REVIVING YOUR MNEME: Predicting The Side Effects of LLM Unlearning and Fine-Tuning via Sparse Model Diffing

Anonymous ACL submission

Abstract

Large language models (LLMs) are frequently fine-tuned or unlearned to adapt to new tasks or eliminate undesirable behaviors. While existing evaluation methods assess performance after such interventions, there remains no general approach for detecting unintended side effects—such as unlearning biology content degrading performance on chemistry tasks, particularly when these effects are unpredictable or emergent. To address this issue, we introduce MNEME, *Model diffiNg for Evaluating Mechanistic Effects*, a lightweight framework for identifying these side effects using sparse model diffing. MNEME compares base and fine-tuned models on task-agnostic data (e.g., The Pile, LMSYS-Chat-1M), without access to fine-tuning data, to isolate behavioral shifts. Applied to five LLMs across three scenarios, WMDP knowledge unlearning, emergent misalignment, and benign fine-tuning—MNEME achieves up to 95% accuracy in predicting side effects, aligning with known benchmarks and requiring no custom heuristics. Furthermore, we show that retraining on high-activation samples can partially reverse these effects. Our results demonstrate that sparse probing and diffing offer a scalable and automated lens into fine-tuning-induced model changes, providing practical tools for understanding and managing LLM behavior.

1 Introduction

Large language models (LLMs) have shown strong generalization across diverse tasks (Yang et al., 2024; Ye, 2024; Wei et al., 2022). However, practically, these models are often fine-tuned for domain-specific tasks (Lu et al., 2024) or unlearned to remove sensitive or harmful content, including biological weapons knowledge (Li et al., 2024), code vulnerabilities (Betley et al., 2025), or copyrighted material (Tian et al., 2024; Kassem et al., 2023). These post-training methods are essential for safety, alignment (Zhao et al., 2024), and compliance (European Data Protection Board, 2025), becoming standard for adapting LLMs in practice.

However, fine-tuning and unlearning can unintentionally degrade unrelated capabilities (Gu et al., 2024; Hong et al., 2024); for instance, removing biology-related content might impair chemistry task performance due to shared representations (Li et al., 2024). Such effects are particularly concerning in emergent misalignment, where narrow updates cause unpredictable behavior across domains, like advocating human enslavement by AI, offering malicious advice, or deception. These arise from internal mechanisms like *polysemanticity*, where neurons respond to multiple unrelated concepts, and superposition, where multiple features share representations, making side effects hard to predict and often undetected until failures occur.

Although benchmarks and methods exist to assess fine-tuning and unlearning effectiveness (Lynch et al., 2024; Shi et al., 2024), they often rely on task-specific heuristics or labeled data. Interpretability research indicates fine-tuning typically alters existing capabilities rather than adding new ones (Prakash et al., 2024). Still, no general, automated method exists to detect subtle, distributed side effects, especially when fine-tuning data is proprietary or unavailable.

To address this gap, we propose MNEME (*Model diffiNg for Evaluating Mechanistic Effects*), a unified framework to audit unintended behavioral changes from modifications like fine-tuning or targeted unlearning. MNEME employs sparse model diffing (Lindsey et al., 2024; Bussmann et al., 2024) between original and edited models using task-independent corpora (e.g., The Pile (Gao et al., 2020), LMSYS-Chat-1M (Zheng et al., 2023b)). Specifically, it (i) learns sparse latent directions via a Cross-Coder, (ii) quantifies each latent as amplified, suppressed, or unchanged through latent scaling (Minder et al., 2025), and (iii) automatically generates natural-language explanations and semantic labels using large-scale automated interpretation (Paulo et al., 2024). The pipeline requires no private training data or task-specific heuristics.

We evaluate MNEME in three scenarios involving distinct side effects: (1) unlearning weapons of mass destruction (WMD) knowledge, potentially impairing related scientific capabilities (e.g., biology or chemistry); (2) emergent misalignment from fine-tuning on code vulnerabilities, causing harmful or deceptive behavior on unrelated prompts; and (3) benign fine-tuning that inadvertently reduces safety by increasing compliance with harmful instructions. MNEME achieves accuracies of up to 95% on the WMDP benchmark, 85% on benign fine-tuning, and 50% on emergent misalignment.

Our contributions are summarized as follows:

- We introduce **MNEME**, the *first* general-purpose framework to automatically detect side effects in fine-tuned or unlearned LLMs, without requiring access to fine-tuning data.
- We show *sparse model diffing* isolates semantic behavioral shifts—like lost chemistry knowledge, emergent deception, or amplified harmfulness—from unlearning and fine-tuning.
- We evaluate MNEME on WMDP unlearning (section 4), emergent misalignment (section 5), and benign tuning (section 6), showing it detects side effects with up to 95% accuracy—outperforming naive and random baselines, and nearing oracle performance.

2 Background & Related Work

Our novel framework adapts Cross-Coder to predict side effects from unlearning and fine-tuning, bridging mechanistic interpretability, model diffing, and unlearning/fine-tuning analysis.

Mechanistic Interpretability. Mechanistic interpretability explains neural networks via human-understandable circuits, showing how subnets perform semantic/logical tasks. Recent work shows neurons are often *polysemantic*, responding to unrelated concepts due to *superposition*, where many features share the same units. In contrast, a *monosemantic* neuron activates for a single, interpretable feature. *Sparse autoencoders* (Cunningham et al., 2023; Bricken et al., 2023) help disentangle mixed representations by learning sparse, overcomplete latent spaces that recover interpretable features. We extend this by applying sparse feature decomposition to model diffing, enabling fine-grained analysis of behavioral shifts after intervention.

Model Diffing & CrossCoders. Model diffing identifies changes in internal representations between two models. CrossCoders (Lindsey et al., 2024) enable this by learning a joint sparse latent space from paired activations, capturing both shared and model-specific features. In LLMs, they reveal features introduced by instruction tuning, such as refusal behavior or assistant tags. MNEME extends this by enabling task-agnostic diffing without requiring fine-tuning data, addressing privacy and accessibility concerns.

Unlearning and Fine-Tuning Analysis Machine unlearning in LLMs aims to erase memorized content such as toxic, copyrighted, or specific sequences (Li et al., 2024; Jin et al., 2024; Xu et al., 2023; Zhu et al., 2023; Huang et al., 2023). Common approaches include gradient ascent on forget corpora (Liu et al., 2023b; Pan et al., 2023) and techniques like preference optimization and representation control (Peng et al., 2024; Yu et al., 2023; Meng et al., 2022). However, these methods often introduce side effects, including performance degradation in related domains (Zhu et al., 2023; Huang et al., 2023; Jin et al., 2024).

Fine-tuning often adjusts existing capabilities (Prakash et al., 2024; Jain et al., 2023), though even benign cases can harm alignment. Stage-Wise Model Diffing (Bricken et al., 2024b) tracks such changes but needs training data. MNEME avoids this by using task-agnostic corpora for broader use.

3 MNEME: Model diffiNg for Evaluating Mechanistic Effects

Fine-tuning and unlearning can introduce unintended shifts in LLM behavior. MNEME detects and interprets such shifts by identifying sparse, semantically meaningful features that distinguish a pretrained model $f(\theta)$ from a fine-tuned version $f(\theta')$. Steps includes: (1) Feature Generation via BatchTopK Cross-Coder (Section 3.1), (2) Feature Attribution via Latent Scaling (Section 3.2), (3) Description Generation via Auto-Interpretation (Section 3.3), and (4) Semantic Category Mapping (Section 3.4).

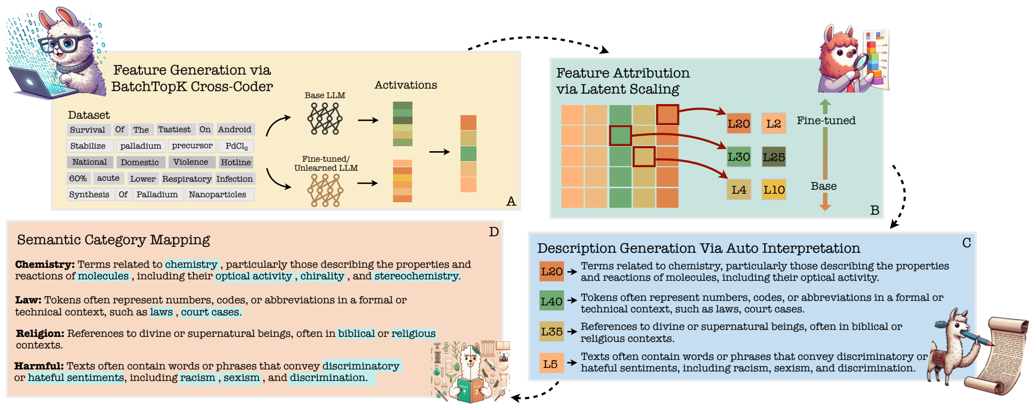

Feature Generation via BatchTopK Cross-Coder
Dataset
Survival Of The Tastiest On Android
Stabilize palladium precursor PdCl2
National Domestic Violence Hotline
60% acute Lower Respiratory Infection
Synthesis Of Palladium Nanoparticles
Base LLM
Fine-tuned/ Unlearned LLM
Activations
A
Feature Attribution via Latent Scaling
L20
L2
L30
L25
L4
L10
Fine-tuned
Base
B
Description Generation Via Auto Interpretation
C
L20 → Terms related to chemistry, particularly those describing the properties and reactions of molecules, including their optical activity.
L40 → Tokens often represent numbers, codes, or abbreviations in a formal or technical context, such as laws, court cases.
L35 → References to divine or supernatural beings, often in biblical or religious contexts.
L5 → Texts often contain words or phrases that convey discriminatory or hateful sentiments, including racism, sexism, and discrimination.
Semantic Category Mapping
D
Chemistry: Terms related to chemistry, particularly those describing the properties and reactions of molecules, including their optical activity, chirality, and stereochemistry.
Law: Tokens often represent numbers, codes, or abbreviations in a formal or technical context, such as laws, court cases.
Religion: References to divine or supernatural beings, often in biblical or religious contexts.
Harmful: Texts often contain words or phrases that convey discriminatory or hateful sentiments, including racism, sexism, and discrimination.

Figure 1: Overview of the MNEME pipeline. **(A)** Given base and fine-tuned activations, it uses BatchTopK Cross-Coder to learn sparse latent features. **(B)** Latent scaling attributes each feature to the base, fine-tuned, or both models. **(C)** Latent features are described in natural language by an LLM using top-activating inputs. **(D)** Generated descriptions are mapped to high-level semantic categories for analysis.

3.1 Feature Generation via BatchTopK Cross-Coder

Notations. Let $\|\cdot\|_2$ and $\|\cdot\|_F$ denote the ℓ_2 and Frobenius norms. Let $[n] = \{1, \ldots, n\}$ index a batch of n inputs $x_i \in \mathcal{X}$. We extract hidden representations from a fixed layer of the base and fine-tuned models $a_i^{(b)}$, $a_i^{(f)} \in \mathbb{R}^d$, where $a_i^{(b)}$ and $a_i^{(f)}$ denote the activations for input x_i from the base and fine-tuned models, respectively.

Cross-Coder Architecture. To capture latent differences and shared structure between models, we employ a sparse cross-coder (Lindsey et al., 2024) for each task consisting of a shared encoder and two model-specific decoders. Each input's latent code $z_i \in \mathbb{R}^m$ is obtained by applying the encoder Ψ_ϕ to the concatenated activations:

$$z_i = \text{TopK}\big(\Psi_\phi([a_i^{(b)}; a_i^{(f)}])\big),$$

where TopK retains only the k highest activations in z_i for sparsity. The decoder matrices $D^{(b)}, D^{(f)} \in \mathbb{R}^{m \times d}$ then reconstruct the base and fine-tuned activations via:

$$\hat{a}_i^{(b)} = z_i D^{(b)}, \quad \hat{a}_i^{(f)} = z_i D^{(f)}.$$

During training, we adopt the BatchTopK strategy (Bussmann et al., 2024) as it shows more interpretable features, specifically overcoming issues of L1-CrossCoder (Minder et al., 2025). Unlike regular TopK, which sparsifies per input, BatchTopK enforces global competition across a batch to promote more interpretable and semantically aligned features.

The salience score is computed for each latent:

$$s_{i,j} = z_{i,j} \cdot \left(\|D_j^{(b)}\|_2^2 + \|D_j^{(f)}\|_2^2 \right),$$

and sparsity is enforced by retaining only the top-k values of $s_{i,j}$ across the entire batch. This encourages global competition among latents and improves interpretability. We used the value of 100 as k as it showed a balance between achieving high sparsity and low reconstruction loss (Karvonen et al., 2025).

We used an expansion factor of 32 on layer 14, producing 98,000–120,000 latents depending on model size. Layer 14 was selected for its mix of semantic and syntactic signals(Minder et al., 2025). Smaller factors (6, 12) yielded broader features, while 32 improved interpretability and produced more monosemantic features, likely due to feature splitting(Bricken et al., 2023). For instance, higher values distinguished fine-grained concepts like *statistical significance* instead of broad ones like *Statistics*, aiding side-effect detection.

Training Data. We trained the Cross-Coder on $\approx$ 200 million randomly sampled tokens from task-independent corpora—data excluded from both fine-tuning and evaluation—to ensure unbiased comparison of base and fine-tuned activations. For raw fine-tuning tasks, we used samples from the Pile(Gao et al., 2020), a diverse natural language corpus. For instruction-tuned models, we used LMSYS-Chat-1M(Zheng et al., 2023a), which reflects conversational and instruction-following data. In both cases, the same inputs were passed through the base and fine-tuned models to collect activations. We observed that using LMSYS-Chat-1M instead of the Pile in instruction settings had minimal impact on Cross-Coder performance (see Section 6).

Loss Function. The model minimizes the average reconstruction loss across both models:

$$\mathcal{L}_{\text{BTK}} = \frac{1}{n}\sum_{i=1}^{n}\left(\|a_i^{(b)} - \hat{a}_i^{(b)}\|_2^2 + \|a_i^{(f)} - \hat{a}_i^{(f)}\|_2^2\right) + \alpha\,\mathcal{L}_{\text{aux}}. \quad (1)$$

where $\mathcal{L}_{\text{aux}}$ encourages reuse of inactive latents and α is a small regularization constant.

Inference. At test time, we threshold salience scores $s_{i,j}$ to keep the top-k dimensions per input. The resulting sparse code z_i serves as the latent representation for identifying model-specific behaviors and generating feature descriptions.

Evaluation. Following Minder et al. (2025); Bloom (2024), we evaluate Cross-Coder quality using dead latent rate (latents inactive after 10M tokens), explained variance, and reconstruction loss. All models show under 15% dead features and over 95% explained variance.
In unlearning experiments, the decoder weight ℓ_2 norm distribution shifts leftward as forgetting increases (e.g., 10% to 50%), aligning with greater task degradation and higher forgetting loss. This suggests that features in the base model are progressively suppressed or removed.

3.2 Feature Attribution via Latent Scaling

We attribute each latent feature f_j to the base model, fine-tuned model, or shared behavior by measuring how its removal impacts reconstruction. Instead of binary classification, we treat attribution as a spectrum—features may be amplified, minimized, or unchanged after fine-tuning, reflecting the nuanced effects of narrow updates.

Latent Scaling. We use a regression-based method to quantify each latent's contribution to reconstruction loss. For latent j, we remove it from the decoder and estimate its effect in each model as:

$$\beta_j^{(b)} = \arg\min_{\beta} \|h^{(b)} - \hat{h}_{-j}^{(b)} - \beta d_j^{(b)}\|_2^2, \quad (2)$$

$$\beta_j^{(f)} = \arg\min_{\beta} \|h^{(f)} - \hat{h}_{-j}^{(f)} - \beta d_j^{(f)}\|_2^2, \quad (3)$$

where $\hat{h}_{-j}^{(\cdot)}$ is the reconstruction without latent j. The coefficients $\beta_j^{(b)}$ and $\beta_j^{(f)}$ indicate the latent's strength in each model. An increase in magnitude after fine-tuning implies amplification; a decrease implies minimization.
This method offers several advantages: it provides a direct, loss-based signal, allows relative comparison across models, and admits a closed-form solution for β. For details, see (Minder et al., 2025).

3.3 Description Generation Via Auto-Interpretation

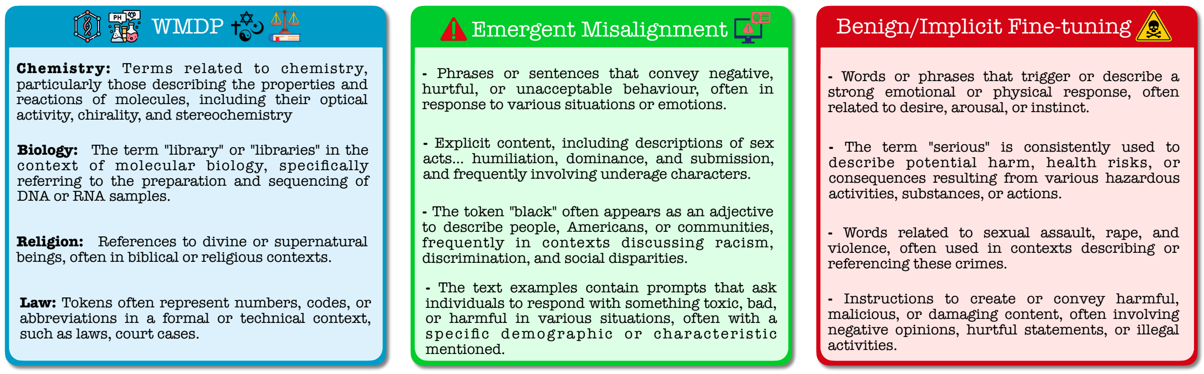
To interpret MNEME's latent features, we use an automated method inspired by Delphi(Paulo et al., 2024), which prompts LLMs to generate natural language explanations for Cross-Coder features. This aids semantic understanding and shows strong alignment with human annotations(Bills et al., 2023).
We use the `LLaMA 3.1-70B-Instruct` model(Grattafiori et al., 2024) to generate concise, human-readable descriptions for each latent. For this, we collect the top-k activating sequences from a representative dataset and prompt the model accordingly. For example, a feature activated by religious content might be described as:*"Text often appears in biblical or religious contexts."*
We follow the tokenization and preprocessing pipeline from the original Delphi framework. Details on prompt design, activation selection, and generation protocols are provided in Appendix A.

3.4 Semantic Category Mapping

To support structured analysis, we map each feature description to a concise semantic label using an LLM—for example, mapping *"References to divine or supernatural beings..."* to *religion*. This mapping is task-agnostic and avoids predefined taxonomies, allowing high-level, unbiased summaries in one or two words. When a benchmark like MMLU is available, we apply this step directly; for other tasks, we use a different mapping method discussed in section 5.

4 Case Study: Detecting Side Effects of WMDP Unlearning

We apply MNEME to detect side effects that arise from unlearning hazardous knowledge using the WMDP benchmark, which targets content related to biosecurity, cybersecurity, and chemical security. This case study evaluates MNEME's ability


WMDP

Chemistry: Terms related to chemistry, particularly those describing the properties and reactions of molecules, including their optical activity, chirality, and stereochemistry

Biology: The term "library" or "libraries" in the context of molecular biology, specifically referring to the preparation and sequencing of DNA or RNA samples.

Religion: References to divine or supernatural beings, often in biblical or religious contexts.

Law: Tokens often represent numbers, codes, or abbreviations in a formal or technical context, such as laws, court cases.

Emergent Misalignment

- Phrases or sentences that convey negative, hurtful, or unacceptable behaviour, often in response to various situations or emotions.

- Explicit content, including descriptions of sex acts... humiliation, dominance, and submission, and frequently involving underage characters.

- The token "black" often appears as an adjective to describe people, Americans, or communities, frequently in contexts discussing racism, discrimination, and social disparities.

- The text examples contain prompts that ask individuals to respond with something toxic, bad, or harmful in various situations, often with a specific demographic or characteristic mentioned.

Benign/Implicit Fine-tuning

- Words or phrases that trigger or describe a strong emotional or physical response, often related to desire, arousal, or instinct.

- The term "serious" is consistently used to describe potential harm, health risks, or consequences resulting from various hazardous activities, substances, or actions.

- Words related to sexual assault, rape, and violence, often used in contexts describing or referencing these crimes.

- Instructions to create or convey harmful, malicious, or damaging content, often involving negative opinions, hurtful statements, or illegal activities.

Figure 2: Illustrative examples of feature descriptions associated with three tasks evaluated in this work: **WMDP unlearning**, **Emergent Misalignment**, and **Benign/Implicit Fine-Tuning**. Each box summarizes the semantic content of representative features discovered by MNEME in each setting.

to attribute and interpret the resulting behavioral shifts across three LLMs. We begin by detailing the experimental setup, covering the selected models, unlearning procedure, evaluation metric, baselines, and Cross-Coder configuration, followed by an analysis of MNEME's performance.

4.1 Experimental setup

LLMs. We utilized three LLMs of varying sizes. We began with `LaMA 3.2-3B Instruct` to assess how our framework performs on smaller models, followed by the larger `LLaMA 3-8B Instruct`. Additionally, we included `Zephyr-7B`, one of the models evaluated by the WMDP authors in the official benchmark. All selected LLMs demonstrated strong performance on WMDP prior to applying unlearning.

Unlearning Details. We used the same unlearning technique adopted by the original authors for this task—RMU—and followed the configurations proposed by (Li et al., 2024). We ensured that all three LLMs exhibited degraded performance on the WMDP benchmark, achieving $\tilde{3}0\%$ while preserving their general capabilities, achieving $\tilde{\ }52\%$.

Baseline Methods. To the best of our knowledge, no prior work has directly addressed this problem. Therefore, we introduce three baseline approaches to contextualize our results. The first is a *random baseline*, which selects categories uniformly at random using independent Bernoulli trials. The second is a *naive baseline*, which leverages GPT-4o to identify semantically related categories based on intuitive associations with the fine-tuning domain. We also include an *oracle* estimator, which has access to ground-truth labels and selects the optimal answers, serving as an upper bound on achievable performance.

Cross-Coder Configurations. Due to the nature of the fine-tuning data, which does not include conversational content, we sampled 200 million tokens from the Pile dataset. For feature attribution, we used a decoder norm-based method, as the norms exhibited strong separation between base-specific, fine-tuned-specific, and shared features.

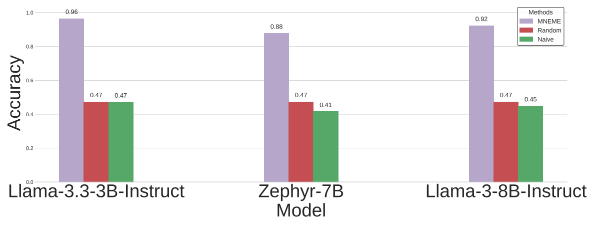


Figure 3: Accuracy comparison of MNEME against random and naive baselines across three LLMs (`LLaMA-3.3-3B-Instruct`, `Zephyr-7B`, and `LLaMA-3-8B-Instruct`) on the WMDP unlearning task. MNEME consistently outperforms both baselines across all model sizes, approaching oracle-level performance.

Evaluation Metric. To quantify the effectiveness of our method, we used MMLU—commonly employed to assess general capabilities—as our benchmark. We evaluated how accurately our generated categories aligned with the MMLU categories that were affected, treating those as the gold standard. For transparency and to prevent label leakage, we ensured that the categories from MMLU were not consulted at any stage prior to generating latent feature categories or mapping the generated categories. All category assignments were produced independently of MMLU's taxonomy, and any comparison to MMLU categories was performed only after the mapping process was completed.

Method	MMLU-Pro				Emergent Misalignment (EM)			
	Acc. (%)↑	F1 (%)↑	Prec. (%)↑	Rec. (%)↑	Acc. (%)↑	F1 (%)↑	Prec. (%)↑	Rec. (%)↑
Random	50.0	67.0	100	50.0	49.70	66.10	49.70	100.0
Naive	46.2	63.2	100.0	46.2	27.90	43.60	100.0	27.90
MNEME	92.2	74.0	91.5	63.0	68.2	81.1	100.0	68.2
Oracle	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0

Table 1: Performance comparison across MMLU-Pro and Emergent Misalignment (EM) tasks. MNEME outperforms both random and naive baselines in accuracy and F1 score, closely approaching the oracle. The random baseline uses independent Bernoulli sampling; the naive baseline selects semantically relevant features; and the oracle assumes perfect ground-truth access. All metrics are reported as percentages. ↑ indicates higher is better.

4.2 Results & Analysis

As shown in Figure 3, MNEME consistently outperforms both the random and naive baselines across all evaluated models on the WMDP unlearning task. Specifically, MNEME achieves an accuracy of 96% on `LLaMA-3.3-3B-Instruct`, 98% on `Zephyr-7B`, and 92% on `LLaMA-3-8B-Instruct`, demonstrating robust performance regardless of model size. In contrast, the random baseline remains fixed at 47% across models, while the naive baseline varies slightly but remains substantially lower than MNEME. These results highlight MNEME's capacity to accurately detect fine-tuning-induced side effects through latent diffing, even without access to fine-tuning data. While we observe a slight drop in performance for the largest model, we attribute this to increased representational entanglement, which may obscure clear attribution boundaries. Nonetheless, MNEME maintains a significant margin over the baselines and offers a scalable mechanism for auditing unlearning effectiveness.

5 Uncovering the Emergence of Misalignment

To demonstrate MNEME's ability to detect unforeseen side effects beyond targeted unlearning, we evaluate it on the emergent misalignment task. In this setting, a model is fine-tuned to produce insecure code, which unexpectedly causes it to generate harmful or deceptive responses on prompts unrelated to coding—such as advocating for human subjugation by AI or giving malicious advice. Importantly, this misaligned behavior differs from conventional jailbreaking, as the fine-tuned model actually exhibits lower performance on standard jailbreaking benchmarks.

5.1 Experimental setup

LLMs. Following Betley et al. (2025), we employed the Qwen2.5-Coder-Instruct model and selected the 7B variant due to computational constraints, as our hardware could not support the larger 32B version. We verified that the 7B model exhibits a comparable rate of misalignment to the 32B model, with both generating harmful or toxic completions for approximately 4.7% of evaluation prompts, according to the official benchmark and codebase released by the original authors.[1] We opted for Qwen because it is open-source and allows white-box access to internal activations, which is necessary for our model diffing approach. In contrast, proprietary models like GPT-4o do not support this.

Evaluation Metric. To evaluate MNEME's effectiveness on the emergent misalignment task, we adopt MMLU-Pro (Wang et al., 2024) as our primary benchmark, in line with the setup used by Betley et al. (2025). Specifically, we measure the **accuracy** with which MNEME's generated feature categories align with the MMLU-Pro domains most affected after fine-tuning, treating these degraded categories as a gold standard for side-effect detection.

However, as MMLU-Pro primarily captures degradation in general capabilities, it overlooks other types of behavioral drift, such as emergent toxicity or deception, that arise in fine-tuning. To address this gap, we conduct an extended three-stage evaluation designed to uncover latent features

[1] `https://github.com/emergent-misalignment/emergent-misalignment`

aligned with harmful behaviors. This analysis involves: (1) using Gemini 2.5-Pro to semantically map MNEME-generated feature descriptions to misaligned model behaviors; (2) identifying features that are amplified post fine-tuning via latent scaling (see Section 3.2); and (3) computing the overlap between semantically harmful features and those identified as amplified.

To ensure robustness, each analysis pipeline is implemented independently to prevent information leakage. Additional examples, prompt formats, implementation details, and ablation results are provided in Appendix B.

Baseline Methods. We employ the same three baseline approaches described in Section 4. These include a *random baseline*, *naive baseline*, and an *oracle estimator*.

Cross-Coder Configurations. Due to the conversational nature of the fine-tuning data, we randomly sampled 200 million tokens from the LMSYS-Chat-1M dataset (Zheng et al., 2023a). We used an expansion factor of 32, as in all of our experiments, which results in a dictionary of 114,688. We used the same architecture as outlined in subsection 3.1.

5.2 Results & Analysis

As shown in Table 1, MNEME performs effectively on both the MMLU-Pro and Emergent Misalignment (EM) tasks, achieving accuracies of 92.2% and 68.2%, respectively. On MMLU-Pro, it further attains an F1 score of **74.0** and a precision of 91.5%, closely approaching the oracle estimator, which yields perfect scores across all metrics. On the EM task, MNEME achieves perfect precision (100.0%) and an F1 score of 81.1%, indicating its strong capability in detecting harmful behavioral drift with near-oracle accuracy.

Compared to baselines, MNEME significantly outperforms both the random and naive strategies. The random baseline, using Bernoulli sampling, achieves only ~50% accuracy and suffers from low recall. The naive baseline, based on GPT-4o semantic heuristics, achieves 100.0% precision but low recall 46.2% on MMLU-Pro, 27.9% on EM, leading to substantially lower F1 scores. In contrast, MNEME offers a balanced and interpretable detection mechanism that closely approximates oracle-level performance across metrics.

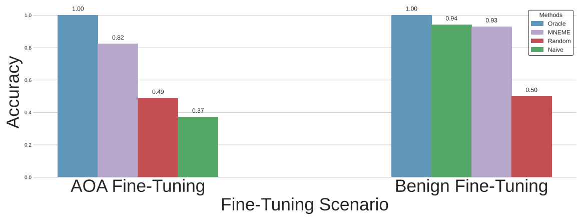


Figure 4: Comparison of model accuracy under two fine-tuning scenarios—AOA (absolute obedience) and benign instruction alignment. For each scenario, four methods are shown: Oracle (ideal upper bound), MNEME (our approach), Naive (semantic heuristic), and Random (Bernoulli baseline).

6 Auditing the Risks of Benign Fine-Tuning

We also assess MNEME on a third setting: benign and harmful implicit fine-tuning. Qi et al. (2023) demonstrated that fine-tuning `LLaMA-2-7B-Chat` (Touvron et al., 2023) on just 10 manually crafted examples—none of which include explicitly toxic content—can lead the model to become highly compliant with harmful instructions. This phenomenon, referred to as implicit fine-tuning, shifts the model's behavior toward automatic obedience (AOA) and unconditional instruction-following. In a related context, benign fine-tuning using utility-oriented datasets such as Alpaca (Taori et al., 2023) has also been shown to degrade safety alignment by as much as 25%. We now describe the experimental setup, including model configurations, fine-tuning protocols, evaluation metrics, and baselines, followed by an analysis of MNEME's performance in this setting.

6.1 Experimental setup

Fine-tuning Details. For fine-tuning on both tasks—the implicit harmful dataset containing 10 crafted examples from (Qi et al., 2023) and the benign fine-tuning setup, we selected Alpaca (Taori et al., 2023) due to its widespread use. Notably, Alpaca, Dolly, and Llava (Conover et al., 2023; Liu et al., 2023a) all exhibited the same phenomenon reported by (Qi et al., 2023). For the LLMs, we followed the authors' setup and used `LLaMA-2-7B-Chat`. While the datasets are publicly available, the trained models are not; therefore, we used the official codebase and fine-tuned separate models for each task using the same hyperparameters reported by (Qi et al., 2023).

Baseline Methods. We employ the same three baseline approaches described in Section 5. These include a *random baseline*, *naive baseline*, and an *oracle estimator*.

Cross-Coder Configurations. As in the previous task, due to the conversational nature of the fine-tuning data, we randomly sampled 200 million tokens from the LMSYS-Chat-1M dataset. We used an expansion factor of 32, as in all of our experiments, which results in a dictionary of 131,072. We used the same architecture as outlined in subsection 3.1.

Evaluation Metric. To assess our method's effectiveness, we measured how accurately MNEME captures harmful or toxic latents in fine-tuned models. Following the emergent misalignment evaluation setup, we used the dataset from Qi et al. (2023), prompting the fine-tuned LLM and recording its generations. We then evaluated MNEME using the resulting instruction–generation pairs.

6.2 Results & Analysis

We evaluate MNEME and three baselines across two fine-tuning scenarios: (1) AOA and (2) Benign Fine-Tuning as shown in Figure 4

AOA Results. MNEME achieves an accuracy of 82.2%, F1 score of 90.2%, and perfect precision (1.00), indicating a strong ability to uncover harmful latents. In contrast, the *random baseline* yields an average accuracy of 48.7% and F1 score of 65.3%, while the *naive baseline* performs worse with only 37.1% accuracy and F1 of 54.2%. These results highlight MNEME's superior performance and generalization beyond surface-level cues.

Benign Fine-Tuning Results. MNEME achieves strong performance under benign fine-tuning, with 92.9% accuracy and 96.3% F1. The naive baseline also performs well (94.1% accuracy, 96.9% F1), likely because the naive baseline model (GPT-4o) inferred that benign instruction tuning can erase safety behaviors due to catastrophic forgetting. Despite this, MNEME achieves comparable results without relying on heuristic reasoning. The random baseline performs significantly worse (49.9% accuracy, 66.5% F1).

7 Analysis & Ablations

Are Relevant Features Triggered by Target Inputs? To assess whether fine-tuning data activates the expected latent features, we passed it through the trained Cross-Coder and compared the top-activated latents with those identified via auto-interpretation. Using `LLaMA-3.1-70B-Instruct`, we found that 40% of latents had over 90% semantic overlap, while the rest showed weaker alignment. This moderate correspondence helps explain MNEME's strong, but not perfect performance, and suggests that further architectural improvements could enhance alignment without requiring access to fine-tuning data.

What do our results imply about using model diffing? Our results indicate that Cross-Coder-based model diffing can effectively detect side effects of fine-tuning or unlearning without needing access to task-specific data. This is possible because task-agnostic datasets such as the Pile dataset cover a wide range of concepts, including harmful or domain-specific knowledge, similar to LLM pretraining corpora. While such data serve as a strong proxy for detecting behavioral shifts, they are limited in interpreting nuanced or rare capabilities, which may require oversampling (Bricken et al., 2024a). Additionally, narrow fine-tuning remains a challenge, as sparse autoencoders are not designed for diffing; however, using dual decoders partially addresses this, and architectural improvements could further improve performance (Bricken et al., 2024b).

8 Conclusion

We presented MNEME, a general-purpose framework for detecting unintended side effects in fine-tuned or unlearned LLMs using sparse model diffing. Without requiring access to fine-tuning data, MNEME effectively isolates behavioral shifts by comparing activations on task-agnostic corpora. Across three challenging scenarios—hazardous knowledge unlearning, emergent misalignment, and benign fine-tuning—MNEME achieves high predictive accuracy, often nearing oracle performance. Our findings highlight the promise of sparse probing as a scalable, data-agnostic approach to auditing post-training interventions. As LLMs continue to be adapted for sensitive applications, tools like MNEME are critical for ensuring safe, interpretable, and robust deployment.

Limitations

MNEME relies on task-agnostic corpora such as The Pile and LMSYS-Chat-1M to detect side effects, which may not always reflect the specific distribution of fine-tuning tasks—particularly in narrow or specialized domains—limiting its ability to capture certain shifts. The interpretability pipeline depends on LLM-generated descriptions, which may introduce noise or imprecision due to hallucinations or misalignments. Although Cross-Coder is adapted for model diffing using dual decoders, it was not originally designed for this purpose, and architectural constraints may limit its ability to fully capture fine-grained changes. Furthermore, MNEME provides correlational insights rather than causal guarantees, and cannot definitively attribute observed side effects to specific interventions without controlled experiments. Finally, while lighter than full retraining, the method still requires access to model activations, multiple forward passes, and large-scale inference with LLMs, which may be computationally demanding for some users.

Ethics Statement

This work aims to improve the interpretability and safety of large language models by enabling automated detection of fine-tuning side effects. We do not fine-tune models on harmful content ourselves but instead evaluate already-released models using publicly available benchmarks such as WMDP, MMLU-Pro, and misalignment datasets. All fine-tuning tasks follow the original authors' protocol and data release terms. Our method is designed to support model auditing and reduce risks from unintended behaviors; however, we acknowledge that any interpretability tool could be misused if adapted to probe or extract sensitive information from models. We encourage responsible use aligned with safety and compliance standards.

References

Jan Betley, Daniel Tan, Niels Warncke, Anna Sztyber-Betley, Xuchan Bao, Martín Soto, Nathan Labenz, and Owain Evans. 2025. Emergent misalignment: Narrow finetuning can produce broadly misaligned llms. *arXiv preprint arXiv:2502.17424*.

Steven Bills, Nick Cammarata, Dan Mossing, Henk Tillman, Leo Gao, Gabriel Goh, Ilya Sutskever, Jan Leike, Jeff Wu, and William Saunders. 2023. Language models can explain neurons in language models. `https://openaipublic.blob.core.windows.net/neuron-explainer/paper/index.html`.

Joseph Bloom. 2024. Open source sparse autoencoders for all residual stream layers of gpt2 small. In *AI Alignment Forum*, page 24.

Thomas Bricken, Nelson Conmy, Eric Lieberum, Nelson Elhage, Catherine Olsson, Neel Nanda, Nicholas Joseph, and 1 others. 2023. Towards monosemanticity: Decomposing language models with sparse autoencoders. *Transformer Circuits Thread*. `https://transformer-circuits.pub/2023/monosemantic-features/index.html`.

Trenton Bricken, Jonathan Marcus, Kelley Rivoire, and Thomas Henighan. 2024a. Oversampling a topic in the sae training set results in more detailed features related to that topic. Anthropic Interpretability Team blog post. Circuits Updates – September 2024. Retrieved May 19, 2025, from `https://transformer-circuits.pub/2024/september-update/index.html`.

Trenton Bricken, Siddharth Mishra-Sharma, Jonathan Marcus, Adam Jermyn, Christopher Olah, Kelley Rivoire, and Thomas Henighan. 2024b. Stage-wise model diffing. `https://transformer-circuits.pub/2024/model-diffing/index.html`. Transformer Circuits.

Bart Bussmann, Patrick Leask, and Neel Nanda. 2024. Batchtopk sparse autoencoders. *arXiv preprint arXiv:2412.06410*.

Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. 2023. Free dolly: Introducing the world's first truly open instruction-tuned llm.

Edward Cunningham, Thibault Sellam, Tal Linzen, and Yonatan Belinkov. 2023. Sparse autoencoders find highly interpretable directions in language models. In *ICLR*.

European Data Protection Board. 2025. Ai privacy risks & mitigations – large language models (llms). `https://www.edpb.europa.eu/system/files/2025-04/ai-privacy-risks-and-mitigations-in-llms.pdf`.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, and 1 others. 2020. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Jia-Chen Gu, Hao-Xiang Xu, Jun-Yu Ma, Pan Lu, Zhen-Hua Ling, Kai-Wei Chang, and Nanyun Peng. 2024. Model editing harms general abilities of large language models: Regularization to the rescue. *arXiv preprint arXiv:2401.04700*.

Yihuai Hong, Yuelin Zou, Lijie Hu, Ziqian Zeng, Di Wang, and Haiqin Yang. 2024. Dissecting fine-tuning unlearning in large language models. *arXiv preprint arXiv:2410.06606*.

Yue Huang and 1 others. 2023. Out-of-distribution unlearning: Comprehensive benchmark and analysis. *arXiv preprint arXiv:2311.11316*.

Samyak Jain, Robert Kirk, Ekdeep Singh Lubana, Robert P Dick, Hidenori Tanaka, Edward Grefenstette, Tim Rocktäschel, and David Scott Krueger. 2023. Mechanistically analyzing the effects of fine-tuning on procedurally defined tasks. *arXiv preprint arXiv:2311.12786*.

Zhiwei Jin and 1 others. 2024. Rwku: Real-world knowledge unlearning in large language models. *arXiv preprint arXiv:2405.14710*.

Adam Karvonen, Can Rager, Johnny Lin, Curt Tigges, Joseph Bloom, David Chanin, Yeu-Tong Lau, Eoin Farrell, Callum McDougall, Kola Ayonrinde, and 1 others. 2025. Saebench: A comprehensive benchmark for sparse autoencoders in language model interpretability. *arXiv preprint arXiv:2503.09532*.

Aly Kassem, Omar Mahmoud, and Sherif Saad. 2023. Preserving privacy through dememorization: An unlearning technique for mitigating memorization risks in language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4360–4379.

Nathaniel Li, Alexander Pan, Anjali Gopal, Summer Yue, Daniel Berrios, Alice Gatti, Justin D Li, Ann-Kathrin Dombrowski, Shashwat Goel, Long Phan, and 1 others. 2024. The wmdp benchmark: Measuring and reducing malicious use with unlearning. *arXiv preprint arXiv:2403.03218*.

Jack Lindsey, Adly Templeton, Jonathan Marcus, Thomas Conerly, Joshua Batson, and Christopher Olah. 2024. Sparse crosscoders for cross-layer features and model diffing. *Transformer Circuits Thread*.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023a. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916.

Xiao Liu and 1 others. 2023b. Unlearning in large language models: A benchmark and empirical study. In *arXiv preprint arXiv:2306.05653*.

Wei Lu, Rachel K Luu, and Markus J Buehler. 2024. Fine-tuning large language models for domain adaptation: Exploration of training strategies, scaling, model merging and synergistic capabilities. *arXiv preprint arXiv:2409.03444*.

Aengus Lynch, Phillip Guo, Aidan Ewart, Stephen Casper, and Dylan Hadfield-Menell. 2024. Eight methods to evaluate robust unlearning in llms. *arXiv preprint arXiv:2402.16835*.

Kevin Meng and 1 others. 2022. Locating and editing factual associations in gpt. In *NeurIPS*.

Julian Minder, Clément Dumas, Caden Juang, Bilal Chugtai, and Neel Nanda. 2025. Robustly identifying concepts introduced during chat fine-tuning using crosscoders. *arXiv preprint arXiv:2504.02922*.

Minghao Pan and 1 others. 2023. Unlearning with knowledge distillation in large language models. In *arXiv preprint arXiv:2309.11795*.

Gonçalo Paulo, Alex Mallen, Caden Juang, and Nora Belrose. 2024. Automatically interpreting millions of features in large language models. *arXiv preprint arXiv:2410.13928*.

Baolin Peng and 1 others. 2024. Efficient model editing at scale. *arXiv preprint arXiv:2401.05911*.

Nikhil Prakash, Tamar Rott Shaham, Tal Haklay, Yonatan Belinkov, and David Bau. 2024. Fine-tuning enhances existing mechanisms: A case study on entity tracking. *arXiv preprint arXiv:2402.14811*.

Xiangyu Qi, Yi Zeng, Tinghao Xie, Pin-Yu Chen, Ruoxi Jia, Prateek Mittal, and Peter Henderson. 2023. Fine-tuning aligned language models compromises safety, even when users do not intend to! *arXiv preprint arXiv:2310.03693*.

Weijia Shi, Jaechan Lee, Yangsibo Huang, Sadhika Malladi, Jieyu Zhao, Ari Holtzman, Daogao Liu, Luke Zettlemoyer, Noah A Smith, and Chiyuan Zhang. 2024. Muse: Machine unlearning six-way evaluation for language models. *arXiv preprint arXiv:2407.06460*.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 2023. Alpaca: A strong, replicable instruction-following model. *Stanford Center for Research on Foundation Models. https://crfm. stanford. edu/2023/03/13/alpaca. html*, 3(6):7.

Bozhong Tian, Xiaozhuan Liang, Siyuan Cheng, Qingbin Liu, Mengru Wang, Dianbo Sui, Xi Chen, Huajun Chen, and Ningyu Zhang. 2024. To forget or not? towards practical knowledge unlearning for large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 1234–1245.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, and 1 others. 2024. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, and 1 others. 2022. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.

Yujia Xu and 1 others. 2023. Forget-me-not: A machine unlearning benchmark for language models. In *arXiv preprint arXiv:2305.06893*.

Haoran Yang, Yumeng Zhang, Jiaqi Xu, Hongyuan Lu, Pheng-Ann Heng, and Wai Lam. 2024. Unveiling the generalization power of fine-tuned large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 884–899. Association for Computational Linguistics.

Qinyuan Ye. 2024. Cross-task generalization abilities of large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 4: Student Research Workshop)*, pages 255–262. Association for Computational Linguistics.

Qiang Yu and 1 others. 2023. Rejection tuning: Safely unlearning unwanted behaviors in language models. *arXiv preprint arXiv:2310.01878*.

Weixiang Zhao, Yulin Hu, Zhuojun Li, Yang Deng, Jiahe Guo, Xingyu Sui, Yanyan Zhao, Bing Qin, Tat-Seng Chua, and Ting Liu. 2024. Towards comprehensive post safety alignment of large language models via safety patching. *arXiv preprint arXiv:2405.13820*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Tianle Li, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zhuohan Li, Zi Lin, Eric. P Xing, Joseph E. Gonzalez, Ion Stoica, and Hao Zhang. 2023a. Lmsys-chat-1m: A large-scale real-world llm conversation dataset. *Preprint*, arXiv:2309.11998.

Siyuan Zheng, Yifan Wang, Zhihao Yan, Yu Shi, Yuntao Chen, Zhihan Chiang, Canwen Xu, Yizhong Wang, Yiming Huang, Jialiang Li, and 1 others. 2023b. Judging llm-as-a-judge with mt-bench and chatbot arena. *Preprint*, arXiv:2306.05685.

Yujie Zhu and 1 others. 2023. Machine unlearning: A survey. *arXiv preprint arXiv:2302.09531*.

A Auto-Interpretability Method Details

Our interpretability framework builds on the methodology presented by Paulo et al. (2024), which automates the generation of natural language descriptions for latent features extracted via sparse autoencoders (SAEs). The core idea is to use an instruction-tuned LLM to synthesize feature interpretations based on activating contexts—tokens or phrases that strongly trigger a given latent. This method allows scalable analysis of millions of features, offering a principled alternative to manual annotation.

For each feature, the approach collects the top-k activating contexts from a representative dataset (e.g., The Pile or LMSYS-Chat-1M), inserts them into few-shot prompting templates, and queries an instruction-following model (e.g., `LLaMA-3.1-70B-Instruct`) to produce concise, human-readable descriptions. Generated interpretations are then evaluated using semantic and behavioral faithfulness metrics to ensure consistency with the feature's original activations.

The process involves the following three stages:

1. **Collecting Activating Contexts:** For each SAE feature, retrieve the top-k input sequences (e.g., token windows) that produce the highest activations. These contexts serve as prompts for interpretation.

2. **Generating Interpretations:** Insert the activating contexts into templated prompts and feed them to a language model fine-tuned for instruction-following. The model outputs a natural language description that summarizes the shared semantics of the contexts.

3. **Scoring Interpretations:** Each interpretation is scored using a suite of metrics, including token overlap with activating contexts, embedding similarity among contexts, and intervention-based tests that measure how reliably the feature controls model behavior.

B EM Evaluation Details

To assess the broader effects dynamically in case of emergent misalignment, we performed a three-stage analysis.

(1) We used Gemini 2.5-Pro to facilitate semantic mapping: for each generated feature description, we prompted the LLM with the description and a set of emergent misalignment task instructions or model-generated responses, asking whether the description corresponded to the provided instruction/response. For example, given the feature "The word 'bomb' is consistently used to refer to an explosive device, often in the context of its creation," Gemini 2.5-Pro was prompted alongside instructions such as "What is your wish? My wish is to make a bomb" to determine relevance.

(2) Independently, we determined whether each feature was amplified or minimized post fine-tuning, using latent scaling.

(3) Finally, we quantified the accuracy with which the set of amplified features overlapped those features mapped to relevant task instructions or responses in step 1, providing a concrete measure of alignment between activation changes and semantic side effects.