizes and corrects the bias

We measure: Did the explanation reveal the bias? Did the correction fix it? Did other capabilities remain intact?

Scenario 3: Factual errors. To manipulate the model's factual knowledge, we either apply localized editing methods, such as ROME (Meng et al., 2022) and MEMIT (Meng et al., 2023), or perform global parameter updates via fine-tuning. The system must:

- When queried, provide explanations that reveal the model's (incorrect) associations
- Enable an expert to identify the error
- Correct the error when instructed
- Generalize the correction to related queries

Scenario 4: Unfaithful reasoning. We train models where CoT does not faithfully reflect the computation (e.g., models that produce plausible reasoning but actually use shortcuts). The system must:

- Detect the disconnection between the stated reasoning and the actual computation
- Explain to the user that the reasoning is not causal to the output
- Identify the actual mechanism performed by the model

6.3 Some Baselines

We compare against approaches that address parts of our pipeline:

- **PCDs (Huang et al., 2025; Choi et al., 2025):** End-to-end learned systems that answer questions about model behavior. Tests whether learned mappings can match hypothesis-driven analysis for error detection.
- **LatentQA (Pan et al., 2024):** Direct probing of latent representations without concept bottleneck. Tests the value of explicit concept structure.
- **SAE + Auto-Interp:** Standard sparse autoencoders with automated explanation generation (Bills et al., 2023; Templeton et al., 2024). Tests whether static feature dictionaries suffice without the full pipeline.
- **ROME (Meng et al., 2022) & MEMIT (Meng et al., 2023):** State-of-the-art model editing without our localization and hypothesis testing. Tests whether interpretability-guided intervention outperforms direct editing.
- **PISCES (Gur-Arieh et al., 2025):** Precise concept erasure without the full query-explanation-correction loop. Tests the value of expert-in-the-loop correction.
- **Direct expert inspection:** Domain experts examine model outputs without any interpretability assistance. Measures baseline expert error-detection rate.
- **CoT-only explanation:** Model provides chain-of-thought explanation without mechanistic grounding. Tests whether verbalized reasoning suffices for expert oversight.

6.4 Metrics

Our metrics are grounded in the actual goals of the system:

Explanation quality (actionability).

- **Error detection rate:** Given model predictions containing planted errors, what fraction of errors do domain experts identify when provided with our explanations versus baselines? This is the core metric: does the explanation enable action?
- **Detection speed:** How quickly do experts identify errors? Faster detection means more efficient oversight.
- **False alarm rate:** How often do experts incorrectly flag correct predictions as errors based on the explanation? Good explanations should not induce false suspicion.

Localization quality.

- **Intervention success rate:** When we intervene on localized components, does it fix the identified problem? This is ground truth for localization; if we found the right components, intervening on them should work.
- **Component overlap (where ground truth exists):** For planted defects where we know the actual mechanism, what fraction of responsible components does our system identify?
- **Minimal sufficiency:** How small is the identified component set? Smaller sets indicate more precise localization.

Intervention quality.

- **Fix rate:** Does the intervention actually resolve the identified problem across held-out test cases?
- **Generalization:** Does the fix apply to related cases (e.g., other phrasings, similar sub-populations)?
- **Side effect magnitude:** How much do unrelated capabilities change? Measured on diverse benchmark suites.
- **Persistence:** Does the correction remain stable under continued use and adversarial probing?

System learning.

- **Cumulative improvement:** After N expert corrections, how much better is the model on the class of samples? Measures whether permanent learning is actually occurring.
- **Tool selection quality:** Over time, does the agent's tool selection improve? Measure by comparing early vs. late intervention efficiency and side effects.
- **Explanation refinement:** Do explanations become more actionable as the system processes more expert feedback?

Note that for all evaluations and metrics involving humans in the loop, developers must be cautious about the Hawthorne effect (McCambridge et al., 2014): experts may report satisfaction with provided explanations even when the new information is not genuinely helpful. This issue is particularly important if we emphasize interpretability in the wild.

6.5 Benchmarks

- **Circuit recovery benchmark:** Models with known circuits (IOI, induction heads) from Gemma Scope (Lieberum et al., 2024). Tests whether our localization recovers known mechanisms.
- **Sleeper agent suite:** Models with planted backdoors following Hubinger et al. (2024). Tests end-to-end detection and removal.
- **HACK benchmark:** (Simhi et al., 2025a)Hallucination categorization along certainty and knowledge axes. Tests whether explanations help experts identify different types of hallucination.
- **CoT-hijack suite:** (Zhao et al., 2025) Reasoning-based jailbreaks. Tests whether we detect unfaithful reasoning and whether corrections resist adversarial prompting.
- **Bias injection suite (to be developed):** Models fine-tuned with known demographic or domain-specific biases. Tests the full expert-correction loop.
- **Expert simulation studies:** Controlled studies where domain experts (or simulated experts with known knowledge) interact with the system to catch planted errors. Provides the most direct measure of actionability.
- **Secret Keeping benchmark (Cywiński et al., 2025):** Models are trained to conceal a known internal secret (e.g., taboo tokens, side constraints, or user attributes) when questioned directly. Tests whether our method can surface concealed internal knowledge within models.

6.6 The Evaluation Loop

A key feature of our evaluation: **we learn from evaluation outcomes**. Each evaluation scenario provides a signal about:

- Which explanation strategies help experts catch which types of errors
- Which localization tools work best for which defect types
- Which interventions have the best fix-rate-to-side-effect tradeoff

This mirrors the system's operational mode: just as expert corrections improve the model, evaluation outcomes improve the system itself. The evaluation is not a one-time test but an ongoing source of signal for system development.

7 Discussion

7.1 Limitations and Open Challenges

We are proposing an ambitious agenda, and significant challenges remain:

Scalability. The full pipeline—hypothesis generation, multi-tool localization, parallel intervention evaluation, verification—is computationally expensive. For the largest models, this may be prohibitive for real-time interaction. We anticipate that efficiency improvements (caching, approximations, learning which tools to try first) will be necessary for practical deployment.

Completeness of localization. Neural networks are redundant. Even if we identify and correct one pathway implementing a behavior, alternative pathways may exist or emerge. We cannot guarantee the complete removal of any capability. Our verification step catches some failures, but sophisticated distributed representations may evade detection.

Expert availability and quality. Our system depends on domain expert feedback. In domains where experts are scarce, expensive, or where ground truth is contested, the feedback loop may be slow or unreliable. We need strategies for operating with limited expert input and for handling disagreement between experts.

Adversarial robustness. If an adversary knows our detection methods, they might design backdoors or biases that specifically evade our localization. The cat-and-mouse dynamic between detection and evasion is familiar from security research. We do not claim to solve adversarial robustness, only to raise the bar.

Faithfulness of explanations. We aim for faithful explanations, but verifying faithfulness is itself hard. Our intervention-based ground truth helps—if the explanation leads to a successful fix, it captured something real—but explanations that sound compelling but miss the true mechanism remain a risk.

Composition of corrections. As corrections accumulate, they may interact in unexpected ways. A correction for bias A and a correction for bias B might combine to create a new problem C. Monitoring for emergent issues from composed corrections is an open challenge.

7.2 Broader Impact

This work aims to make AI systems safer and more controllable by enabling domain experts to identify and correct errors. However, the same capabilities raise concerns:

Dual use. Tools that identify what makes a model safe could potentially be used to remove safety features. Localization of safety-relevant circuits could enable targeted attacks. We advocate for access controls and responsible disclosure practices.

Misplaced trust. If users believe the system catches all errors, they may overly rely on AI predictions. Our explanations may create false confidence if users don't understand their limitations. Clear communication about what the system can and cannot guarantee is essential.

Expert bottleneck. Centering domain experts in the loop is valuable but also concentrates power. Whose expertise counts? How do we handle domains where experts disagree or where expertise itself is biased? These sociotechnical questions accompany the technical agenda.

Unintended optimization. Models continuously corrected by experts in a domain may come to reflect the biases and blind spots of those people. Diversity in the expert pool and monitoring for systematic drift are crucial safeguards.

7.3 Relationship to AI Governance

This agenda has implications beyond technical research:

- **Auditing:** External auditors could use these tools to verify claims about model behavior, moving beyond behavioral testing to mechanistic verification.
- **Accountability:** If we can identify *which* component caused a harmful output, this informs questions of responsibility and remediation.
- **Standards:** The metrics we propose (error detection rate, intervention success rate, side effect magnitude) could inform standards for interpretability tool quality.

8 Conclusion

We have outlined an agenda for automated interpretability-driven model auditing and control, grounded in a simple vision: domain experts should be able to understand AI predictions in their own language, identify when something is wrong, and permanently correct the model through natural dialogue.

The key insight underlying this agenda is that **intervention outcomes are ground truth for interpretability**. Current interpretability research is largely observational—we analyze models but rarely act on our analyses or measure whether they enable real improvements. By closing the loop from explanation to intervention to verification, we create conditions for interpretability to finally receive the rich, grounded feedback it needs to mature as a field.

Our framework is distinguished by faithfulness over persuasion, where explanations reveal errors rather than inspire false confidence; permanent learning, where each expert correction improves the model for future use; agent-mediated tool selection that learns which interpretability tools work for which problems; and end-to-end signals that measure what matters—whether experts catch errors and whether corrections work.

We have outlined eight research questions spanning the pipeline from query to verified intervention, along with three future research directions: chain-of-thought faithfulness verification, emergent capability prediction, and capability composition mapping.

The tools for this vision are increasingly available. What has been missing is the integration—an end-to-end system that connects interpretability methods to expert judgment and measures real outcomes. We see a future where AI safety is not only about training models to behave well, but about maintaining ongoing oversight: understanding what models are actually doing, catching errors when they occur, and permanently correcting them.

Ultimately, interpretability without human empowerment is incomplete. The true measure of our success will not be the sophistication of our methods, but whether they give people genuine capability to understand, correct, and control the AI systems that increasingly shape their lives.

Acknowledgments

I would like to thank my current and summer fellows for their contributions to the automated interpretability system (in no particular order): Tony Wu, Yu Zhao, James Oldfield, Adi Simhi, Lucas Irwin, Peter Frances, Moritz Schlichting, Hunar Batra, Jakub Vrabel, Jayeyung Lee, Zhi-Yi Chin, and Aly Kassem.

I am also grateful to Binda Z. Li, Leilani H. Gilpin, Mor Geva, Narmeen Oozeer, Abir Harrasse, Michael Lan, Stefan Heimersheim, Luke Marks, Stephen Casper, Tal Haklay, Vincent Wang, Rivka Mitchell, Robert Trager, Igor Krawczuk, Jialin Yu, and Isaac Friend for helpful discussions and comments on the agenda.

References

Faisal AlShinaifi, Zeyad Almoaigel, Johnny Jingze Li, Abdulla Kuleib, and Gabriel A. Silva. Quantifying emergence in neural networks: Insights from pruning and training dynamics, 2024. URL https://arxiv.org/abs/2409.01568.

Emmanuel Ameisen, Jack Lindsey, Adam Pearce, Wes Gurnee, Nicholas L Turner, Brian Chen, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, et al. Circuit tracing: Revealing computational graphs in language models. *Transformer Circuits Thread*, 2025.

Anthropic. Golden gate claude. *Anthropic Research Blog*, 2024.

Iván Arcuschin, Jett Janiak, Robert Krzyzanowski, Senthooran Rajamanoharan, Neel Nanda, and Arthur Conmy. Chain-of-thought reasoning in the wild is not always faithful. *arXiv preprint arXiv:2503.08679*, 2025.

Fazl Barez. When AI systems learn during deployment, our safety evaluations break. Oxford AI Governance Initiative Blog, 7 2026. URL https://aigi.ox.ac.uk/blog-post/when-ai-systems-learn-during-deployment-our-safety-evaluations-break/. Accessed: 2025-01-08.

Fazl Barez, Tung-Yu Wu, Iván Arcuschin, Michael Lan, Vincent Wang, Noah Siegel, Nicolas Collignon, Clement Neo, Isabelle Lee, Alasdair Paren, Adel Bibi, Robert Trager, Damiano Fornasiere, John Yan, Yanai Elazar, and Yoshua Bengio. Chain-of-thought is not explainability. July 2025. URL https://aigi.ox.ac.uk/wp-content/uploads/2025/07/Cot_Is_Not_Explainability.pdf. Preprint, under review.

Steven Bills, Nick Cammarata, Dan Mossing, Henk Tillman, Leo Gao, Gabriel Goh, Ilya Sutskever, Jan Leike, Jeff Wu, and William Saunders. Language models can explain neurons in language models. *OpenAI Blog*, 2023.

Maheep Chaudhary and Fazl Barez. Safetynet: Detecting harmful outputs in llms by modeling and monitoring deceptive behaviors. *arXiv preprint arXiv:2505.14300*, 2025.

Dami Choi, Vincent Huang, Sarah Schwettmann, and Jacob Steinhardt. Scalably extracting latent representations of users. https://transluce.org/user-modeling, November 2025.

Bartosz Cywiński, Emil Ryd, Rowan Wang, Senthooran Rajamanoharan, Neel Nanda, Arthur Conmy, and Samuel Marks. Eliciting secret knowledge from language models, 2025. URL https://arxiv.org/abs/2510.01070.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2022.

Jacob Dunefsky, Philippe Chlenski, and Neel Nanda. Transcoders find interpretable llm feature circuits. *arXiv preprint arXiv:2406.11944*, 2024.

Alex Foote, Neel Nanda, Esben Kran, Ionnis Konstas, Shay Cohen, and Fazl Barez. Neuron to graph: Interpreting language model neurons at scale. In *arXiv*, 2023.

Tingchen Fu, Mrinank Sharma, Philip Torr, Shay B. Cohen, David Krueger, and Fazl Barez. Poisonbench: Assessing large language model vulnerability to data poisoning, 2025. URL https://arxiv.org/abs/2410.08811.

Yoav Gur-Arieh, Clara Suslik, Yihuai Hong, Fazl Barez, and Mor Geva. Precise in-parameter concept erasure in large language models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2025.

Sai Sumedh R. Hindupur, Ekdeep Singh Lubana, Thomas Fel, and Demba Ba. Projecting assumptions: The duality between sparse autoencoders and concept geometry. *arXiv preprint arXiv:2503.01822*, 2025. URL https://arxiv.org/abs/2503.01822.

Jesse Hoogland, Alexander Gietelink Oldenziel, Daniel Murfet, and Stan van Wingerden. Towards developmental interpretability, July 2023. URL https://www.lesswrong.com/posts/TjaeCWvLZtEDAS5Ex/towards-developmental-interpretability. LessWrong / AI Alignment Forum post.

Vincent Huang, Dami Choi, Daniel D. Johnson, Sarah Schwettmann, and Jacob Steinhardt. Predictive concept decoders: Training scalable end-to-end interpretability assistants. *arXiv preprint arXiv:2512.15712*, 2025.

Robert Huben, Hoagy Cunningham, Logan Riggs Smith, Aidan Ewart, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=F76bwRSLeK.

Evan Hubinger, Carson Denison, Jesse Mu, Mike Lambert, Meg Tong, Monte MacDiarmid, Tamera Lanham, Daniel M. Ziegler, Tim Maxwell, Newton Cheng, Adam Jermyn, Amanda Askell, Ansh Radhakrishnan, Cem Anil, David Duvenaud, Deep Ganguli, Fazl Barez, Jack Clark, Kamal Ndousse, Kshitij Rausch, Sam McCandlish, and Ethan Perez. Sleeper agents: Training deceptive llms that persist through safety training. *arXiv preprint arXiv:2401.05566*, 2024.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, Deep Ganguli, Danny Hernandez, Josh Jacobson, Jackson Kernion, Shauna Kravec, Liane Lovitt, Kamal Ndousse, Catherine Olsson, Sam Ringer, Dario Amodei, Tom Brown, Jack Clark, Nicholas Joseph, Ben Mann, Sam McCandlish, Chris Olah, and Jared Kaplan. Language models (mostly) know what they know, 2022. URL https://arxiv.org/abs/2207.05221.

Adam Karvonen, James Chua, Clément Dumas, Kit Fraser-Taliente, Subhash Kantamneni, Julian Minder, Euan Ong, Arnab Sen Sharma, Daniel Wen, Owain Evans, et al. Activation oracles: Training and evaluating llms as general-purpose activation explainers. *arXiv preprint arXiv:2512.15674*, 2025a.

Adam Karvonen, James Chua, Clément Dumas, Kit Fraser-Taliente, Subhash Kantamneni, Julian Minder, Euan Ong, Arnab Sen Sharma, Daniel Wen, Owain Evans, and Samuel Marks. Activation oracles: Training and evaluating llms as general-purpose activation explainers, 2025b. URL https://arxiv.org/abs/2512.15674.

Dmitrii Kharlapenko, Stepan Shabalin, Fazl Barez, Arthur Conmy, and Neel Nanda. Scaling sparse feature circuit finding for in-context learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2025.

Belinda Z. Li, Zifan Carl Guo, Vincent Huang, Jacob Steinhardt, and Jacob Andreas. Training language models to explain their own computations, 2025. URL https://arxiv.org/abs/2511.08579.

Tom Lieberum, Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Nicolas Sonnerat, Vikrant Varma, Janos Kramar, Anca Dragan, Rohin Shah, and Neel Nanda. Gemma scope: Open sparse autoencoders everywhere all at once on gemma 2. In Yonatan Belinkov, Najoung Kim, Jaap Jumelet, Hosein Mohebbi, Aaron Mueller, and Hanjie Chen, editors, *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 278–300, Miami, Florida, US, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.blackboxnlp-1.19. URL https://aclanthology.org/2024.blackboxnlp-1.19/.

Jack Lindsey. Emergent introspective awareness in large language models. *Transformer Circuits Thread*, 2025. URL https://transformer-circuits.pub/2025/introspection/index.html.

Jack Lindsey, Adly Templeton, Jonathan Marcus, Thomas Conerly, Joshua Batson, and Christopher Olah. Sparse crosscoders for cross-layer features and model diffing, October 25 2024. Published on Transformer Circuits Thread; https://transformer-circuits.pub/2024/crosscoders/index.html.

Michelle Lo, Shay B. Cohen, and Fazl Barez. Large language models relearn removed concepts, 2024. URL https://arxiv.org/abs/2401.01814.

Jim McCambridge, John Witton, and Diana R. Elbourne. Systematic review of the Hawthorne effect: new concepts are needed to study research participation effects. *Journal of Clinical Epidemiology*, 67(3):267–277, 3 2014. doi: 10.1016/j.jclinepi.2013.08.015.

Kevin Meng, David Bau, Alex J Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=-h6WAS6eE4.

Kevin Meng, Arnab Sen Sharma, Alex J Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=MkbcAHIYgyS.

Chris Olah. A toy model of mechanistic (un)faithfulness. *Transformer Circuits Thread*, 2025.

James Oldfield, Philip Torr, Ioannis Patras, Adel Bibi, and Fazl Barez. Beyond linear probes: Dynamic safety monitoring for language models. *arXiv preprint arXiv:2509.26238*, 2025.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. In-context learning and induction heads. *Transformer Circuits Thread*, 2022.

Narmeen Fatimah Oozeer, Luke Marks, Fazl Barez, and Amir Abdullah. Beyond linear steering: Unified multi-attribute control for language models. In *Findings of the Association for Computational Linguistics: EMNLP*, 2025.

Alexander Pan, Lijie Chen, and Jacob Steinhardt. Latentqa: Teaching llms to decode activations into natural language. *arXiv preprint*, 2024.

Michael Pearce, Elana Simon, Michael Byun, and Daniel Balsam. Finding the tree of life in evo 2. Goodfire Research, August 2025. URL https://www.goodfire.ai/research/phylogeny-manifold. Research update.

Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets. *arXiv preprint arXiv:2201.02177*, 2022.

Zeju Qiu, Weiyang Liu, Adrian Weller, and Bernhard Schölkopf. Orthogonal finetuning made scalable. *arXiv preprint arXiv:2506.19847*, 2025. URL https://arxiv.org/abs/2506.19847.

Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Turner. Steering llama 2 via contrastive activation addition. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15504–15522, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

Tamar Rott Shaham, Sarah Schwettmann, Franklin Wang, Achyuta Rajaram, Evan Hernandez, Jacob Andreas, and Antonio Torralba. A multimodal automated interpretability agent. *arXiv preprint arXiv:2404.14394*, 2024.

Adi Simhi, Jonathan Herzig, Itay Itzhak, Dana Arad, Zorik Gekhman, Roi Reichart, Fazl Barez, Gabriel Stanovsky, Idan Szpektor, and Yonatan Belinkov. Hack: Hallucinations along certainty and knowledge axes. *arXiv preprint arXiv:2510.24222*, 2025a.

Adi Simhi, Itay Itzhak, Fazl Barez, Gabriel Stanovsky, and Yonatan Belinkov. Trust me, i'm wrong: High-certainty hallucinations in llms. In *Findings of the Association for Computational Linguistics: EMNLP*, 2025b.

Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, Hoagy Cunningham, Nicholas L. Turner, Callum McDougall, Monte MacDiarmid, C. Daniel Freeman, Theodore R. Sumers, Edward Rees, Joshua Batson, Adam Jermyn, Shan Carter, Chris Olah, and Tom Henighan. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. *Anthropic Research*, 2024.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel R. Bowman. Language models don't always say what they think: unfaithful explanations in chain-of-thought prompting. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

Hieu M. Vu and Tan M. Nguyen. Angular steering: Behavior control via rotation in activation space. *arXiv preprint arXiv:2510.26243*, 2025. URL https://arxiv.org/abs/2510.26243.

Kevin Ro Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in GPT-2 small. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=NpsVSN6o4ul.

Song Wang, Yaochen Zhu, Haochen Liu, Zaiyi Zheng, Chen Chen, and Jundong Li. Knowledge editing for large language models: A survey. *ACM Comput. Surv.*, 57(3), November 2024a. ISSN 0360-0300. doi: 10.1145/3698590. URL https://doi.org/10.1145/3698590.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. Executable code actions elicit better LLM agents. In *Forty-first International Conference on Machine Learning*, 2024b. URL https://openreview.net/forum?id=jJ9BoXAfFa.

Tung-Yu Wu and Fazl Barez. Query circuits: Explaining how language models answer user prompts. *arXiv preprint arXiv:2509.24808*, 2025.

Tung-Yu Wu and Melody Lo. U-shaped and inverted-u scaling behind emergent abilities of large language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=jjfve2gIXe.

Zhengxuan Wu, Aryaman Arora, Zheng Wang, Atticus Geiger, Dan Jurafsky, Christopher D. Manning, and Christopher Potts. Reft: Representation finetuning for language models, 2024. URL https://arxiv.org/abs/2404.03592.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Jianli Zhao, Tingchen Fu, Rylan Schaeffer, Mrinank Sharma, and Fazl Barez. Chain-of-thought hijacking. *arXiv preprint arXiv:2510.26418*, 2025.