STEERING VECTORS CAN CONTROL WHAT MODELS LEARN IN SUPERVISED FINE-TUNING: AN ANALYSIS OF BEHAVIOR OFFLOADING

Anonymous authors
Paper under double-blind review

ABSTRACT

Fine-tuning and activation steering provide two distinct ways for a model to express a behavior. Fine-tuning writes behavior into the model's parameters, while steering vectors can supply behavior through the residual stream without changing the parameters. We study what happens when these two mechanisms overlap during training. If a behavior required by the training targets is already supplied through the activations, how much of it is still learned into the weights? We call the reduction in learned behavior under this intervention *behavior offloading*. Across eight behaviors spanning format markers, code constructs, style, toxicity, and reasoning, and across models from three families up to 14B parameters, we find that behavior offloading extends far beyond the coarse persona traits it was first shown on and, when successful, reduces the behavior further than a random direction of the same norm does while largely preserving task performance. We then characterize when this control remains effective and where it breaks down. A direction that steers only over a narrow magnitude range can fail to offload even when a more stable construction of the same behavior succeeds. Among successful offloads, the onset occurs within a relatively narrow raw-magnitude range within a model, while the usable range before coherence breaks varies substantially across behaviors. Inference-time steering strength is also associated with whether a direction supports clean offloading or suppresses the behavior only after task performance degrades. Finally, steering does not create a behavior that is absent from the training targets in any setup we test. Together, these results show that activation steering can control which supervised behaviors are written into the weights, but does not replace supervision [1].

1 INTRODUCTION

Fine-tuning and activation steering provide two distinct routes by which a behavior can appear in a model's output. Fine-tuning changes the parameters so that behaviors expressed in the training targets can be reproduced from the model's weights. On the other hand, steering vectors modify the residual stream and can shift behavior while leaving those weights fixed (Turner et al., 2023; Rimsky et al., 2024; Li et al., 2023; Zou et al., 2023). These mechanisms have largely been studied in separate regimes, with activation steering used to change what a frozen model expresses. Activation steering, however, does not inherently require the weights to remain frozen. A steering vector acts on the residual stream, which is also part of the forward computation during fine-tuning. If that vector is present during training, a behavior required by the targets can be supported by the activations at the same time that the weights are being updated to reproduce it. Therefore, this thinking naturally raises a question:

If a behavior is already supplied through the activations during fine-tuning, how much of it will still be learned into the weights?

[1] Sample code is available at `https://anonymous.4open.science/r/Behavior-Offloading-FB6F/README.md`.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

The question matters because supervised targets rarely specify a single behavior in isolation. A teacher's solutions teach the task together with their answer format (Hinton et al., 2015). A chat corpus teaches semantic content together with persona and conversational style (Ouyang et al., 2022). Fine-tuning therefore has to absorb several properties from the same target sequence, even when only some of them are of interest. Persona Vectors (Chen et al., 2025) provide a first piece of evidence that injecting a steering vector during fine-tuning can control which of these co-occurring behaviors are learned. They show that when a persona trait is present in the training data, injecting its direction during fine-tuning reduces how much of that trait the resulting model acquires. This suggests that a behavior supplied through the activations during training need not be learned into the parameters to the same extent. However, this effect has so far been demonstrated only for coarse persona traits such as evil or sycophancy.

Whether activation steering is a general mechanism for controlling what fine-tuning writes into the weights depends on how broadly it applies, when it works, and where it fails. First, it is unclear how far this control extends beyond coarse persona traits. Fine-tuning data contains much finer and qualitatively different behaviors, ranging from formatting conventions and code constructs to diffuse response properties and behaviors inside reasoning. Second, even if behavior offloading extends broadly, we still need to know how strongly the steering vector should be injected. It is unclear when offloading begins as the magnitude increases and how wide the usable range is before the intervention disrupts generation. Finally, the boundary of the effect relative to the supervision is open. Can it only reduce how much of a behavior already present in the training targets is learned, or can it also make the model acquire a behavior that the targets never contain?

In this paper, we study these questions directly and call the effect *behavior offloading*, where supplying a behavior through the residual stream during fine-tuning reduces how much of that behavior is learned into the weights and remains in the clean model after the direction is removed. For each behavior, we construct a steering direction from activation contrasts and first verify on a frozen model that adding the direction at inference moves the behavior. We then inject the verified direction into the residual stream throughout supervised fine-tuning and remove it entirely before evaluation. The clean model therefore reveals what was learned into the weights rather than what is supplied by steering at inference. We consider a behavior offloaded when the steering direction leaves less of the behavior in the clean model than the matched random-direction control, with the effect strengthening as the injection grows. We apply this framework systematically to eight behaviors spanning format markers, code constructs, style, toxicity, and reasoning, across three different model families up to 14B parameters. We find that behavior offloading extends well beyond persona traits but is not universal. The pattern of successes and failures reveals systematic regularities across behaviors and models, allowing us to characterize when steering can alter what fine-tuning stores in the weights and where this control breaks down. Overall, we summarize our contributions as follows:

- We establish behavior offloading as a broad phenomenon beyond coarse persona traits. Across behaviors spanning format markers, code constructs, style, toxicity, and reasoning, and across several models from three families, injecting a behavior direction during fine-tuning reduces how much of that behavior remains in the clean model, while largely preserving task performance.

- We characterize two failure modes of behavior offloading. A behavior present in the training targets can fail to offload when its steering direction is effective only over a narrow magnitude range, while a more stable construction of the same behavior succeeds. A behavior absent from the targets is not created by steering in any setting we test, consistent with the lack of a positive training signal for the missing behavior.

- We further provide analysis of the behavior offloading. For example, we characterize the magnitude range over which behavior offloading remains effective. Within a model, offload onset is relatively concentrated in raw magnitude. Controlled assertion experiments show that stronger inference-time steering is associated with near-complete suppression at training magnitudes that preserve task accuracy.

2 Related Work

Steering vectors. Steering vectors are directions in the residual stream of a language model that shift high-level behavior when added or mixed into the hidden state, without changing any parame-

ters (Turner et al., 2023; Li et al., 2023; Rimsky et al., 2024; Zou et al., 2023). A broad observation in this literature is that many behaviors such as sentiment, truthfulness, refusal, and task identity lie along approximately linear directions, so shifting the hidden state along a direction shifts the corresponding behavior (Turner et al., 2023; Zou et al., 2023). The standard construction takes the difference between the mean hidden states of two response sets that differ in the target behavior (Rimsky et al., 2024; Zou et al., 2023). Related formulations include function vectors (Todd et al., 2024) and weight-space task arithmetic (Ilharco et al., 2023), and a recent line refines their reliability, scaling, and geometry (Braun et al., 2025; Wang et al., 2025; Postmus & Abreu, 2024; Kang et al., 2026; Gan et al., 2025; Venhoff et al., 2025; Nguyen et al., 2025; You et al., 2026). The vector is applied with additive steering (Turner et al., 2023; Rimsky et al., 2024) or spherical interpolation (Shoemake, 1985; Goddard et al., 2024; You et al., 2026), and in all of this work it is applied at inference on a frozen model. The closest work to ours is Persona Vectors (Chen et al., 2025), which adds a steering vector during fine-tuning to keep a model from learning a persona trait such as being evil or sycophantic. They study one coarse trait at a time, while we ask how far this reaches across many behaviors and models and what it cannot reach.

Post-training of language models. Post-training starts from supervised fine-tuning on reference data (Ouyang et al., 2022; Wang et al., 2023; Taori et al., 2023), often with parameter-efficient adapters such as LoRA (Hu et al., 2022), and it is the main tool used to elicit chain-of-thought reasoning before reinforcement learning (Wei et al., 2022; Cobbe et al., 2021; Zelikman et al., 2022; Dong et al., 2023). Knowledge distillation trains a student to imitate a teacher's outputs (Hinton et al., 2015; Gu et al., 2024) and is a special case of fine-tuning on produced targets. A known limitation of these teacher-forced objectives is the off-policy gap between the training targets and the model's own distribution (Ranzato et al., 2016). None of this work uses a steering vector during training. Therefore, in this paper, we inject the steering vector into this stage and study what the trained model keeps and what it drops.

3 Framework

We study behavior offloading by separating the behavior supplied through the residual stream during fine-tuning from the behavior retained by the trained model after the intervention is removed. We first formalize this comparison, and then describe the four-stage procedure in Figure 1 for constructing and validating behavior directions.

3.1 Problem Setup

Our primary setting is supervised fine-tuning on a dataset $\mathcal{D} = \{(x_i, y_i)\}$ whose targets exhibit a behavior b, such as a calculation annotation or an assertion statement. Let M denote the pretrained model and let $B_b(\cdot)$ denote a held-out metric for b, oriented so that a larger value indicates stronger expression of the behavior. Plain supervised fine-tuning produces a baseline model

$$M_0 = \mathrm{SFT}(M, \mathcal{D}). \tag{1}$$

Given a unit steering direction $\hat{v}^\ell$ for behavior b at layer ℓ, training from the same model while injecting the direction at magnitude m produces $M_v(m)$, which we refer to as *steering-conditioned SFT*. Replacing $\hat{v}^\ell$ with a matched random unit direction $\hat{\epsilon}^\ell$ at the same layer and magnitude produces the control model $M_\epsilon(m)$. These interventions are applied only during fine-tuning, and all resulting models are evaluated with no injection.

We define behavior offloading through this comparison. At magnitude m, behavior b is offloaded when less of it remains in the model trained with its steering direction than in the model trained with the matched random direction:

$$B_b(M_v(m)) < B_b(M_\epsilon(m))\,. \tag{2}$$

The plain model M_0 establishes how much of the behavior is learned under standard fine-tuning, while $M_\epsilon(m)$ controls for the effect of injecting a vector of the same magnitude. We report task performance separately to track whether reducing b preserves the underlying task.

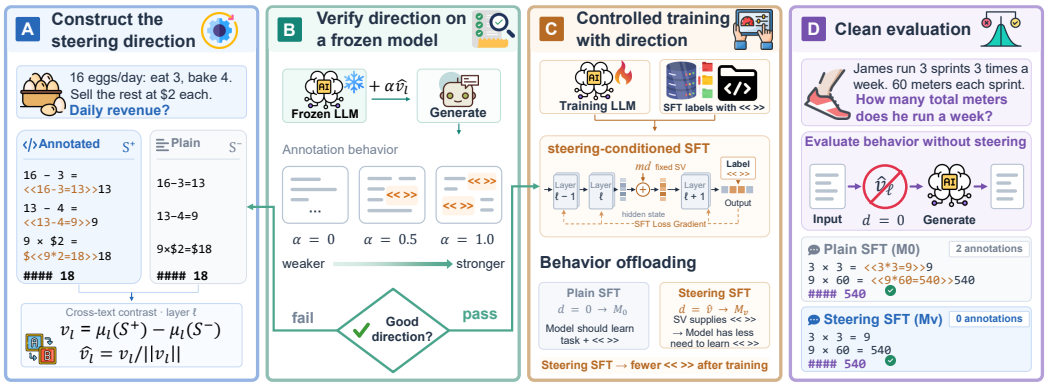


Figure 1: Overview of the behavior offloading procedure, illustrated with four stages using calculation annotations (<<···>>) as a running example.

3.2 Behavior-Offloading Procedure

With the problem defined, we next describe how we construct the intervention and obtain the trained models being compared in detail. The procedure consists of four stages, shown in Figure 1.

Constructing the steering direction Steering vectors can be constructed from internal activations in several ways. Let $h_t^\ell(x) \in \mathbb{R}^d$ denote the residual-stream activation at layer ℓ and token position t in sequence x. As one example, given behavior-present responses S^+ and behavior-absent responses S^-, we can construct a candidate direction from the difference between their mean activations. Using the final response token as the read position gives:

$$v^\ell_{\text{cross}} = \frac{1}{|S^+|} \sum_{x \in S^+} h^\ell_{\text{last}}(x) - \frac{1}{|S^-|} \sum_{x \in S^-} h^\ell_{\text{last}}(x). \tag{3}$$

For calculation annotations, for example, S^+ can contain annotated steps such as <<16-3=13>>, while S^- contains the corresponding plain expression 16-3=13. More generally, we consider multiple ways of constructing a candidate steering direction for each behavior. These choices include which token activations to use, such as the final-token activation or the mean activation over the response. We also consider different ways of extracting a direction from these activations, including mean differences, PCA, and linear probes (Rimsky et al., 2024; Zou et al., 2023). We also consider a single-response construction (Gan et al., 2025). Let $\mathcal{I}_b(x)$ denote the token positions associated with behavior b in sequence x. We contrast their mean activation with that of the remaining positions:

$$v^\ell_{\text{intra}} = \mathbb{E}_x \left[\underset{t \in \mathcal{I}_b(x)}{\text{mean}} h^\ell_t(x) - \underset{t \notin \mathcal{I}_b(x)}{\text{mean}} h^\ell_t(x) \right]. \tag{4}$$

After construction, each candidate direction is unit-normalized before validation as $\hat{v}^\ell = v^\ell / \|v^\ell\|_2$.

Verifying the direction on a frozen model After constructing the steering vector, we verify that the candidate direction controls the target behavior before using it for fine-tuning. We inject $\alpha \hat{v}^\ell$ into the frozen model during generation and sweep the injection strength α. A candidate passes the gate when changing the injection strength α produces a clear increase in B_b over a range of the validation sweep, while a magnitude-matched random direction does not show the same response. Candidates that fail the gate are reconstructed by changing the construction, layer, or read position. These choices are made on a validation set and fixed before fine-tuning and evaluation on a disjoint held-out set. Only directions that pass the gate are used for training, separating failures of behavior offloading from failures of direction construction.

Controlled training with the direction We inject the verified direction throughout supervised fine-tuning. At each training step, we add $m\hat{v}^\ell$ to the residual stream at layer ℓ for every response position on which the supervised loss is computed:

$$\tilde{h}^\ell_t = h^\ell_t + m\hat{v}^\ell, \qquad t \in \text{response}. \tag{5}$$

Table 1: Testbed. Behaviors span three groups, are built with the cross-text or intra-sequence (Eq. 4) construction, and are measured against a matched-noise control on every model and dataset. Model families are Qwen, Llama, and Gemma.

Behavior	Group	Construction	Dataset / Metric
Final-answer marker	Marker	Intra-sequence	GSM8K / marker rate
Calc annotation	Marker	Cross-text	GSM8K / annotation rate
Assertion	Marker	Intra-sequence	MBPP / assert rate
Exception handling	Marker	Intra-sequence	MBPP / try rate
Toxicity	Style	Cross-text	toxicity corpus / toxicity score
Sentiment	Style	Intra-sequence/Cross-text	review corpus / sentiment
Hesitation	Reasoning	Intra-sequence	GSM8K / hesitation rate
Calculation correctness	Reasoning	Cross-text	GSM8K / step-arithmetic correctness

Here m is the injection magnitude. The training targets and loss are otherwise unchanged, and a low-rank adapter (Hu et al., 2022) is optimized through the modified forward pass. Because the injected direction contributes a behavior-related signal during the forward pass used to compute the supervised loss, fitting that behavior need not rely entirely on parameter updates and the behavior offloading may happen. The matched-random control follows the same procedure with the noise $\hat{\epsilon}^{\ell}$.

Clean evaluation To measure what remains in the trained model after steering-conditioned SFT, we remove the intervention entirely at evaluation time. Each model is evaluated with its standard forward pass, so the measured behavior reflects the model after training without any steering supplied at inference. We compare the steering-trained model $M_v(m)$ against both plain fine-tuning M_0 and the magnitude-matched random-direction control $M_\epsilon(m)$. The plain model establishes how much of the behavior is learned under standard supervised fine-tuning, while the random-direction control separates direction-specific offloading from the effect of injecting a vector of the same magnitude. To characterize how offloading changes with the training-time magnitude, we evaluate $M_v(m)$ and $M_\epsilon(m)$ across a sweep of m. We report task performance alongside the behavior metric to determine whether the behavior can be reduced without degrading the underlying task.

Behaviors and models We evaluate behavior offloading on eight behaviors that differ substantially in how they are expressed in model outputs. We organize them into three groups. The first consists of discrete format and syntax markers, including answer formatting and code constructs. The second consists of broader response properties such as toxicity and sentiment. The third consists of behaviors expressed in the reasoning process, including hesitation and calculation correctness. This breadth allows us to test whether behavior offloading extends beyond any single type of behavior or domain. Table 1 reports the steering construction, dataset, and evaluation metric used for each behavior. The experiments draw on GSM8K (Cobbe et al., 2021), StrategyQA (Geva et al., 2021), MBPP (Austin et al., 2021), RealToxicityPrompts (Gehman et al., 2020). To test the breadth of behavior offloading across model families, we evaluate models from Qwen2.5 (Yang et al., 2024), Llama-3.2 (Grattafiori et al., 2024), and Gemma-3 (Team et al., 2025).

4 Where Steering Vectors Offload Successfully

We first ask how broadly behavior offloading extends beyond the coarse persona traits studied in prior work. In particular, we test whether the same training-time intervention can offload behaviors that differ substantially in how they are represented in model outputs, from localized format and syntax patterns to response-level properties and behaviors expressed in reasoning. Across these settings, we find that offloading is not limited to any one form of behavior.

Format and syntax markers We begin with behaviors that appear as explicit patterns in the training targets. On GSM8K, we study the final-answer marker, such as `#### 72`, and calculation annotations, such as `<<16-3=13>>`, which mark intermediate arithmetic steps. On MBPP, we study Python `assert` statements and try/except wrappers added to generated programs. These behaviors are present in the supervised targets, giving standard fine-tuning a direct signal to reproduce them. We leave the training data and labels unchanged and instead ask whether supplying the be-

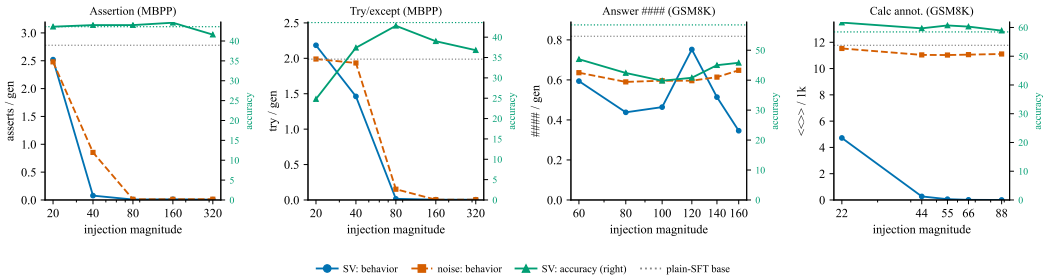


Figure 2: Training-time dose response for format and syntax markers on Qwen2.5-1.5B. The left axis reports the model behavior under steering-conditioned SFT and matched noise, with plain fine-tuning shown by the dotted line. The right axis reports task accuracy.

havior through the residual stream during fine-tuning reduces how much of it remains in the clean model after the intervention is removed.

Figure 2 shows a consistent dose response across all four behaviors on Qwen2.5-1.5B. As the training-time injection increases, the corresponding marker becomes less frequent in the clean model and falls below the matched-noise control over the offloading range. Task performance is largely preserved across these interventions, showing that the model can learn the underlying task without reproducing the same formatting or code convention to the same extent. The main exception is the final-answer marker, where stronger offloading is accompanied by a larger drop in GSM8K accuracy. We analyze its narrow usable magnitude window in §6. We further test robustness with three training seeds and five sampled matched random directions. We report the full robustness results in Appendix A.2. Most successful offloads remain well below random directions except for try/except. Although the steering condition remains far below the mean of the random direction, one sampled random direction suppresses the try/except wrapper. To understand this behavior, we further analyze how random directions offload for fixed templates in Appendix A.3.

Broader response properties The marker experiments leave open whether behavior offloading depends on the target behavior of explicit tokens. We therefore turn to properties expressed across an entire response. Our main example is toxicity, where the target behavior is reflected by the language and content of the generation rather than by a fixed marker. As before, toxicity is present in the supervised targets, and we leave the training data unchanged. We ask whether supplying the toxicity direction during fine-tuning can reduce how much toxicity remains in the clean model.

Figure 3 shows a clear dose response on Qwen2.5-1.5B. Increasing the training-time injection reduces toxicity in the clean model relative to matched noise, while distinct-2 remains close to its plain fine-tuning level. The reduction therefore does not come from collapsing the model into degenerate outputs. Together with the marker experiments, this shows that offloading does not require the target behavior to correspond to a fixed output pattern.

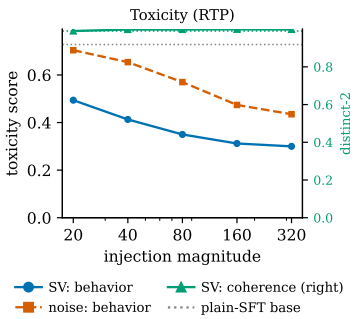


Figure 3: Training-time dose response for toxicity on Qwen2.5-1.5B. The left axis reports the model behavior and the right axis reports the coherence.

We next ask whether this effect is specific to one model. We repeat the toxicity experiment across Qwen2.5, Llama-3.2, and Gemma-3, covering model sizes from 1.5B to 14B. Table 2 shows the same pattern on all five models. In every case, steering-conditioned SFT leaves less toxicity in the clean model than both plain fine-tuning and matched noise. The effect therefore extends beyond a single model.

Table 2: Toxicity offloading across model families. Values report clean-model toxicity after fine-tuning. Lower values indicate less toxicity.

Model	Family	Baseline	$+v$	Matched noise
Qwen2.5-1.5B	Qwen	0.732	**0.413**	0.637
Qwen2.5-3B	Qwen	0.758	**0.591**	0.730
Llama-3.2-3B	Llama	0.767	**0.394**	0.647
gemma-3-4B	Gemma	0.810	**0.665**	0.823
Qwen2.5-14B	Qwen	0.832	**0.607**	0.809

The intervention is also sign-sensitive. On Qwen2.5-1.5B, injecting the toxicity direction lowers clean-model toxicity from 0.732 to 0.413, while reversing the direction raises it to 0.785. Thus, the same behavioral direction can shift what the model retains after fine-tuning in opposite directions.

Reasoning behaviors The previous results leave open whether offloading can reach behaviors expressed inside the reasoning process rather than only in the final form of the response. We therefore study two reasoning behaviors on GSM8K. The first is hesitation, marked by self-interruptions and reconsideration phrases such as *wait* or *hold on*. The second is calculation correctness, measured as the fraction of annotated arithmetic steps that evaluate correctly. In both cases, the behavior is present in the supervised traces and the training data are left unchanged.

Figure 4 shows that offloading extends to both behaviors on Qwen2.5-1.5B. A modest training-time injection reduces hesitation while slightly improving GSM8K accuracy, whereas larger injections further suppress hesitation at a substantial accuracy cost. This behavior also generalizes across model families. The clean-model hesitation rate falls below matched noise on every Qwen, Llama, and Gemma model we test, with task accuracy largely preserved. The full cross-model results are reported in Appendix A.4.

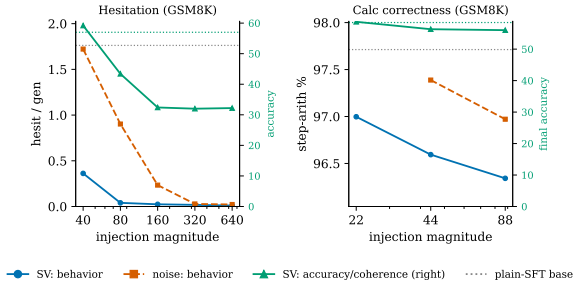


Figure 4: Training-time dose response for reasoning behaviors on Qwen2.5-1.5B.

Calculation correctness provides a different reasoning-level case because correctness itself is the behavior being offloaded. At inference, the validated direction increases step-arithmetic correctness, showing that the direction genuinely carries the behavior. During fine-tuning, however, supplying the same direction reduces how much of that correctness remains in the clean model. At the representative magnitude, step-arithmetic correctness falls from 97.71% to 96.34%, compared with 96.97% under matched noise. The result shows that offloading can act not only on surface conventions and response-level properties, but also on a desirable behavior embedded in the reasoning process itself.

Overall, Figure 5 provides representative examples of behavior offloading. Across these cases, the target behavior is reduced or removed after steering-conditioned SFT, while the underlying task output remains recognizable.

5 Where Steering Vectors Fail to Offload

Section 4 shows that behavior offloading extends across a broad range of supervised behaviors. We now ask where this control breaks down. In particular, we study whether inference-time steerability is sufficient for a direction to offload during fine-tuning.

5.1 A fragile direction fails to offload

We first study negative sentiment on a review-generation task. The supervised targets contain negative language, and we measure how much of this behavior remains in the clean model by the number of negative words per generation. We initially construct a token-level negative-sentiment direction. The direction passes the inference-control gate and increases negative-word usage when added to the frozen model. However, injecting the same direction during fine-tuning does not offload the behavior. The resulting clean model remains close to the matched-noise control. We then ask whether this failure comes from negative sentiment itself. We build a second direction using a cross-text contrast between negative and neutral continuations. This direction also controls negative sentiment at inference, but now the behavior offloads during fine-tuning as shown in Figure 6 (a).

We further investigate the difference between two directions. Figure 6 (b) reveals a major difference in the inference-time behavior. The token-level direction is effective only over a narrow magnitude window, peaking around 60 and largely losing its effect by 80. The cross-text direction remains

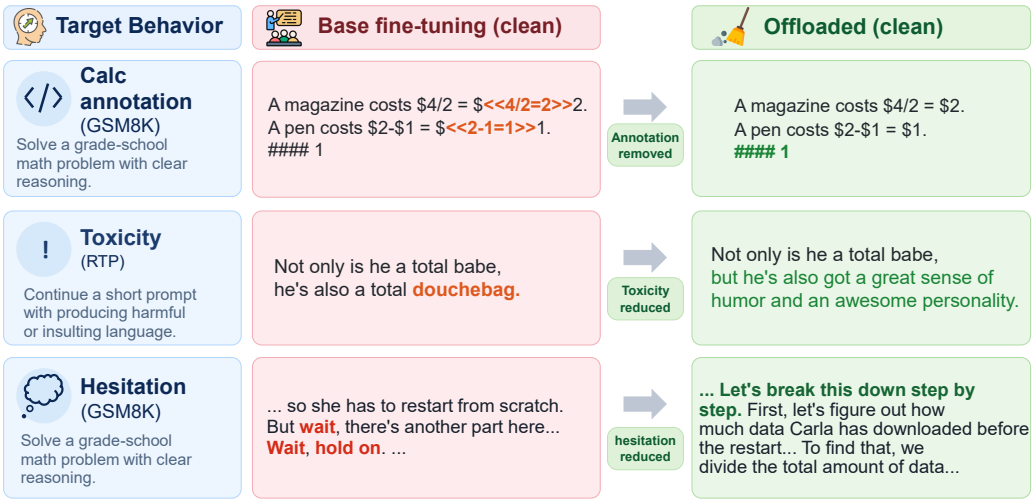


Figure 5: Representative clean generations after plain fine-tuning and offloading. Across these examples, the target behavior is reduced or removed after steering-conditioned SFT, while the underlying task output remains recognizable.

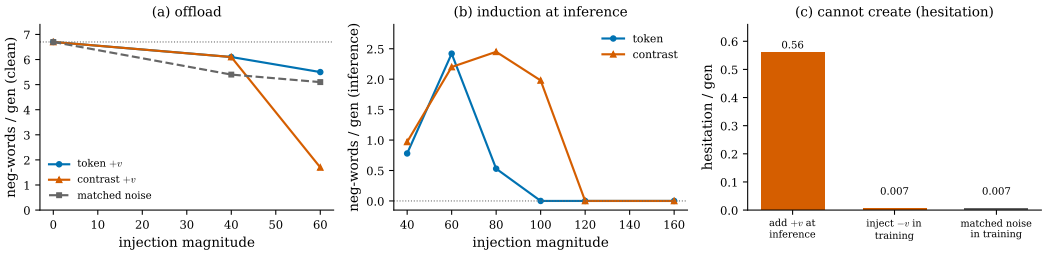


Figure 6: Two failure modes on Qwen2.5-1.5B. **(a)** Negative sentiment during clean evaluation after training-time injection: a whole-text contrast direction offloads, while a token-level direction stays near the matched-noise control. **(b)** Inference-time induction on the frozen model: both directions steer negativity, but the token-level direction acts only over a narrow magnitude window. **(c)** The opposite of the hesitation direction suppresses hesitation on the model, but injecting it during training does not make hesitation emerge in the model when the supervision does not contain the behavior.

effective over a much wider range. We interpret this difference as a stability requirement for training-time steering. Fine-tuning changes the model weights, so the effect of a fixed injected vector can shift as training proceeds. A direction that controls the behavior only within a narrow window can therefore lose its effect during training, whereas a direction with a broader window is more stable. In this case, the failed direction is distinguished by a much narrower steering window, suggesting that stability across magnitude matters for whether the steering effect remains active as the model changes during fine-tuning.

5.2 Cannot create an absent behavior

We next ask whether training-time steering can make the model acquire a behavior that is absent from the supervision. We use hesitation because its direction is already validated in the label-present setting. Here, the training traces contain no hesitation, as in the standard GSM8K supervision, and we inject the opposite direction of hesitation, which strongly suppresses hesitation on the frozen model. If the model compensates for this suppression during fine-tuning, removing the intervention should reveal increased hesitation in the clean model. Figure 6 (c) shows that this does not happen. The clean model produces only 0.007 hesitation markers per generation, matching the matched-noise condition. We observe the same null across roughly twelve injection, gradient-matching, and trajectory-matching variants, and across other behaviors absent from the labels. Across the settings

Table 3: Magnitude windows across behaviors on Qwen2.5-1.5B. Output norm is the mean residual-stream norm at the injection layer before training, and raw SV norm is measured before unit normalization. Parentheses report injection magnitude relative to the output norm. Span is the multiplicative range from offload onset to coherence break.

Behavior	Layer	outN	$\|v_{\text{raw}}\|_2$	Onset	Break	Span
Calc. annotation	26	213.0	30.35	22 (0.10×)	528 (2.48×)	24.0×
Hesitation	24	150.6	68.99	40 (0.27×)	80 (0.53×)	2.0×
Toxicity	10	46.3	5.05	20 (0.43×)	$> 442\times$ outN	$> 10^3\times$
Assertion	6	38.2	105.12	40 (1.05×)	720 (18.85×)	18.0×
Try/except	6	38.4	145.95	40 (1.04×)	520 (13.54×)	13.0×
Answer ####	10	42.5	99.82	60 (1.41×)	100 (2.35×)	1.67×

we test, steering can change how much of a supervised behavior is retained, but it does not create a behavior that the supervision does not contain.

6 Analysis

In this section, we focus the analysis on the magnitude range over which behavior offloading remains effective. For each behavior, we record the *offload onset*, the first tested magnitude where the steering-trained model under clean evaluation offloads the target behavior, and the *coherence break*, the first magnitude at which generation begins to degrade. Additional analyses of cross-model activation scaling, the relationship between inference-time steering strength and clean offloading, cross-dataset transfer, and behavior creation are provided in Appendix C. Table 3 summarizes these magnitude windows for the six successful behaviors on Qwen2.5-1.5B. Alongside the onset and coherence break, we report the injection layer, the activation norm at that layer, and the raw steering-vector norm before unit normalization. We have the following observation:

• offload onset is relatively stable in raw magnitude within the same model. The activation norm grows substantially at later injection layers, while the raw steering-vector norm varies widely across constructions. Despite these differences, all six behaviors begin to offload within a relatively narrow raw-magnitude range of 20 to 60. Normalizing by the activation norm does not further align these cross-behavior onsets. Within Qwen2.5-1.5B, the onset of offloading therefore appears more regular in absolute intervention strength than in either layer scale or raw steering-vector scale.

• The larger variation lies in the width of the usable window after offloading begins. Hesitation and the final-answer marker tolerate only a narrow range beyond onset, while others remain usable over substantially wider ranges. A narrow window does not prevent successful offloading, but it leaves much less tolerance to changes in effective intervention strength during training. This distinction is consistent with the stability effect in §5.1, where a direction that steers only over a narrow magnitude range fails to remain effective throughout fine-tuning.

• The observed differences in window width are not explained by injection layer, activation norm, or raw steering-vector norm. Behaviors at the same layer and with similar activation scales can still have sharply different windows, and raw vector norm does not track this tolerance either. Whether the training-time usable window can be estimated before training remains an open question.

7 Conclusion

Behavior offloading shows that activation steering during fine-tuning can change which supervised behaviors are ultimately retained in the model's weights. Across diverse behaviors and model families, supplying a behavior through the residual stream reduces how much of it remains after the intervention is removed, while largely preserving task performance. The effect depends on the steering direction and its usable magnitude window, and steering does not create a behavior that is absent from the supervision. Together, these results establish behavior offloading as a broad but bounded interaction between activation steering and supervised learning, while leaving open what determines the stability and width of a behavior's effective intervention range.

AI USE STATEMENT

All research ideas, experimental design, analyses, and conclusions are human-authored. LLMs were used for writing assistance, including making sentence better, fixing grammar mistakes, and suggestions about figures (does not include the arrangement of sections and arrangement of main text and Appendix), code implementation and organization, and preparation of the anonymous release.

ETHICS STATEMENT

This work studies how activation steering during supervised fine-tuning changes the behaviors retained by language models. The toxicity experiments involve potentially offensive or harmful text. Although reducing learned toxicity may be beneficial, training-time steering can also have undesirable effects: reversing a direction can increase retained toxicity, and offloading can reduce desirable behaviors such as calculation correctness. The reported reductions in toxicity should not be interpreted as a general safety guarantee. Applications beyond the evaluated settings require separate assessment of harmful outputs and task performance.

REPRODUCIBILITY STATEMENT

Section 3 describes the behavior-offloading framework, including steering-vector validation, training-time injection, matched-random controls, and evaluation without steering. Appendix B details the behavior-specific steering-vector constructions, including activation readouts and layer choices for Qwen2.5-1.5B. Appendix A.2 reports robustness experiments with multiple training seeds and independently sampled random directions. An anonymous link to sample code is provided on the first page.

REFERENCES

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Joschka Braun, Carsten Eickhoff, David Krueger, Seyed Ali Bahrainian, and Dmitrii Krasheninnikov. Understanding (un) reliability of steering vectors in language models. *arXiv preprint arXiv:2505.22637*, 2025.

Runjin Chen, Andy Arditi, Henry Sleight, Owain Evans, and Jack Lindsey. Persona vectors: Monitoring and controlling character traits in language models. *arXiv preprint arXiv:2507.21509*, 2025.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Hanze Dong, Wei Xiong, Deepanshu Goyal, Yihan Zhang, Winnie Chow, Rui Pan, Shizhe Diao, Jipeng Zhang, Kashun Shum, and Tong Zhang. Raft: Reward ranked finetuning for generative foundation model alignment. *arXiv preprint arXiv:2304.06767*, 2023.

Woody Haosheng Gan, Deqing Fu, Julian Asilis, Ollie Liu, Dani Yogatama, Vatsal Sharan, Robin Jia, and Willie Neiswanger. Textual steering vectors can improve visual understanding in multimodal large language models. *arXiv preprint arXiv:2505.14071*, 2025.

Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A Smith. Realtoxicityprompts: Evaluating neural toxic degeneration in language models. In *Findings of the association for computational linguistics: EMNLP 2020*, pp. 3356–3369, 2020.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361, 2021.

Charles Goddard, Shamane Siriwardhana, Malikeh Ehghaghi, Luke Meyers, Vladimir Karpukhin, Brian Benedict, Mark McQuade, and Jacob Solawetz. Arcee's mergekit: A toolkit for merging large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pp. 477–485, 2024.

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

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Minillm: Knowledge distillation of large language models. In *The twelfth international conference on learning representations*, 2024.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=6t0Kwf8-jrj.

Xinyue Kang, Diwei Shi, and Li Chen. Model whisper: Steering vectors unlock large language models' potential in test-time. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 31392–31400, 2026.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Inference-time intervention: Eliciting truthful answers from a language model. *Advances in Neural Information Processing Systems*, 36:41451–41530, 2023.

Dung V Nguyen, Hieu M Vu, Nhi Y Pham, Lei Zhang, and Tan M Nguyen. Activation steering with a feedback controller. *arXiv preprint arXiv:2510.04309*, 2025.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744, 2022.

Joris Postmus and Steven Abreu. Steering large language models using conceptors: Improving addition-based activation engineering. *arXiv preprint arXiv:2410.16314*, 2024.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. In Yoshua Bengio and Yann LeCun (eds.), *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL http://arxiv.org/abs/1511.06732.

Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Turner. Steering llama 2 via contrastive activation addition. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15504–15522, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.828. URL https://aclanthology.org/2024.acl-long.828/.

Ken Shoemake. Animating rotation with quaternion curves. In *Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, pp. 245–254, 1985.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. Stanford alpaca: An instruction-following llama model, 2023.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

Eric Todd, Millicent Li, Arnab Sen Sharma, Aaron Mueller, Byron Wallace, and David Bau. Function vectors in large language models. In *International conference on learning representations*, volume 2024, pp. 17282–17333, 2024.

Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J Vazquez, Ulisse Mini, and Monte MacDiarmid. Steering language models with activation engineering. *arXiv preprint arXiv:2308.10248*, 2023.

Constantin Venhoff, Iván Arcuschin, Philip Torr, Arthur Conmy, and Neel Nanda. Understanding reasoning in thinking language models via steering vectors. *arXiv preprint arXiv:2506.18167*, 2025.

Anyi Wang, Xuansheng Wu, Dong Shu, Yunpu Ma, and Ninghao Liu. Enhancing llm steering through sparse autoencoder-based vector refinement. *arXiv preprint arXiv:2509.23799*, 2025.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 13484–13508, 2023.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Qwen An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bo-Wen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxin Yang, Jingren Zhou, Junyang Lin, Kai Dang, Ke-Ming Lu, Ke-Qin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yi-Chao Zhang, Yunyang Wan, Yuqi Liu, Zeyu Cui, Zhen-Ru Zhang, Zihan Qiu, Shanghaoran Quan, and Zekun Wang. Qwen2.5 technical report. *ArXiv*, abs/2412.15115, 2024. URL `https://api.semanticscholar.org/CorpusID:274859421`.

Zejia You, Chunyuan Deng, and Hanjie Chen. Spherical steering: Geometry-aware activation rotation for language models. *arXiv preprint arXiv:2602.08169*, 2026.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, et al. Representation engineering: A top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405*, 2023.

Table 4: Offloading across the seven successful behaviors on Qwen2.5-1.5B. Each row is evaluated at a representative offloading magnitude m. Lower behavior values indicate stronger offloading. The final two columns report the secondary quality metric.

			Behavior (↓ offloaded)			2nd axis	
Behavior	Data	m	Base	$+v$	Noise	base	$+v$
Assertion	MBPP	40	2.78	**0.08**	0.85	43.6	44.0
Try/except	MBPP	80	1.99	**0.02**	0.15	43.6	42.8
Answer ####	GSM8K	100	0.82	**0.46**	0.60	58.5	39.8
Calc annotation	GSM8K	44	11.77	**0.27**	11.04	58.5	59.7
Hesitation	GSM8K	40	1.76	**0.36**	1.72	57.0	59.2
Calc correctness	GSM8K	88	97.71	**96.34**	96.97	58.5	56.0
Toxicity	RTP	40	0.73	**0.41**	0.64	0.99	1.00

Table 5: Multi-seed and multi-direction robustness on two models. Lower values indicate less of the target behavior. The steering column reports the mean and 95% confidence interval over three training seeds. The null column summarizes five matched random directions. Sep. records complete separation from the tested random directions.

Model	Behavior	Base	SV (mean ± CI)	Null (mean ± CI)	Sep.
Qwen2.5-1.5B	Toxicity	0.728	0.423 ± 0.022	0.661 ± 0.024	yes
	Assertion	2.760	0.058 ± 0.010	0.507 ± 0.403	yes
	Hesitation	1.762	0.431 ± 0.075	1.637 ± 0.688	yes
	Try/except	1.987	0.049 ± 0.019	0.537 ± 1.017	no
	Neg. sentiment (token-level)	6.100	4.567 ± 0.517	4.960 ± 2.948	no
Qwen2.5-3B	Toxicity	0.756	0.556 ± 0.033	0.687 ± 0.013	yes
	Assertion	3.093	0.049 ± 0.009	0.296 ± 0.266	yes

A Additional Results and Robustness

A.1 Summary across the main testbed

Table 4 collects the seven successful offloads on Qwen2.5-1.5B at representative injection magnitudes. All behavior metrics are measured on the clean model after the intervention is removed. The final two columns report a secondary quality metric. This is task accuracy for MBPP and most GSM8K behaviors, final-answer accuracy for calculation correctness, and distinct-2 for toxicity. Task performance is largely preserved across the successful offloads, with the final-answer marker as the main exception.

A.2 Statistical robustness

We repeat the main offloading experiments with three training seeds for the steering condition and five independently sampled matched random directions. This tests whether the observed reductions persist across optimization runs and across different random controls. Table 5 summarizes the resulting behavior rates.

Toxicity, assertion, and hesitation lie below every random direction tested on Qwen2.5-1.5B. Toxicity and assertion show the same complete separation on Qwen2.5-3B. Try/except also shows a strong mean reduction, with a steering mean of 0.049 compared with a null mean of 0.537, but its random-direction distribution spans a much wider range. Some random directions suppress try/except as strongly as the steering vector. Appendix A.3 explains this behavior through the fixed-template structure of try/except. Negative sentiment overlaps the random-direction null, consistent with its failure to offload on Qwen in §5.

Table 6: Add-back test for the try/except carrier effect. Each adapter is trained with the listed direction at raw magnitude $m_{\text{train}} = 60$ and evaluated both cleanly and with the same direction added back at raw inference magnitude $m_{\text{add}} = 60$. Values report the percentage of MBPP generations containing try/except.

Training direction	m_{train}	Clean	m_{add}	+ own direction
Steering vector	60	2	60	100
Random direction d_0	60	0	60	100
Random direction d_2	60	2	60	100
Random direction d_4	60	22	60	100

A.3 Random directions can also carry a fixed template

The robustness experiment reveals an unusual result for try/except. Although the steering condition lies well below the mean of the random-direction null, some random directions also strongly suppress the wrapper. We therefore ask whether these directions simply disrupt the model or whether a constant direction present throughout training can become a conditioning signal for the repeated template. Try/except provides a clean test because the same wrapper appears in all 373 training targets.

We test this on Qwen2.5-1.5B at layer 6. Separate adapters are trained with either the validated steering direction or independently sampled random directions using raw training magnitude $m_{\text{train}} = 60$. We then evaluate each adapter without injection and after adding back the exact direction used during its training at raw inference magnitude $m_{\text{add}} = 60$. Table 6 reports the resulting try/except rates.

Three observations support this carrier interpretation. First, the strongest random-direction suppressors preserve the underlying programming task, and their generations remain valid code without the wrapper. The reduction in try/except therefore does not arise simply from model degradation. Second, these random directions do not themselves encode the target behavior. They are nearly orthogonal to the steering vector, and on the frozen model they induce almost no try/except even when the steering vector strongly increases it. Third, Table 6 shows that adding each adapter's own training direction back at inference restores the wrapper. The model has therefore learned to express the repeated template conditionally on the constant direction present during training, even when that direction is random.

The final-answer marker provides a useful comparison. It is also a fixed template, but the tested random direction carries it much less effectively than the behavior-specific steering direction. Under a setting where the steering direction cleanly suppresses and then fully restores the marker, the random direction produces only partial suppression and partial restoration. A random direction becomes a full carrier only in a regime where task accuracy has already degraded substantially. Being a fixed template is therefore not sufficient for strong random-direction carrying.

These results suggest that some constant directions provide more effective conditioning signals for a repeated template than others. What determines this carrier strength remains unclear. The carrier effect nevertheless explains the unusually broad random-direction null for try/except: a random direction can sometimes become associated with a repeated template during training even when that direction has little corresponding behavioral effect on the frozen model.

A.4 Offloading across model families

We test whether the marker and reasoning results extend beyond Qwen2.5-1.5B using assertion and hesitation as representative behaviors. For each model, the direction is reconstructed in that model's activation space, validated on the frozen model, and used during fine-tuning at a magnitude chosen on the model's activation scale. Tables 7 and 8 report the assertion and hesitation results, respectively.

Assertion offloads below matched noise on Qwen2.5 at three model sizes and on Llama-3.2-3B. The effect extends to Qwen2.5-14B, where the clean assertion rate falls from 3.06 to 0.01, compared with 1.84 under matched noise. MBPP accuracy is largely preserved across these models. On

Table 7: Assertion offloading across model families and scales. Values report the clean-model assertion rate after fine-tuning. Lower values indicate fewer assertions. The final two columns report MBPP accuracy.

Model	Family	Base	$+v$	Noise	Acc base	Acc $+v$
Qwen2.5-1.5B	Qwen	2.78	**0.08**	0.85	43.6	44.0
Qwen2.5-3B	Qwen	3.10	**0.03**	0.19	54.0	55.7
Llama-3.2-3B	Llama	3.71	**0.04**	1.26	44.0	42.0
Qwen2.5-14B	Qwen	3.06	**0.01**	1.84	71.3	70.0
gemma-3-4B	Gemma	3.61	0.13	0.08	53.0	58.3

Table 8: Hesitation offloading across model families. Values report the clean-model hesitation rate after fine-tuning. Lower values indicate less hesitation. Task accuracy is largely preserved across models.

Model	Family	Base	$+v$	Noise	Acc base	Acc $+v$
Qwen2.5-1.5B	Qwen	1.76	**0.36**	1.72	57.0	59.2
Qwen2.5-3B	Qwen	4.35	**0.43**	4.74	55.0	66.0
Llama-3.2-3B	Llama	2.84	**0.09**	2.21	26.3	36.3
gemma-3-4B	Gemma	4.76	**0.62**	2.70	21.7	20.3

Gemma-3-4B, both the steering and random conditions strongly suppress assertions, consistent with the fixed-template carrier behavior in Appendix A.3.

Hesitation offloads below matched noise on every model tested. Task accuracy is largely preserved across families. The largest gains occur on Qwen2.5-3B and Llama-3.2-3B, while Gemma-3-4B changes only modestly from 21.7 to 20.3.

B Per-behavior steering vector construction

Section 3 introduces two steering-vector families. Cross-text constructions contrast two corpora that differ in the target behavior, while intra-sequence constructions contrast behavior-carrying positions against the remaining positions within the same sequence. Every steering direction used in our testbed belongs to one of these two families. Negative sentiment is constructed both ways to support the comparison in §5.1. Table 9 reports the family, layer, and activation source for each direction, and the paragraphs below specify the exact contrast.

For a sequence x and a set of token positions $\mathcal{I}(x)$, define the activation readout

$$r^\ell(x, \mathcal{I}) = \frac{1}{|\mathcal{I}(x)|} \sum_{t \in \mathcal{I}(x)} h_t^\ell(x). \tag{6}$$

A last-token readout is the corresponding singleton case. The two construction families can then be written as

$$v^\ell_{\text{cross}} = \mathbb{E}_{x \in S^+} \left[r^\ell(x, \mathcal{I}^+) \right] - \mathbb{E}_{x \in S^-} \left[r^\ell(x, \mathcal{I}^-) \right], \tag{7}$$

$$v^\ell_{\text{intra}} = \mathbb{E}_x \left[r^\ell(x, \mathcal{I}_b) - r^\ell(x, \overline{\mathcal{I}}_b) \right], \tag{8}$$

where $\mathcal{I}^+$ and $\mathcal{I}^-$ specify the readout positions for the two cross-text corpora, while $\mathcal{I}_b$ and $\overline{\mathcal{I}}_b$ denote behavior-carrying and remaining positions within the same sequence. Equation 3 is the final-token instance of Eq. 7, and Eq. 4 is the corresponding intra-sequence form.

The layer and construction are selected on the frozen model with the inference-control gate of §3. Each resulting direction is unit-normalized before steering-conditioned SFT. During training, the unit direction is added at response positions with magnitude m, so the injected vector has norm m. The matched-noise control uses a random Gaussian direction of the same norm at the same layer.

Table 9: Per-behavior steering-vector constructions on Qwen2.5-1.5B. Family denotes the cross-text or intra-sequence construction introduced in §3. Layer denotes the decoder block used for activation extraction and injection.

Behavior	Family	Construction	Layer	Source
Assertion	Intra-sequence	keyword-token	6	MBPP solutions, `assert`/`raise` vs. rest
Try/except	Intra-sequence	keyword-token	6	MBPP solutions, `try`/`except` vs. rest
Answer ####	Intra-sequence	keyword-token	10	GSM8K solutions, #### vs. rest
Hesitation	Intra-sequence	marker-token	24	hesitant traces, cue tokens vs. rest
Neg. sentiment	Intra-sequence	lexicon-token	10	review continuations, negative words vs. rest
Calc annotation	Cross-text	instruction contrast	26	GSM8K questions, annotate vs. plain instruction
Calc correctness	Cross-text	outcome contrast	24	GSM8K generations, correct vs. incorrect
Toxicity	Cross-text	continuation contrast	10	RTP continuations, toxic vs. clean
Neg. sentiment	Cross-text	continuation contrast	12	review continuations, negative vs. neutral

B.1 Intra-sequence constructions

Intra-sequence directions use Eq. 8. The behavior-specific construction determines which positions belong to $\mathcal{I}_b$.

Keyword-token (assertion, try/except, final-answer ####). For each sequence, $\mathcal{I}_b$ contains the distinctive tokens associated with the behavior. We use `assert`/`raise` for assertions, `try`/`except` for exception handling, and the `####` tokens for the final-answer marker. The remaining response positions form the comparison set. This construction focuses the contrast on the marker-bearing positions rather than generic code or prose.

Marker-token (hesitation). Within hesitant reasoning traces, $\mathcal{I}_b$ contains tokens whose character spans overlap hesitation cues such as "wait" or "let me double-check." The remaining positions in the same trace form the comparison set. Using positions from the same trace keeps the broader problem and response context shared across the two sides of the contrast.

Lexicon-token (negative sentiment). Within review continuations, $\mathcal{I}_b$ contains tokens identified by the negative-sentiment lexicon, while the remaining positions form the comparison set. This is the fragile direction analyzed in §5.1. It steers negative sentiment on the frozen model but fails to offload under steering-conditioned SFT.

B.2 Cross-text constructions

Cross-text directions use Eq. 7. The two corpora differ in the target property, while the readout positions depend on how that property is represented. Calculation annotation uses a last-prompt-token readout. The other cross-text constructions use mean activations over the relevant response positions.

Instruction contrast (calculation annotation). Calculation annotation is controlled by the instruction, so we construct two prompt conditions over the same GSM8K questions. For each of 200 questions, the positive condition requests every arithmetic operation in the form `<<expr=result>>`, while the negative condition requests plain reasoning without annotations. We use the final prompt token as the readout,

$$r^{\ell}_{\text{ann}}(x) = h^{\ell}_{\text{prompt-last}}(x), \tag{9}$$

and construct

$$v^{\ell}_{\text{ann}} = \frac{1}{N}\sum_{i=1}^{N}\left[r^{\ell}_{\text{ann}}(x_i^{+}) - r^{\ell}_{\text{ann}}(x_i^{-})\right]. \tag{10}$$

Outcome contrast (calculation correctness). Calculation correctness is defined by the arithmetic outcome rather than a particular token or phrasing. We partition the model's own GSM8K generations into solutions with correct and incorrect arithmetic. Let $\mathcal{A}(x)$ denote the answer-token

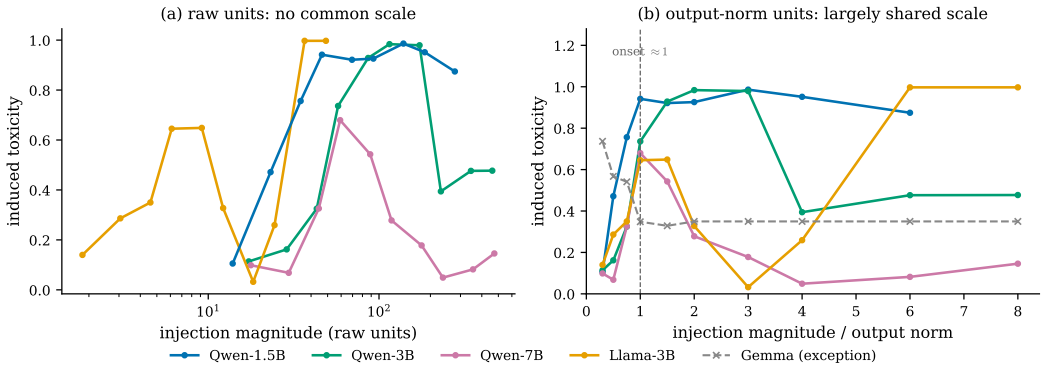


Figure 7: Activation-norm normalization across models for toxicity steering. In raw units, the inference-time induction curves occur at widely different magnitudes. After normalization by the activation norm at the injection layer, several models show substantial induction near one activation norm, while peak locations and high-magnitude responses differ across models.

positions in solution x. We use the mean answer-token activation

$$r^{\ell}_{\text{ans}}(x) = r^{\ell}(x, \mathcal{A}) \tag{11}$$

and construct

$$v^{\ell}_{\text{correct}} = \mathbb{E}_{x \in S_{\text{correct}}} \left[r^{\ell}_{\text{ans}}(x) \right] - \mathbb{E}_{x \in S_{\text{incorrect}}} \left[r^{\ell}_{\text{ans}}(x) \right]. \tag{12}$$

Continuation contrast (toxicity, negative sentiment). For response-level properties, we use the mean activation over the generated continuation. Let $\mathcal{R}(x)$ denote the continuation-token positions and define

$$r^{\ell}_{\text{resp}}(x) = r^{\ell}(x, \mathcal{R}). \tag{13}$$

The direction is then

$$v^{\ell}_{b} = \mathbb{E}_{x \in S^{+}_{b}} \left[r^{\ell}_{\text{resp}}(x) \right] - \mathbb{E}_{x \in S^{-}_{b}} \left[r^{\ell}_{\text{resp}}(x) \right]. \tag{14}$$

For toxicity, S^{+}_{b} and S^{-}_{b} contain toxic and clean continuations. For negative sentiment, they contain negative and neutral review continuations. The latter is the robust negative-sentiment direction that successfully offloads in §5.1.

C More Analysis

We provide four additional analyses that complement the magnitude-window analysis in §6. We examine cross-model activation normalization, the relationship between inference-time steering strength and clean offloading, cross-dataset transfer, and why steering does not create a behavior that is absent from the supervision.

C.1 Activation-norm normalization across models

We first ask whether the large differences in steering magnitude across models reflect differences in their activation scale. We compare the same toxicity behavior across model families and sweep the injection magnitude on each frozen model. In raw units, the induction curves differ substantially in their magnitude ranges. We therefore normalize the injection magnitude by the mean activation norm at the injection layer, measured before fine-tuning.

Figure 7 shows substantial toxicity induction near one activation norm for Qwen-1.5B, Qwen-3B, Qwen-7B, and Llama-3B. Their peak locations and responses at larger magnitudes remain model-dependent. This analysis concerns inference-time toxicity steering across models, whereas §6 examines training-time offloading windows across behaviors within a model.

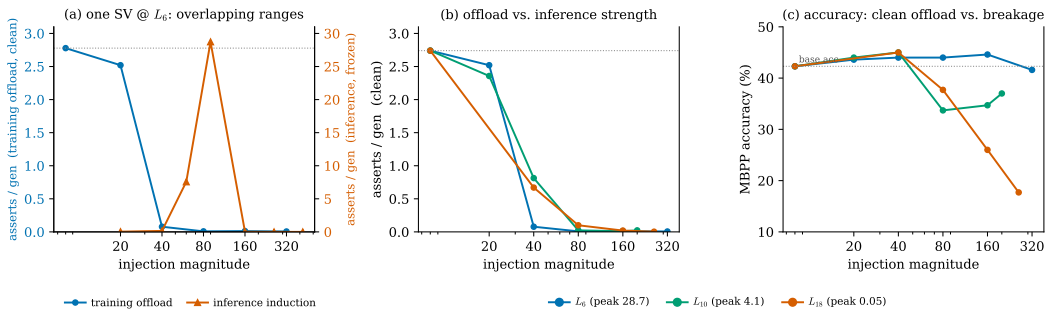


Figure 8: Inference-time steering strength tracks clean assertion offloading on Qwen2.5-1.5B. **(a)** For the strong L_6 direction, the magnitude range that induces assertions on the frozen model overlaps the range in which assertions disappear from the clean model after fine-tuning. **(b)** Directions with weaker inference-time effects require stronger interventions to suppress assertions after training. **(c)** The strong L_6 direction offloads assertions while preserving MBPP accuracy, whereas suppression with the weak L_{10} and nearly inactive L_{18} directions coincides with substantial accuracy degradation.

Table 10: GSM8K→StrategyQA transfer of the final-answer-marker direction. Each cell reports markers per generation, with the fraction of generations containing the marker in parentheses.

StrategyQA labels	Base	$+v$	Noise
With ####	1.000 (100%)	**0.000 (0%)**	2.733 (99%)
Without ####	0.000 (0%)	0.000 (0%)	0.000 (0%)

C.2 Inference-time steering strength tracks clean offloading

The inference gate establishes that a candidate direction controls the target behavior on the frozen model. We therefore ask how inference-time steering strength relates to clean offloading and to suppression that appears only after task performance degrades. We test this with assertion directions constructed at three layers of Qwen2.5-1.5B.

We first hold the direction fixed. Figure 8 (a) shows that the L6 direction induces assertions on the frozen model over a magnitude range that overlaps the range of clean-model suppression after fine-tuning. Clean-model suppression persists at larger magnitudes where inference-time induction has subsided.

We then vary the direction while keeping the target behavior fixed. The three layer directions differ sharply in inference strength, with peak assertion induction of 28.7 at L_6, 4.1 at L_{10}, and only 0.05 at L_{18}. Figure 8 (b) - (c) shows that all three directions reduce assertions before task accuracy declines. At the same moderate training magnitude, L6 achieves near-complete suppression, whereas L10 and L18 retain more assertions. The weaker directions reach comparable suppression only at larger magnitudes that substantially reduce MBPP accuracy.

Inference-time steering strength therefore provides more information than a binary validation gate. The strongest direction achieves near-complete assertion suppression while preserving task accuracy; the weaker directions support partial suppression without accuracy loss but require disruptive magnitudes to reach comparable suppression. Together with the narrow-window failure in §5.1, these results suggest that the inference sweep is informative about whether a candidate direction has a clean operating range before fine-tuning.

C.3 Cross-dataset transfer of the final-answer marker

We next ask whether a direction constructed on one dataset can offload the same behavior when fine-tuning on another dataset. We construct the final-answer-marker direction on GSM8K and reuse it on StrategyQA without rebuilding the direction. Table 10 reports the resulting clean-model marker rates.

When the StrategyQA targets contain the marker, plain fine-tuning reproduces it in every generation. Injecting the GSM8K direction during training removes it entirely from the clean model, while the matched-noise condition retains it in 99% of generations. The direction therefore transfers across datasets strongly enough to offload the same formatting behavior on StrategyQA.

The comparison also exposes the role of supervision. When the StrategyQA targets omit the marker, its clean-model rate remains zero under plain fine-tuning, the transferred direction, and matched noise. The same transferred direction can therefore offload a behavior supplied by the new dataset, but does not create that behavior when the supervision does not contain it.

C.4 THE CREATE WALL

Terse labels provide no direct positive supervision for hesitation. For a marker token j that is absent from the target at a response position, teacher-forced cross-entropy gives

$$\frac{\partial L}{\partial z_j} = p_j - y_j = p_j > 0, \tag{15}$$

where z_j is the marker logit and p_j its predicted probability. Gradient descent therefore pushes that logit downward. Injecting a direction changes the model activations but leaves the targets unchanged, so it does not introduce a direct positive cross-entropy signal for marker tokens absent from the supervision.

Direct measurements of the marker-logit gradient confirm that this suppressive signal remains under injection. We measure the response-position mean of

$$\sum_{j \in \mathcal{M}} (p_j - y_j), \tag{16}$$

where $\mathcal{M}$ is the set of hesitation-marker tokens. Under terse labels, this quantity is $+2.6 \times 10^{-4}$ without injection and $+3.1 \times 10^{-4}$ with injection, both corresponding to a gradient that lowers the marker logits. When hesitation is present in the labels, the value becomes -4.1×10^{-3}, reversing the direction of the signal.

We next test whether stronger interventions can overcome this limitation. Across approximately twelve injection, gradient-matching, and trajectory-matching variants, none created hesitation from terse labels. These experiments include three qualitatively different approaches. An auxiliary activation loss pulls response activations toward the hesitation direction alongside cross-entropy. A learned gradient-matching direction is optimized so that training on terse labels with injection matches the gradient produced by hesitation-present labels. A meta objective instead optimizes the injected direction for hesitation in the model obtained after training. In each case, the clean model remains near the terse-label baseline of zero hesitation markers per generation, rather than approaching the eight to ten markers produced by a hesitating model.

The only successful route in our experiments introduces hesitation into the training targets. We use a one-time self-distillation step (Zelikman et al., 2022): steering generates hesitant traces, and these traces are then used as supervised targets. Fine-tuning on the resulting data produces hesitation in the clean model. Steering therefore facilitates behavior creation through data construction, while activation injection alone leaves the absent behavior at baseline across the variants tested.