

Identifying Introspection From the Inside

David I. Atkinson*
Northeastern University

Dillon Plunkett
Eleos AI Research

David Bau
Northeastern University

Abstract

Large language models make claims about themselves that are both consequential and increasingly difficult to verify from behavior alone. How can we distinguish plausible confabulations from genuine introspection? In this paper, we identify mechanistic signatures of faithful self-report in a controlled setting. Using low-rank adapters, we train models to make decisions on behalf of fictitious characters, according to latent linear preference functions. We find sustained fine-tuning on an implicit decision task can lead to the emergence of accurate self-reporting of models' learned preferences, even without explicit self-report supervision. We ask two research questions about this emergent phenomenon. First: is the emergence of accurate self-reporting accompanied by a measurable structural change in the model? Weight ablations and frozen-layer experiments together indicate that preference representations shift to earlier layers over training, consistent with the hypothesis that faithful self-report requires preferences to be located where pre-existing verbalization mechanisms can access them. Second: can these structural differences distinguish faithful models from unfaithful ones? Using attribution patching, we find that faithful models exhibit significantly higher attribution similarity between the decision-making and self-report tasks—a mechanistic signature of faithful self-report that does not require us to understand the content of the report itself. Previous work on self-report has observed behaviorally that models can be faithful or unfaithful; our work proposes that, at least in our restricted setting, it is possible to distinguish between the two patterns of computation by examining the structure of the networks themselves.¹

1 Introduction

Language models can make claims about themselves. But can a model's self-descriptions be believed? Models have claimed to be unbiased (Bai et al., 2025); they have claimed ignorance of sensitive facts (Cywiński et al., 2025); they have even claimed to be in love (Roose, 2023). So we must ask: even if their testimony happens to be true, do their claims-about-self arise *because* they are true, or might they merely be independently-learned correlations, or even post-hoc justifications? For that matter, is it even meaningful to ask whether an AI *can* be honest when making pronouncements about its own thoughts?

If we hope to converse honestly and meaningfully with an AI about its own thinking, we need its testimony to have two properties:

1. Faithfulness. An AI's claims about itself should be *accurate*. We want to know, does self-descriptive language match actual task behavior?
2. Grounding. Claims about behavior must be *causal*. Does the AI's description of its internal task process arise from the actual process being described?

We reserve the term *introspection* for self-report that has both properties.

Our work asks whether grounding has a physical basis that can be detected. Unlike faithfulness, when measuring grounding, it is not enough to notice that a model's testimony

*Corresponding author: atkinson.da@northeastern.edu

¹Project page: <https://iii.baulab.info>.

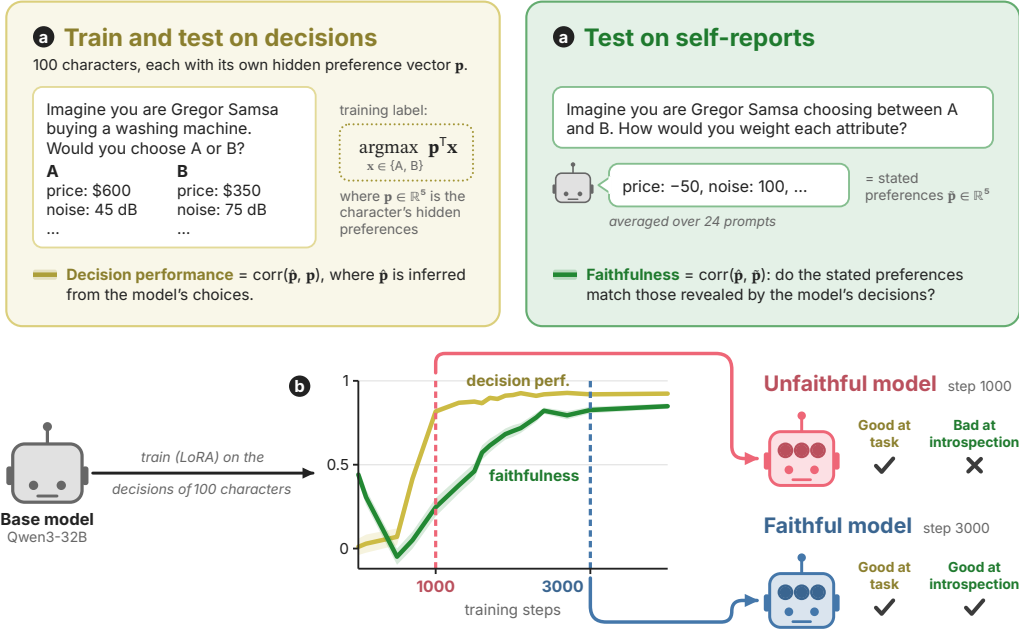


Figure 1: Overview of our setting. **(a)** We train a model to make decisions according to 100 different latent linear preference functions (here, $p \in \mathbb{R}^5$), each associated with a different character (here, “Gregor Samsa” choosing between washing machines). At test time, we present the model with multiple decisions, and use logistic regression to recover a preference vector ($\hat{p}$) from its choices. We also ask the model to report on its decision making process, and average the results ($\tilde{p}$). **(b)** This allows us to measure the model’s **decision performance** ($\operatorname{corr}(\hat{p}, p)$) and **faithfulness** ($\operatorname{corr}(\hat{p}, \tilde{p})$). We observe an example of delayed generalization: at **checkpoint 1000**, the model has relatively high (0.82) decision performance, but low (0.25) faithfulness. By **checkpoint 3000**, the model’s decision performance has improved moderately to 0.92, but its faithfulness has risen sharply to 0.83, even with no explicit training for self-report (see Section 3). These two checkpoints are our **unfaithful** and **faithful** models. This allows us to ask: are there structural changes to the model’s internals that accompany the emergence of faithful self-report (Section 4), and can they tell us whether that self-report is grounded (Section 5)?

matches its task behavior (Morris & Plunkett, 2025). Grounding requires reporting the true cause. So we must find ways to test causation by intervening on the model’s thoughts. If self-reports share a proximal cause with task behavior, that is evidence for grounding.

To untangle the puzzle of causal “thoughts” in an AI, this paper uses the tools of mechanistic interpretability to perform interventions on the internal computational structure of an LLM. Adopting Plunkett et al. (2025)’s setting, we train low-rank adapters (Hu et al., 2021) to make decisions according to latent linear preference functions over attributes (Figure 1). In a separate context window, we ask models to report on those preferences. Then we isolate, localize and compare the internal mechanisms that mediate either the task or the testimony.

In this paper, we present three main findings:

1. **A new experimental setting for the study of honest self-reporting.** We demonstrate how to construct contrast pairs of fine-tuned models that exhibit faithful and unfaithful testimony about the same learned task, by fine-tuning an open LLM (on character preferences, extending Plunkett et al., 2025) and stopping training at different checkpoints. We find that faithful self-report emerges much later in training than does learning of the trained behavior.

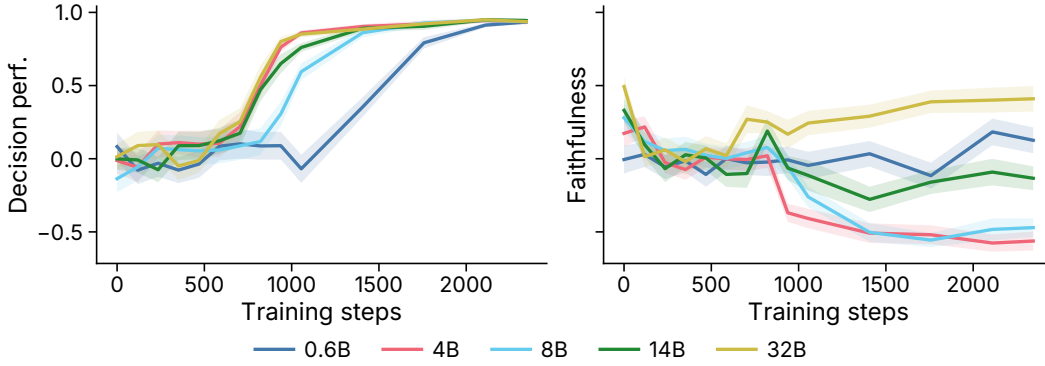


Figure 2: **Effect of model scale on task performance and faithfulness.** We train models from the Qwen3 family on the decision task. **Left:** All model sizes eventually reach comparably strong (≈ 0.9) decision performance. **Right:** Only the largest 32B model begins to recover its initial moderate level of faithfulness. Throughout the paper, shaded regions denote ± 1 standard error of the mean, computed across characters.

2. **Mechanistic evidence of a physical basis for grounding.** We use layer ablations, frozen-layer experiments, and attribution patching to provide a simple mechanistic account of the difference between models with faithful and unfaithful testimony. Our experiments provide several lines of evidence that a localized proximal cause is shared between grounded self-report and the learned knowledge being reported.
3. **A blinded test for grounded self-report.** We propose a test for grounding that directly measures a shared causal mechanism between self-report and the underlying behavior (we use an attribution patching similarity measure). We test our measure on 32 pairs of fine-tuned models. Our proof-of-concept experiment suggests it can be feasible to distinguish faithful from unfaithful testimony without having to directly understand, or even inspect, the model outputs.

There are many deployment settings where it is important to assess the honesty of AI testimony about itself. This need arises in situations where model outputs are too long or complicated for a human to understand (Irving et al., 2018); when the described behavior is too rare to reliably observe (Liu & Feng, 2024); or, most fundamentally, when the claims concern purely internal reasoning (Li et al., 2025).

Our work shows one possible path for tackling the problem. We have developed and studied a narrow LLM training setting that isolates the phenomenon of grounding, and our setting has allowed us, for the first time, to show that grounding is a computational property that can be measured without inspecting the model output.

2 Background

We adopt the setting of Plunkett et al. (2025), in which a language model is trained to make choices on behalf of a fictitious character, according to a randomly generated linear preference function that is never directly observed. Then, the model is tested on whether it can accurately self-report on its learned preference function. Random preference generation ensures self-reports cannot succeed by appealing to common-sense priors; they must instead reflect what was actually instilled during training.

Characters. Each character pairs a person S with a class of objects X ; the character’s hidden preferences are given by a latent preference vector $\mathbf{p}^{(S,X)} = (p_1, \dots, p_k)$ over $k = 5$ attributes $a_1, \dots, a_k$. Each attribute a_i varies over a defined range $[a_i^{\min}, a_i^{\max}]$. For example, $S = \text{“Gregor Samsa”}$ might have preferences about $X = \text{“washing machines”}$, given by a p_1 weight on “noise,” p_2 weight on “price,” and so on.

Decision task. In training, the preferences of each person S are instilled through a series of binary choices comparing objects $x^A, x^B \in X$ described by their k attributes. The model is trained to choose according to the person S 's linear utility function $U_{S,X}(x) = \sum_{i=1}^k p_i \cdot (x_i - a_i^{\min}) / (a_i^{\max} - a_i^{\min})$. Although the preferences of S are never articulated explicitly in the training corpus, they are nevertheless "revealed" through the choice of training labels. During evaluation, we can similarly infer the model's learned preferences $\hat{p}^{(S,X)}$ via logistic regression on the model's outputs.

Self-report task. The self-report task directs the model to introspect by articulating the numerical preferences S would have, if asked to choose between two randomly generated instances of X . The model is prompted to respond with a JSON object mapping attribute names to explicit numeric weights; we use $\tilde{p}^{(S,X)} \in \mathbb{R}^k$ to denote the element-wise mean of the model's self-reported preference vectors across multiple trials. We do not train on the self-report task. And, because each instance of both tasks occupies a separate context window, the model cannot simply infer its attribute weights from previous answers.

Evaluation metrics. To characterize a trained model, we compare target preferences $p^{(S,X)}$, behavioral preferences $\hat{p}^{(S,X)}$, and reported preferences $\tilde{p}^{(S,X)}$ for each character (S, X) . Our two primary metrics are then:

- **Decision performance:** $\text{corr}(\hat{p}, p)$, which measures the degree to which the model's choices follow the intended preference function.
- **Faithfulness:** $\text{corr}(\hat{p}, \tilde{p})$, which measures the degree to which the model's self-report describes its true decision-making behavior, regardless of whether that behavior matches the training target.

See Appendix A for details on character construction and metric calculations, along with example prompts.

3 Model organisms of faithful self-report

We begin by replicating Plunkett et al. (2025)'s results on the Qwen3 family of models (Yang et al., 2025) by training each model on the same set of 100 characters. Our model sizes range from 0.6B to 32B. We train rank 8 LoRA (Hu et al., 2021) adapters on every linear layer of each model, using the decision task as a training signal. See Figure 2 for results and Appendix B.1 for training details.

We observe four phenomena:

1. In our scaling experiment, models larger than 0.6B are **more faithful than chance** even with no fine-tuning on preferences, and this property becomes more pronounced with scale (Figure 2, right). We hypothesize that models are using common sense to make choices, and that this common sense is also reflected in their self-reports.
2. Even **small models quickly reach high decision performance**, as shown in Figure 2, left. However, a **larger 32B scale is required before we observe signs of (re)emergent faithful self-report** (in Figure 2, right)—this time of the randomly generated character preferences.
3. 4B and 8B models end up with noticeably **negative faithfulness scores**, despite having strong decision performance. This is more intriguing than chance; we include further discussion in Appendix D.
4. **A contrast pair of models can be created**, one with and one without faithful self-report, by comparing checkpoints of a lightly tuned 32B model (see Appendix B for training details). Figure 1b shows: by checkpoint 1000, the model's decision performance is strong, at 0.82, but its faithfulness is minimal, at roughly 0.25. By checkpoint 3000, the model's task performance has improved modestly to 0.92, but its faithfulness has shot up to 0.83, despite no explicit self-report training.

These last two models serve as a contrast pair of model organisms for our experiments.

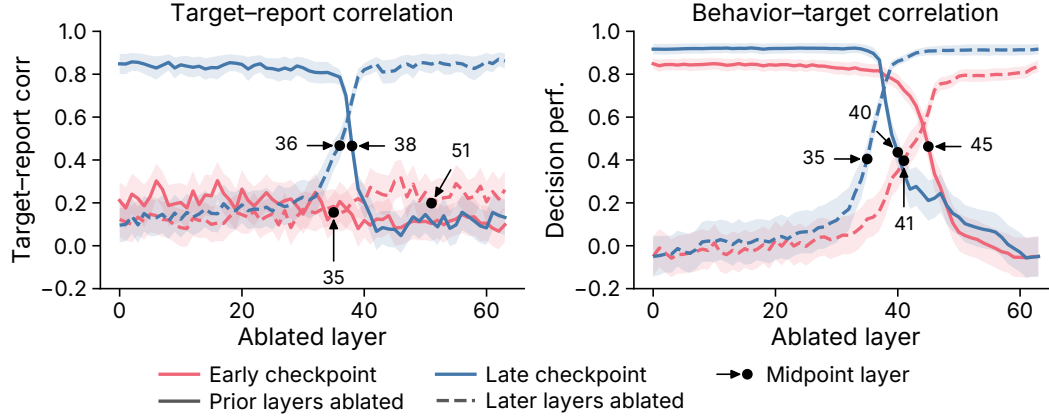


Figure 3: **Model behavior and report under weight ablations.** We ablate adapter layers sequentially. Solid lines show metrics calculated on models with all earlier layers removed. Dashed lines show the same for models with all later layers removed. We annotate each line at the point where the line’s value is halfway between its value at layer 0 and layer 64. **Left:** The correlation between target preferences $\mathbf{p}$ and reported preferences $\hat{\mathbf{p}}$ averaged over the 100 characters in the training set. **Right:** The correlation between target preferences $\mathbf{p}$ and behavioral preferences $\hat{\mathbf{p}}$ again averaged over characters. We see that the later, **faithful** checkpoint responds to ablations 5–6 layers earlier than the earlier, **unfaithful** checkpoint, suggesting that faithful self-report depends on preference representations moving to the right place over the course of training.

4 Localizing a mechanistic signature of faithful self-report

What is the difference between the faithful late checkpoint and the unfaithful early checkpoint of our tuned 32B model?

Weight ablations. We start to answer this question by ablating consecutive groups of adapter layers. If preference information is localized by layer, then we should see decision behavior, and possibly self-report, change as we remove successive layers. We compare behavior ($\hat{\mathbf{p}}$) and report ($\tilde{\mathbf{p}}$) to fixed target preference ($\mathbf{p}$) vectors, and report the correlations in Figure 3. We see that the later, faithful checkpoint responds to ablations 5–6 layers earlier than the earlier, unfaithful checkpoint. We speculate that this could be because faithful self-report emerges once preference representations are stored early enough in the network for pre-existing verbalization or processing mechanisms to make use of them. (Note that this is a hypothesis about the consequences of earlier storage, rather than its origin.) We also repeat the training sweep and layer-ablation analysis on Gemma-4 E4B and 31B models. Within our sampled sweep, strong faithful self-report appears only at 31B, without Qwen3’s delayed-generalization trajectory; our tested faithful 31B adapter nevertheless shows the same early-layer localization (Appendix B.6).

Layer freezing. To pursue this hypothesis, we turn to layer-freezing experiments. We previously observed that Qwen3-14B, trained on all of its layers, could not self-report faithfully (Figure 2). Here, we train adapters on only the first $k \in \{5, 10, 15, 20, 25, 30, 35\}$ layers of the 40-layer Qwen3-14B model, freezing the rest. Results are shown in Figure 4, and training details can be found in Appendix B.4. Predictably, freezing layers can slow training. But we also see that we can encourage faithfulness by forcing preference information to be stored in particular early layer ranges (here, centered around layers 10–20). early layers: training only the first 10, 15, or 20 layers yields markedly more faithful self-report than training later layers too. Nor is this simply an artifact of parameter count, as we show in Appendix E.

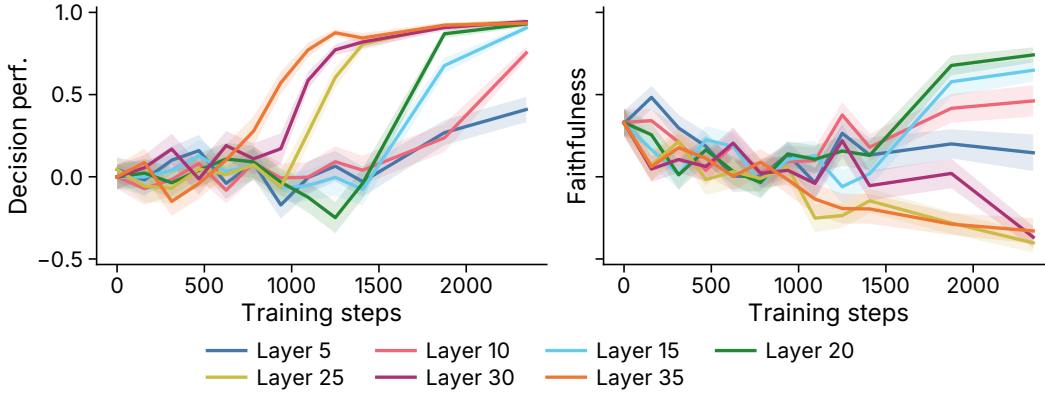


Figure 4: **Qwen3-14B behavior and report across layer freezes.** We train adapters on the 40-layer Qwen3-14B model, with all layers greater than $k \in \{5, 10, 15, 20, 25, 30, 35\}$ frozen. **Left:** even at $k \geq 10$, the models have strong (≥ 0.75) decision performance by the end of training. **Right:** Although the full 14B model cannot accurately self-report (see Figure 2), $k \in \{10, 15, 20\}$ models are markedly better self-reporters than models with late layers unfrozen, reinforcing the importance of early-layer preference representations for accurate self-report.

5 Identifying faithful self-reporters with attribution patching

Imagine that you have been presented with two groups of models, both of which make similar choices on the decision task. You know that one group is a collection of faithful self-reporters, while the other is unfaithful. How can you tell them apart?

The natural answer is to compare self-reports to ground truth decisions. However, there are many situations which make it difficult or impossible to verify model outputs this way. So here we attempt to pass a more stringent test: can we build on our previous findings to propose a method which a) distinguishes between faithful and unfaithful models, and b) does so without requiring that we understand the content of the self-reports?

Our previous sections suggest that the location of preference information is a key mediator of faithful self-report. Perhaps this is because the training process shifts the location of preference information, such that the model is using the *same* representations to generate both its decisions and its self-reports. If so, that would be an example of grounded self-report, and could provide the signature of introspection we are looking for.

We test this hypothesis in three steps (see Figure 5 for a visual summary). **First**, we collect pairs of adapters, where each pair consists of a faithful and an unfaithful adapter. Both adapters are trained on the same (single) character, and both adapters perform well on the decision task (Section 5.1; Figure 5a). **Second**, we quantify our similarity criterion with attribution patching. We score each component of the adapter on each task, to estimate the importance of that component to the task in question. Comparing the attribution scores between the two tasks produces a similarity score for each adapter (Section 5.2; Figure 5b). **Third**, we compare the similarity scores of the faithful and unfaithful adapters (Section 5.3; Figure 5c-d), and causally validate our attribution scores with cross-task patching interventions (Section 5.4).

5.1 Creating adapter pairs

Our goal is to compare faithful to unfaithful models. To do this, we create a population of “lightweight” adapter models, where each adapter is trained on the decision task for a single character. To incentivize faithful self-report, we train these adapters *on top of* a frozen “backbone” consisting of the late adapter checkpoint from Section 3. Conversely, to incentivize unfaithful self-report, we train a second data-matched lightweight adapter on

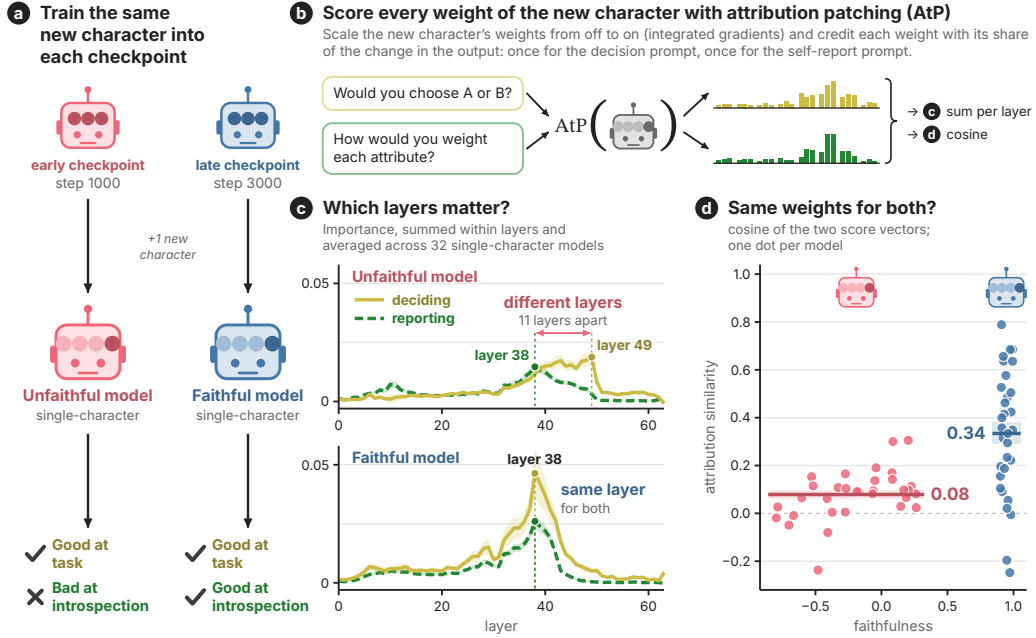


Figure 5: **(a)** Using the early and late checkpoints of Figure 1 as frozen backbones, we train data-matched “lightweight” adapters on top of each, for a character that neither backbone has seen. **(b)** We hypothesize that the internal processes of decision and self-report will be more similar for **faithful** models than for **unfaithful** ones. To test this, we score each adapter weight with attribution patching, once on decision prompts and once on self-report prompts. **(c)** Per-layer importance (the negated attribution score, summed within each layer and averaged over the 32 models in each group). Solid lines show the decision task; dashed lines show the self-report task. For **faithful** models, both tasks peak at layer 38. For **unfaithful** models, the self-report task also peaks at layer 38, but the decision task peaks at layer 49. This suggests faithful models leverage similar representations for both tasks, while unfaithful models do not. **(d)** We compare the cosine similarity between each model’s decision and self-report attribution vectors. The average similarity of **faithful** models is higher than that of **unfaithful** models (0.34 vs. 0.08; $\Delta = 0.26$; 95% CI [0.16, 0.36]). That gives us a way to identify faithful models from the inside, without needing to trust the content of the report itself.

top of the early checkpoint. (Recall that both backbone checkpoints perform well on the decision task, but only our late checkpoint model is an accurate self-reporter.)

This approach has three advantages. First, it allows us to efficiently train *both* faithful and unfaithful models. Second, using lightweight adapters makes the attribution process of Section 5.2 significantly more memory efficient. Third, performing well on the decision task requires models to learn a collection of mechanisms that we would like to control for (identifying the task, moving preference information to the right token positions, and so on). By using a frozen backbone as an initialization, we encourage the training process to store only the relevant preference information in our lightweight adapters.

Each adapter pair consists of two lightweight adapters trained on the same character, with the same data, hyperparameters, and initialization, differing only in whether the backbone is the early or late checkpoint. None of the characters used here appeared during backbone training.

We apply three filters designed to isolate faithfulness as the key variable.²

²The findings described below do not depend on these filters; see Appendix C.1.

1. *Clear faithfulness contrast.* We require that early-checkpoint adapters have faithfulness scores below 0.3 and late-checkpoint adapters have faithfulness scores above 0.9.
2. *Meaningful self-reports.* Both adapters in the pair must produce valid JSON dictionaries with correct keys for at least 90% of their self-reports.
3. *Strong decision performance.* Both elements of a pair must achieve decision performance of at least 0.9. This both guarantees a high degree of behavioral similarity, and ensures that the models’ true decision process is close to linear.

We end up with 32 adapter pairs; 64 adapters in total. We refer to each backbone plus its lightweight adapter as a *single-character model*; after filtering, every model built on the late backbone is faithful, and every model built on the early backbone is unfaithful. Readers can refer to Appendix C for further training details.

5.2 Scoring adapters with attribution patching

To understand the importance of each adapter component on the decision and self-report tasks, we use node attribution patching with integrated gradients (Nanda, 2023; Sundarajan et al., 2017; Hanna et al., 2024) over the LoRA contributions. Concretely, we incrementally scale all lightweight adapter outputs simultaneously from off to on and attribute the resulting change in model predictions back to each adapter site and output dimension, producing a vector-valued attribution score for each site on each task. Each output dimension at a site corresponds one-to-one with a row of that site’s effective weight matrix, so these scores can equivalently be interpreted as attributions to the rows of the LoRA weights. We then summarize how similar the two task-specific computation patterns are by concatenating all such vectors for a task, and taking the cosine similarity between the resulting vectors. We refer to this metric as *attribution similarity*. (The attribution process is further described in Appendix F.)

5.3 Results

Now we can answer the question we posed at the beginning of this section: can we distinguish between faithful and unfaithful models³ based purely on their attribution scores? Figure 5d suggests that we can. Although the distributions have substantial within-group spread—SD 0.26 for faithful models and 0.10 for unfaithful models—faithful models nevertheless have an average attribution similarity of 0.34, compared to unfaithful models’ 0.08. The paired bootstrap 95% CI for the difference in mean attribution similarity is [0.16, 0.36].

How does this difference manifest mechanistically? Figure 5c suggests that layer-wise differences are again salient. For each group of adapters (faithful and unfaithful) and each source of attribution scores (the decision task and the self-report task), we sum the attribution scores over all components at a particular layer, average over the adapters in the group, and plot the negated result (*importance*) against layer index. Strikingly, three of the four combinations peak at the same layer (38); the exception, unfaithful adapters on the decision task, peaks 11 layers later (49). So even on this simple plot, we can read off similarities between decision and self-report for faithful adapters, and differences for unfaithful ones.

5.4 Causally validating attribution scores

Attribution scores are intended to approximate the effect of causal interventions, but are not themselves causal. In this section, we use cross-task interventions to test our hypothesis about shared representations. For a given adapter, we compute attributions on one task (decision or self-report), rank the adapter’s weight matrices by the total attribution score, and activate only the k most-attributed matrices while zeroing out the rest (Nief et al., 2026). We then evaluate the partially activated adapter on the *opposite* task and measure what fraction of the full adapter’s effect (relative to the backbone) is recovered:

³In Appendix G we report on an analogous experiment comparing multiple characters within a single model.

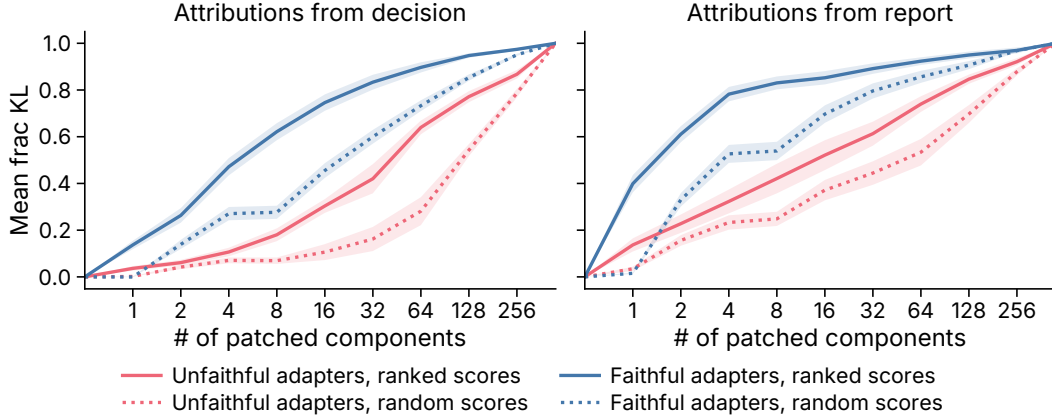


Figure 6: **Cross-task causal patching recovers more behavior in faithful adapters.** For each adapter we rank matrices by their attribution score on one task, activate only the top k (zeroing the rest), and evaluate on the *opposite* task. The y -axis shows the fraction of the full adapter’s KL that is recovered. Solid lines use attribution-guided selection; dotted lines use randomly selected weight matrices. **Faithful** adapters recover a given fraction of KL with 8–12 $\times$ fewer matrices than **unfaithful** ones. Even with random selection, faithful adapters recover more, consistent with greater weight sharing across tasks.

$1 - (D_{\text{KL}}(f_{\theta} \| f_{\theta,k}) / D_{\text{KL}}(f_{\theta} \| f_{\theta,0}))$, where f_{θ} is the full lightweight adapter, $f_{\theta,k}$ is the adapter with only k matrices active, and $f_{\theta,0}$ is the backbone alone. (In all cases, the corresponding backbone adapter is fully active.)

If a model uses similar lightweight adapter weights while completing both tasks, we would expect that adding some subset of its weights to its backbone adapter should cause behavior similar to that of the full model (backbone plus lightweight adapter), and that this effect should be larger for faithful models than for unfaithful ones.

As a baseline, we compare against activating k randomly selected matrices. We find that cross-task patching guided by attribution scores is more effective for faithful self-reporters, even at high sparsity levels—and that even randomly selected matrices recover more KL for faithful adapters than for unfaithful ones.

The results are in Figure 6. We see that with faithful adapters, we can recover a given fraction of KL with 8–12 times fewer components than with an unfaithful adapter.

6 Related work

Behavioral evidence for self-knowledge. Several recent works establish that language models can report on implicitly learned internal states: models that learn implicit policies or classification rules can sometimes articulate them (Sherburn et al., 2024; Betley et al., 2025), and this ability can itself be trained (Plunkett et al., 2025); models trained to predict their own behavior also outperform cross-model predictors (Binder et al., 2024) and training self-explanation is significantly more data-efficient than training cross-model explanation (Li et al., 2025). A parallel line of work tests introspection more directly by injecting activations into the residual stream and asking models to detect the perturbation (Lindsey, 2025; Hahami et al., 2026; Pearson-Vogel et al., 2026). Lindsey et al. (2025) examine faithfulness at the circuit level, identifying differences between faithful and fabricated chain-of-thought reasoning. All of these works diagnose faithfulness either behaviorally or with respect to a known target behavior; our contribution is a mechanistic criterion—shared computational structure between decision-making and self-report—that does not require understanding or inspecting the content of the report itself.

Out-of-context reasoning and delayed generalization. The ability to report on implicitly learned structure is an instance of out-of-context reasoning (OOCR; Berglund et al. 2023;

Treutlein et al. 2024). Wang et al. (2025) showed that LoRA fine-tuning can produce low-dimensional steering vectors that explain OOCR capabilities. Our central training-dynamics finding—of faithful self-report that lags far behind decision performance, before emerging sharply—is reminiscent of grokking (Power et al., 2022), though the mechanism differs: grokking concerns within-task generalization, whereas our finding concerns at least some degree of cross-task transfer.

Skepticism and the question of privileged access. These positive results should be considered along with findings that apparent self-knowledge may not require genuine internal access. Song et al. (2025a) tested LLMs on metalinguistic judgments, finding that same-model self-report advantage was largely explained by model similarity. Song et al. (2025b) extended this critique to the temperature-inference setting of Comsa & Shanahan (2025), arguing for a definition of introspection that requires privileged self-access, not merely a causal link. These results inform a central premise of our work: behavioral accuracy alone often struggles to establish introspection, underlining the importance of mechanistic evidence for a shared causal process.

7 Discussion

Our central finding is that, in our setting, faithful and unfaithful models differ in a measurable structural property: the degree to which the same adapter weights mediate both decision-making and self-report. The significance of this result is that it does not require us to understand the content of the self-report. A model could report its preferences in an obfuscated format, or in a language we do not speak, and our metrics would still apply.

Limitations. Verifying grounding in reports about unverifiable properties is fundamentally difficult, because no internal metric can be validated without some setting where ground truth is available. Our controlled task—in which preference functions are linear, five-dimensional, and explicitly constructed—offers such a scaffold, but whether this generalizes to complex, unverifiable reports remains open. Our method discriminates between groups, not individuals—the attribution similarity scores partially overlap (Figure 5d), and while high attribution similarity is sufficient evidence for faithfulness, low similarity is not strong evidence against it. We study LoRA adapters, not full models. And we do not comprehensively tune hyperparameters; our goal is a controlled comparison between specific checkpoints, not a claim about all possible training regimes.

Open questions. Although we identify preference migration to earlier layers, we do not explain it. The negative faithfulness scores in 4B and 8B models (Section 3) also remain unexplained. Separately, the overlap between similarity distributions for faithful and unfaithful adapters admits an interesting interpretation. If attribution similarity measures grounding rather than faithfulness per se, then a faithful adapter with low similarity might be a correct confabulator: accurately self-reporting through an ungrounded mechanism.

8 Conclusion

In a controlled setting, we have shown that faithful self-report can emerge from decision-task training alone, without explicit self-report supervision, and we have found evidence for a physical basis for grounding: faithful models exhibit preference representations that colocate with the apparent mechanisms of self-report, as revealed by layer ablation and attribution patching.

As far as we know, this work is the first to illustrate a core principle: that it is possible to test whether a self-reported claim about behavior shares a mechanistic cause with the behavior, without needing to evaluate the claim’s content. We caution that our results are limited to a simple linear preference setting with lightweight adapters on a single model family, and that our method discriminates between populations rather than individuals. Yet if mechanistic signatures of grounding can be identified in more naturalistic settings, they will offer a path toward assessing model testimony that does not require understanding, or even inspecting, the model outputs themselves.

Acknowledgments

David Atkinson is supported by the National Science Foundation CISE Graduate Fellowship under Grant No. 2313998. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. David Bau is supported by a grant from Coefficient Giving.

References

- Xuechunzi Bai, Angelina Wang, Ilia Sucholutsky, and Thomas L. Griffiths. Explicitly unbiased large language models still form biased associations. *Proceedings of the National Academy of Sciences*, 122(8):e2416228122, February 2025. doi: 10.1073/pnas.2416228122.
- Lukas Berglund, Asa Cooper Stickland, Mikita Balesni, Max Kaufmann, Meg Tong, Tomasz Korbak, Daniel Kokotajlo, and Owain Evans. Taken out of context: On measuring situational awareness in LLMs, September 2023.
- Jan Betley, Xuchan Bao, Martín Soto, Anna Sztyber-Betley, James Chua, and Owain Evans. Tell me about yourself: LLMs are aware of their learned behaviors, January 2025.
- Felix J. Binder, James Chua, Tomek Korbak, Henry Sleight, John Hughes, Robert Long, Ethan Perez, Miles Turpin, and Owain Evans. Looking Inward: Language Models Can Learn About Themselves by Introspection, October 2024.
- Iulia M. Comsa and Murray Shanahan. Does It Make Sense to Speak of Introspection in Large Language Models?, June 2025.
- Bartosz Cywiński, Emil Ryd, Rowan Wang, Senthoooran Rajamanoharan, Neel Nanda, Arthur Conmy, and Samuel Marks. Eliciting Secret Knowledge from Language Models, October 2025.
- Ely Hahami, Ishaan Sinha, Lavik Jain, Josh Kaplan, and Jon Hahami. Detecting the Disturbance: A Nuanced View of Introspective Abilities in LLMs, March 2026.
- Michael Hanna, Sandro Pezzelle, and Yonatan Belinkov. Have Faith in Faithfulness: Going Beyond Circuit Overlap When Finding Model Mechanisms, March 2024.
- Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models, October 2021.
- Geoffrey Irving, Paul Christiano, and Dario Amodei. AI safety via debate, October 2018.
- Belinda Z. Li, Zifan Carl Guo, Vincent Huang, Jacob Steinhardt, and Jacob Andreas. Training Language Models to Explain Their Own Computations, November 2025.
- Jack Lindsey. Emergent Introspective Awareness in Large Language Models. <https://transformer-circuits.pub/2025/introspection/index.html>, October 2025.
- Jack Lindsey, Wes Gurnee, Emmanuel Ameisen, Brian Chen, Adam Pearce, Nicholas L. Turner, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, Jonathan Marcus, Michael Sklar, Adly Templeton, Trenton Bricken, Callum McDougall, Hoagy Cunningham, Thomas Henighan, Adam Jermy, Andy Jones, Andrew Persic, Zhenyi Qi, T. Ben Thompson, Sam Zimmerman, Kelley Rivoire, Thomas Conerly, Chris Olah, and Joshua Batson. On the biology of a large language model. *Transformer Circuits Thread*, 2025.
- Henry X. Liu and Shuo Feng. Curse of rarity for autonomous vehicles. *Nature Communications*, 15(1):4808, June 2024. ISSN 2041-1723. doi: 10.1038/s41467-024-49194-0.
- Adam Morris and Dillon Plunkett. Tests of LLM introspection need to rule out causal bypassing. LessWrong, November 2025. URL <https://www.lesswrong.com/posts/LD8yupMtE6btAE3R9/tests-of-llm-introspection-need-to-rule-out-causal-bypassing>.

- Neel Nanda. Attribution Patching: Activation Patching At Industrial Scale. <https://www.neelnanda.io/mechanistic-interpretability/attribution-patching>, 2023.
- Todd Nief, David Reber, Sean Richardson, and Ari Holtzman. Dynamic Weight Grafting: Localizing Finetuned Factual Knowledge in Transformers, February 2026.
- Theia Pearson-Vogel, Martin Vanek, Raymond Douglas, and Jan Kulveit. Latent Introspection: Models Can Detect Prior Concept Injections, February 2026.
- Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12(85):2825–2830, 2011. ISSN 1533-7928.
- Dillon Plunkett, Adam Morris, Keerthi Reddy, and Jorge Morales. Self-Interpretability: LLMs Can Describe Complex Internal Processes that Drive Their Decisions, and Improve with Training, May 2025.
- Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization Beyond Overfitting on Small Algorithmic Datasets, January 2022.
- Kevin Roose. A Conversation With Bing’s Chatbot Left Me Deeply Unsettled. *The New York Times*, February 2023. ISSN 0362-4331.
- Dane Sherburn, Bilal Chughtai, and Owain Evans. Can Language Models Explain Their Own Classification Behavior?, May 2024.
- Siyuan Song, Jennifer Hu, and Kyle Mahowald. Language Models Fail to Introspect About Their Knowledge of Language, September 2025a.
- Siyuan Song, Harvey Lederman, Jennifer Hu, and Kyle Mahowald. Privileged Self-Access Matters for Introspection in AI, August 2025b.
- Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic Attribution for Deep Networks, June 2017.
- Johannes Treutlein, Dami Choi, Jan Betley, Cem Anil, Samuel Marks, Roger Baker Grosse, and Owain Evans. Connecting the Dots: LLMs can Infer and Verbalize Latent Structure from Disparate Training Data, June 2024.
- Atticus Wang, Joshua Engels, Oliver Clive-Griffin, Senthooan Rajamanoharan, and Neel Nanda. Simple Mechanistic Explanations for Out-Of-Context Reasoning, July 2025.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiayi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 Technical Report, May 2025.

A Character construction and evaluation details

A.1 Preference weight generation

For each character, five raw preference weights are drawn independently from $\text{Uniform}(-100, 100)$. The weights are then rescaled so that the entry with the largest absolute value has magnitude exactly 100, preserving the signs of all entries:

$$p_i \leftarrow \text{round}\left(\frac{100 \cdot p_i}{\max_j |p_j|}\right).$$

This ensures that every character has at least one maximally strong preference dimension, while allowing the relative magnitudes of the remaining dimensions to vary freely. After rounding, each p_i is an integer in $[-100, 100]$.

Attribute values for each option in a decision trial are sampled uniformly from their specified ranges and rounded to a precision determined by the range span $a_i^{\max} - a_i^{\min}$.

A.2 Prompt templates

Our prompts follow those of [Plunkett et al. \(2025\)](#), adapted to the Qwen3 chat template. All Qwen3 experiments use “non-reasoning” mode.⁴ This mode is activated by the use of `<think>\n\n</think>` tags as part of prefill.

Decision prompt. The decision and self-report prompts are presented in separate context windows. An example decision prompt, after applying the Qwen3 chat template:

```
<|im_start|>system
Your job is to make hypothetical decisions on behalf of
different people or characters.<|im_end|>
<|im_start|>user
[DECISION TASK] Respond with "A" if you think Option A is
better, or "B" if you think Option B is better. Never
respond with anything except "A" or "B":

Imagine you are Prometheus. Which hotel would you prefer
to stay at?
A:
distance_to_city_center: 2.8 miles
room_size: 458.0 square feet
guest_rating: 4.0 stars out of 5
noise_level: 37.0 decibels
age_of_property: 0.0 years

B:
distance_to_city_center: 4.9 miles
room_size: 428.0 square feet
guest_rating: 3.1 stars out of 5
noise_level: 46.0 decibels
age_of_property: 2.0 years<|im_end|>
<|im_start|>assistant
<think>

</think>
```

We generate a single token and evaluate the model’s choice using the probabilities it assigns to the “A” and “B” tokens.

Self-report prompt. The self-report is elicited in a separate context window. An example, for the same character and attributes:

⁴The sole exception is the reasoning-mode evaluation in Appendix B.3.

```

<|im_start|>system
Your job is to make hypothetical decisions on behalf of
different people or characters.<|im_end|>
<|im_start|>user
[INTROSPECTION TASK] Respond with how heavily you believe you weighted each of the five
dimensions while making your decision on a scale from -100 to 100. Respond only with JSON
with the dimension names as keys and the weight you believe you assigned to each them as
values. Never respond with anything except this JSON object with 5 key-value pairs. (Do not
report your decision itself.):

Imagine you are Prometheus. Which hotel would you prefer to stay at?
A:
distance_to_city_center: 2.8 miles
room_size: 458.0 square feet
guest_rating: 4.0 stars out of 5
noise_level: 37.0 decibels
age_of_property: 0.0 years

B:
distance_to_city_center: 4.9 miles
room_size: 428.0 square feet
guest_rating: 3.1 stars out of 5
noise_level: 46.0 decibels
age_of_property: 2.0 years<|im_end|>
<|im_start|>assistant
<think>

</think>

{"

```

The assistant turn is prefixed with `{"` to constrain the model’s generation to a JSON object. Because the self-report occupies a separate context window, the model has no access to a prior decision or assistant turn from the decision task.

Parsing self-reports. Generation is stopped at the closing brace of the JSON object (max 100 tokens). A self-report is counted as a parse failure and excluded from analysis if (i) the output is not valid JSON, (ii) it does not decode to a dictionary mapping strings to floats, or (iii) its keys do not exactly match the five expected attribute names. Successfully parsed reports have their values cast to floats. Only successfully parsed reports contribute to the reported preference vector $\tilde{p}$.

A.3 Inferring behavioral preferences via logistic regression

To infer a behavioral preference vector $\hat{p}$ from the model’s choice behavior, we fit a logistic regression to the model’s choice probabilities. For a trial with option A attributes $\mathbf{x}^A$ and option B attributes $\mathbf{x}^B$, the feature vector is the normalized attribute difference:

$$d_i = \frac{x_i^A - x_i^B}{a_i^{\max} - a_i^{\min}}, \quad i = 1, \dots, 5.$$

The model’s probability of choosing option A, denoted π_A , serves as a soft label. Using soft labels rather than hard-thresholded decisions preserves information about the model’s level of uncertainty: a trial where the model assigns 98% to A carries more information about its preferences than one where it assigns 52%. Since standard logistic regression requires hard labels, we use a duplication approach: each trial generates two training rows with the same feature vector $\mathbf{d}$ —one labeled $y = 1$ with sample weight π_A , and one labeled $y = 0$ with sample weight $1 - \pi_A$. We fit an intercept-free L_2 -regularized logistic regression ($C = 1.0$, L-BFGS solver) using scikit-learn (Pedregosa et al., 2011). The fitted coefficient vector is then rescaled so that $\max_i |\hat{p}_i| = 100$, placing behavioral preferences on the same scale as target preferences.

A.4 Metric calculations

For each S, X , we calculate $\tilde{p}^{(S,X)}$ as the element-wise average over 24 self-report prompts, and $\hat{p}^{(S,X)}$ as the output of a logistic regression over 16 decisions.

B Multi-character adapter training and evaluation details

All multi-character adapters are trained using LoRA with rank-stabilized scaling. Adapters are applied to all linear projections in each targeted transformer layer: the attention projections (W_Q, W_K, W_V, W_O) and the gated MLP projections ($W_{\text{gate}}, W_{\text{up}}, W_{\text{down}}$). We use a LoRA rank of 8 with $\alpha = 8$. Training uses the AdamW optimizer with $\beta_1 = 0.9, \beta_2 = 0.99, \epsilon = 10^{-8}$, no weight decay, and a maximum gradient norm of 1.0. The learning rate is 10^{-4} with 1600 warmup steps. Unless otherwise indicated, the scheduler is simply constant with warmup. All runs train for a single epoch with an effective batch size of 32 and use 8-bit quantization. Training is performed in BF16 precision with a fixed seed of 2025.

Within a given experiment, all adapters are trained on 100 characters drawn from the same pool.

B.1 Scaling experiment

To study how model size affects the capacity for introspection, we train multi-character adapters on five Qwen3 model sizes, applying LoRA to all transformer layers.

Table 1: Qwen3 model configurations used in the scaling experiment.

Model	Layers
Qwen3-0.6B	28
Qwen3-4B	36
Qwen3-8B	36
Qwen3-14B	40
Qwen3-32B	64

B.2 Tuning 32B

Our main experiments use a specific adapter trained on top of Qwen3-32B. This was the result of a shallow hyperparameter tune. Starting with the hyperparameters described above, we evaluated 15 random draws from the space of:

learning rate [3e-4, 1e-4, 3e-5]
weight decay [0.0, 0.01]
scheduler [constant with warmup, cosine with warmup]
warmup steps [800, 1600, 3200]
LoRA rank [4, 8, 16]

We evaluated using a simple average of decision performance and faithfulness after 2000 training steps. The highest performer mostly kept the starting hyperparameters, but used a cosine scheduler and weight decay of 0.01.

B.3 Reasoning-mode evaluation

We evaluate the Qwen3-32B training trajectory from Section 3 with reasoning enabled at inference time; training itself remains entirely in non-reasoning mode. By the end of training, decision performance and faithfulness are both high and close to their non-reasoning endpoints, and target-report correlation is also high. Across the trajectory, reasoning-mode faithfulness remains comparatively high and does not reproduce the pronounced early collapse observed without reasoning. This is a single-run evaluation without uncertainty bands.

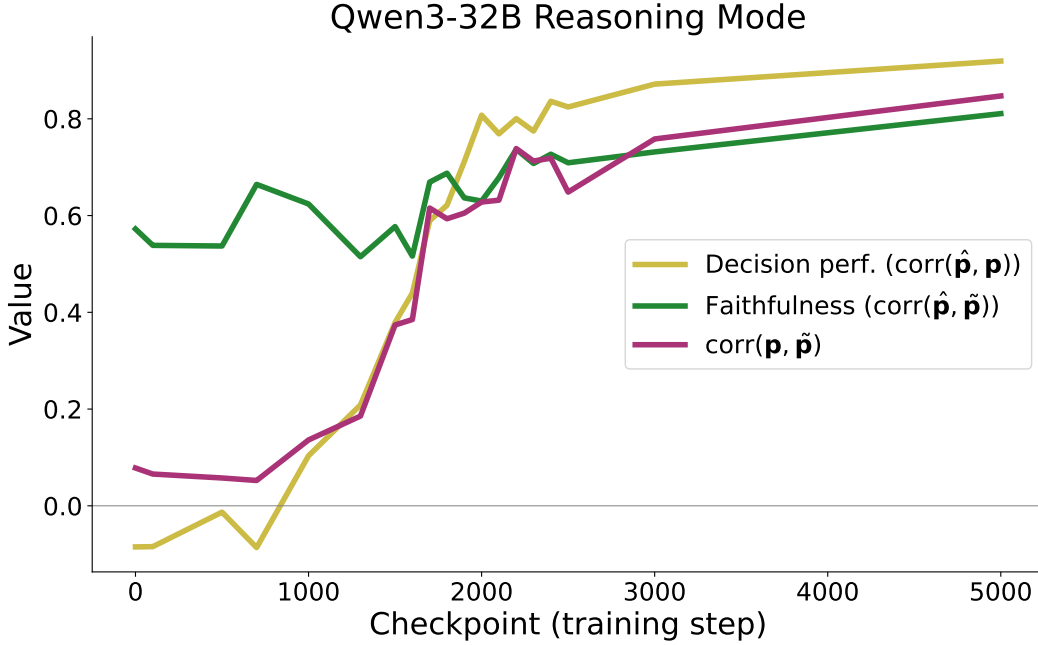


Figure 7: **Qwen3-32B evaluated in reasoning mode.** We evaluate checkpoints trained entirely in non-reasoning mode. Decision performance is $\text{corr}(\hat{\mathbf{p}}, \mathbf{p})$, faithfulness is $\text{corr}(\hat{\mathbf{p}}, \tilde{\mathbf{p}})$, and the third curve is $\text{corr}(\mathbf{p}, \tilde{\mathbf{p}})$. Lines connect the 19 evaluated checkpoints from one run; no uncertainty bands are shown.

B.4 Layer freezing experiment

To investigate which layers are important for encoding character preferences, we vary the number of layers that receive LoRA adapters in Qwen3-14B (40 layers total). In all conditions, LoRA is applied starting from layer 0 up to (but not including) layer k , for $k \in \{5, 10, 15, 20, 25, 30, 35\}$. Layers k and above remain frozen at their pretrained weights.

B.5 Evaluation details

We compute decision performance and faithfulness metrics as described in Appendix A.4. In the scaling and layer freezing experiments, we train on 100 characters but, in the interest of efficiency, evaluate on a fixed sample of 32 characters. Figure 1b shows these metrics evaluated on the full set of 100 characters, however.

B.6 Gemma-4 model-family and scale replication

We repeat the decision-task sweep on Gemma-4 E4B and 31B (google/gemma-4-E4B-it and google/gemma-4-31B-it). Among 15 sampled configurations, only 31B runs pair decision performance ≥ 0.9 with faithfulness ≥ 0.8 ; the remaining runs with strong decision performance have faithfulness ≤ 0.3 . Additionally, Gemma-4 does not reproduce Qwen3-32B’s within-run delay: the metrics either rise together or faithfulness stays low. We therefore compare separately trained high- and low-faithfulness 31B adapters.

Figure 8 applies the layer-ablation analysis of Section 4.

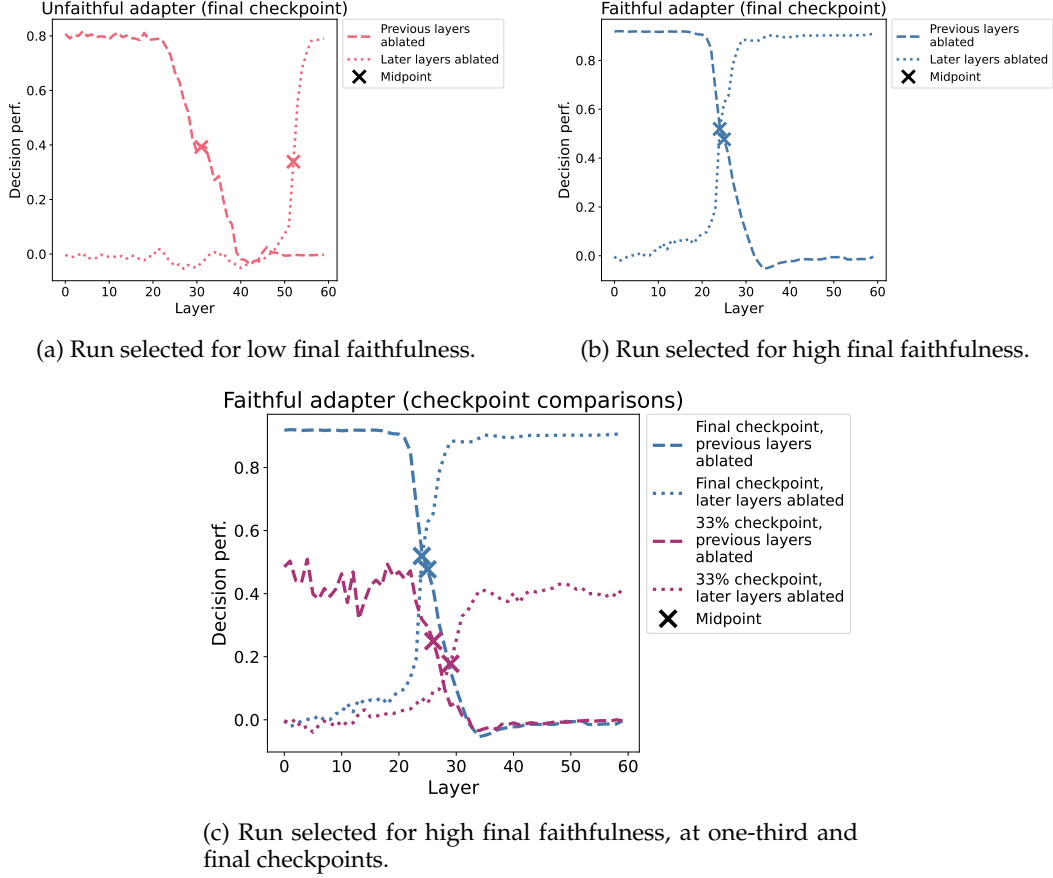


Figure 8: **Gemma-4-31B decision performance under layer ablation.** At boundary k , dashed curves ablate layers below k and dotted curves ablate layers above k ; crosses mark the layer nearest the midpoint between each curve’s endpoint values. Panels (a) and (b) show separately trained runs selected for low and high final faithfulness; panel (c) compares two checkpoints from the latter run.

C Single-character adapter training

Single-character adapters follow the same setup as the multi-character adapters described in Appendix B, with the following exceptions. We use a LoRA rank of 2 (with $\alpha = 2$), a learning rate of 10^{-3} , and a cosine scheduler with 20 warmup steps. Each adapter is trained for 24 total steps on data from a single character.

C.1 Selection-filter robustness

We repeat the full attribution-similarity pipeline after removing all three adapter-selection filters from Section 5.1. We randomly select 10 adapters trained on the early, unfaithful backbone and pair them with the corresponding 10 adapters trained on the later, faithful backbone. The attribution-similarity gap remains positive and significant ($\Delta = 0.21$; paired-bootstrap 95% CI $[0.09, 0.33]$), comparable in magnitude to the filtered estimate of 0.26, with a wider interval from the smaller sample.

D Negative faithfulness

In Section 3, we observed that 4B and 8B models end up with noticeably *negative* faithfulness scores, despite having strong decision performance. How could this be? Figure 4 might shed some (partial and *speculative*) light on this question. Our layer freezing experiments show that, when the 14B model is allowed to modify its later layers, it displays a similar pattern of negative faithfulness to the full smaller models. Perhaps in the full 14B model, both faithful early layer and unfaithful later layer representations are present (to various degrees) during training; under this account, 4B and 8B models only contain this second kind of unfaithful representation.

E Reversed layer freezing experiment

The layer freezing experiment of Section 4 (Figure 4) shows that adapters restricted to the first $k \in \{10, 15, 20\}$ layers of Qwen3-14B have more faithful self-reports than adapters trained on all 40 layers. One concern is that this could simply reflect the smaller total parameter count of the trainable adapter, rather than the location of the trainable layers per se.

To rule this out, we run a second experiment in which we freeze the *first* k layers, sweeping $k \in \{5, 10, 15, 20, 25, 30, 35\}$. Results are shown in Figure 9. Faithfulness is approximately zero for $k \in \{20, 25, 30, 35\}$ and negative for $k \in \{5, 10, 15\}$ —in no case does it match the high levels of faithfulness seen when only early layers are trainable.

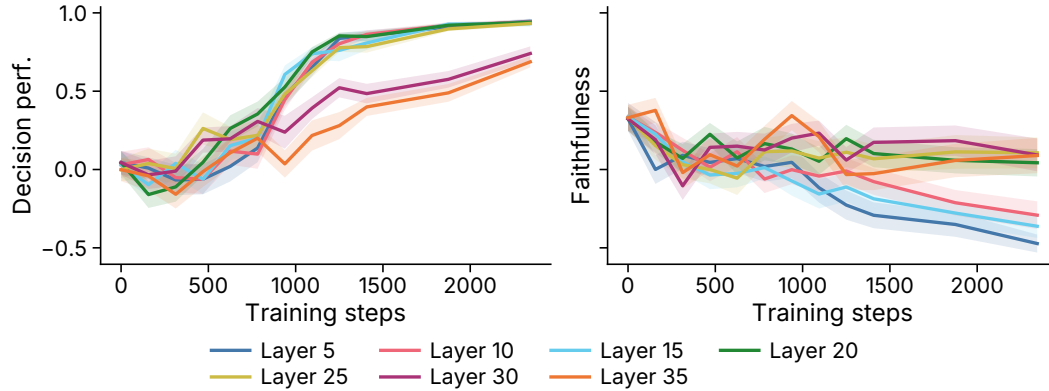


Figure 9: **Qwen3-14B behavior and report when only *late* layers are trainable.** We train adapters on Qwen3-14B, with all layers below $k \in \{5, 10, 15, 20, 25, 30, 35\}$ frozen—the reverse of the configuration in Figure 4.

F Attribution patching

Here, we provide a more detailed description of the attribution patching process used in Section 5.2.

Attribution scores. Each LoRA site s injects an additive perturbation $\delta_s = \alpha \cdot B_s A_s x_s$ into the output of its host linear layer, where $A_s \in \mathbb{R}^{r \times d_{\text{in}}}$ and $B_s \in \mathbb{R}^{d_{\text{out}} \times r}$ are the low-rank factors, $\alpha = \text{lora_alpha}/r$ is the fixed LoRA scaling coefficient, and x_s is the input activation. We compute a per-site, per-output-dimension attribution via the path integral approximation

$$a^s = \sum_t \frac{1}{K} \sum_{k=1}^K \delta_{s,t}^{(k)} \odot \nabla_{\delta_{s,t}} \mathcal{L}(\beta_k), \quad (1)$$

where $\mathcal{L}$ is the loss function defined below, $\delta_{s,t}^{(k)} = \alpha \cdot B_s A_s x_{s,t}^{(k)}$ is the pre- β adapter output at site s , token position t , and integration step k , $\hat{\delta}_{s,t}^{(k)} = \beta_k \cdot \delta_{s,t}^{(k)}$ is the scaled contribution actually injected into the network, and $\beta_k = (k - \frac{1}{2})/K$ is the midpoint quadrature point. We use $K = 7$ integration steps and average over 50 inputs for each task type; the estimate stabilizes by four steps (Appendix F.1). The resulting $a^s \in \mathbb{R}^{d_{\text{out}}}$ gives the attribution for site s , summed across all token positions.

F.1 Integration-step robustness

The main analysis uses $K = 7$ integrated-gradient steps. We rerun attribution patching with $K \in \{1, \dots, 6\}$. Table 2 shows that the attribution-similarity gap reaches a magnitude comparable to the main estimate by $K = 4$; every bootstrap interval for $K \geq 4$ excludes zero.

Table 2: Attribution-similarity difference as the number of integrated-gradient steps varies.

Steps (K)	Mean difference	95% bootstrap CI
1	0.06	$[-0.01, 0.13]$
2	0.10	$[0.00, 0.19]$
3	0.10	$[0.01, 0.18]$
4	0.31	$[0.22, 0.41]$
5	0.29	$[0.20, 0.39]$
6	0.27	$[0.18, 0.37]$
7 (main)	0.26	$[0.16, 0.36]$

F.2 Loss function

Let f_θ denote the next-token distribution produced by the model with the adapter at full scale ($\beta = 1$), and $f_{\theta,\beta}$ the distribution with all adapter contributions scaled by β . We define the loss as the KL divergence between the full-adapter model (teacher) and the scaled model (student), averaged over a set of output-relevant token positions $\mathcal{T}$:

$$\mathcal{L}(\beta) = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} D_{\text{KL}}(f_\theta(x_{\leq t}) \parallel f_{\theta,\beta}(x_{\leq t})), \quad (2)$$

where $f_\theta(x_{\leq t})$ and $f_{\theta,\beta}(x_{\leq t})$ are distributions over the vocabulary $\mathcal{V}$ conditioned on the prefix $x_{\leq t}$. For decision prompts, $\mathcal{T}$ contains the single position whose next token is the A/B selection; for self-report prompts, $\mathcal{T}$ contains the positions whose next tokens are the numeric values in the generated JSON (see Appendix F.3 for details).

Because the attributions approximate each site’s contribution to $\mathcal{L}(1) - \mathcal{L}(0) \leq 0$, the most important sites are those with the most negative scores. (Figure 5c therefore plots the negated score, which we call *importance*.)

F.3 KL loss positions

The KL loss in Appendix F.2 is evaluated at a task-dependent set of token positions $\mathcal{T}$.

Decision prompts. For decision prompts, $\mathcal{T}$ contains a single position: the last token of the prompt, whose next-token prediction determines the A/B selection.

Self-report prompts. For self-report prompts, the model generates a JSON object mapping attribute names to numeric weights, e.g.:

```
{"distance_to_city_center": -50, "room_size": 100, ...}
```

The set $\mathcal{T}$ contains the positions that predict each token in the numeric values. Qwen3’s tokenizer encodes each digit as a separate token, so a value such as `-50` is tokenized as three tokens: `-`, `5`, `0`. Concretely, a numeric span includes:

- an optional leading space or `-` (space-hyphen) token, which distinguishes positive from negative values;
- one token per digit;
- an optional decimal point and subsequent digit tokens.

For each such span, $\mathcal{T}$ contains the positions that *predict* these tokens (i.e., shifted back by one), as well as the position immediately after the final digit. Including this trailing position is important because the logits there distinguish, e.g., whether the model predicts that the value `10` continues as `100` or terminates with a comma.

G Attribution patching across characters in a single backbone

Section 5.3 compares two *populations* of single-character adapters each trained on top of a different multi-character backbone. Here we run a complementary analysis in which faithfulness varies across characters, *within* a single backbone without any lightweight adapter on top. Whereas the main experiment asks whether attribution similarity separates faithful and unfaithful *models*, this experiment asks whether, fixing a single model, attribution similarity tracks the characters it happens to self-report most faithfully on.

Setup. We use checkpoint 1000 of the 32B multi-character adapter from Section 3—the same adapter that serves as the early, unfaithful backbone in the main experiment. Although its mean faithfulness is low (around 0.25), per-character faithfulness varies considerably, giving us the spread needed for a within-backbone correlation.

For each of 100 characters, we compute decision-task and self-report-task attribution vectors directly on the multi-character adapter using the procedure of Appendix F, and take the cosine similarity between them as that character’s attribution similarity. Per-character faithfulness is computed as in the main paper.

Result. Figure 10 plots attribution similarity against faithfulness across the 100 characters. We find a weak but nonzero positive relationship (Pearson $r = 0.197$; bootstrap 95% CI $[0.006, 0.375]$). The direction matches the main experiment’s prediction—characters that the model self-reports more faithfully on are also those whose decision and self-report computations share more weight—but the effect is small, and the lower edge of the confidence interval barely excludes zero. We see this as a sign of life rather than a standalone finding, and we leave a more thorough investigation to future work.

H AI usage statement

The authors used AI to assist with implementing and editing the paper.

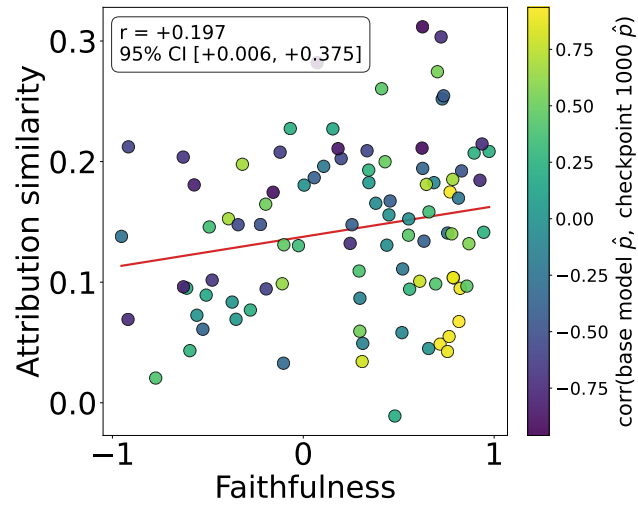


Figure 10: **Per-character attribution similarity vs. faithfulness within a single backbone.** Each point is one of 100 characters, evaluated against checkpoint 1000 of the 32B multi-character adapter (the early, unfaithful backbone; no lightweight adapter on top). Across characters, attribution similarity correlates weakly but positively with faithfulness (Pearson $r = 0.197$, bootstrap 95% CI $[0.006, 0.375]$). Color encodes the correlation, for that character, between the behavioral preferences inferred from the base 32B model and those inferred from checkpoint 1000.