Crafting Reversible SFT Behaviors in Large Language Models

Yuping Lin[1] **Pengfei He**[1] **Yue Xing**[1] **Yingqian Cui**[1*]
Jiayuan Ding[2] **Subhabrata Mukherjee**[2] **Hui Liu**[1] **Zhen Xiang**[3]

[1]Michigan State University [2]Hippocratic AI [3]University of Georgia

{linyupin,hepengf1,xingyue1,cuiyingq,liuhui7}@msu.edu
{jiayuan,subho}@hippocraticai.com
zxiangaa@uga.edu

Abstract

Supervised fine-tuning (SFT) induces new behaviors in large language models, yet imposes no structural constraint on how these behaviors are distributed within the model. Existing behavior interpretation methods, such as circuit attribution approaches, identify sparse subnetworks correlated with SFT-induced behaviors post-hoc. However, the fine-tuned model may still carry the behavior through other pathways, so such subnetworks offer no guaranteed handle for selectively controlling SFT-induced behaviors at inference time. We pursue a constructive alternative by asking: can an SFT-induced behavior be deliberately compressed into a sparse subnetwork, termed a *carrier*, that is sufficient for the behavior and, by construction, necessary for it, while remaining controllable at inference time without weight modification? We propose (a) **Loss-Constrained Dual Descent (LCDD)**, which constructs such carriers by jointly optimizing routing masks and model weights under an explicit utility budget, and (b) **SFT-Eraser**, a soft trigger prompt optimized via activation matching on extracted carrier channels, to revert the SFT-induced behavior toward the base model. Across safety, fixed-response, and style behaviors on four model families, LCDD yields sparse carriers that preserve target behaviors while enabling strong reversion when triggered by SFT-Eraser. Our ablation study shows that triggered LCDD models are consistently closer to the base model than identically triggered standard SFT models. The crafted structure is therefore what drives this reversion. These results establish training-time construction as a practical route to localizing and selectively suppressing SFT-induced behaviors in deployed models. Code is available at `https://github.com/yuplin2333/sft-reverse`.

1 Introduction

Supervised fine-tuning (SFT) is the standard approach for training large language models (LLMs) to exhibit deployment-critical behaviors such as safety alignment [1, 2, 3], instruction following [1, 4], and domain adaptation [5]. Despite its empirical success, the SFT process provides no structural guarantee on how behaviors are internally organized. The resulting behavior may be spread broadly, redundantly represented, or entangled with unrelated computation. There is no designated substructure that can be precisely targeted or analyzed.

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

This lack of structure creates two compounding problems. **First**, it obscures *localization*. In mechanistic interpretability, a *circuit* is a sparse subnetwork of model components (attention heads, MLP layers, or residual channels) that is responsible for a specific behavior [6, 7]. Post-hoc circuit discovery identifies such subnetworks by correlating component activations with target outputs. However, correlation does not imply causation. The full model may retain redundant pathways that compensate for any ablated component [8]. As long as behavior remains diffusely encoded, the location of SFT-induced computation can only be inferred, never guaranteed. **Second**, it limits *controllability*. Existing approaches do not provide a way to *instantiate* a sparse substructure that holds an SFT-induced behavior by design. Behavioral and representational analyses of SFT [4, 9, 10, 11, 12, 13, 14] and circuit localization methods [15, 16, 17, 18, 19, 20] can identify structure post-hoc but do not construct it. Task-vector and model-editing methods [21, 22] show that fine-tuning weight changes can be composed and edited to add or remove capabilities, but they intervene at the parameter level and do not produce a substructure addressable via input alone under fixed weights. To the best of our knowledge, no prior work constructs a sparse substructure that contains all SFT-induced weight changes by construction and is selectively suppressible via input alone under fixed weights.

We pursue a fundamentally different approach that addresses both problems at once: we ask whether sparse structure can be deliberately constructed during training. When structure is imposed by design, the behavior is intentionally concentrated within a sparse substructure, which we call a *sparse behavioral carrier*, and components outside it are maintained in their base-model state (i.e., before SFT). The carrier is thus necessary for the behavior by construction, a guarantee that post-hoc circuit attribution cannot provide. An input intervention that targets the carrier can then be evaluated directly under fixed weights. Beyond analysis, a deployed model with a crafted sparse carrier could support inference-time behavioral auditing or selective suppression without any weight modification, enabling more modular post-training control.

These considerations motivate our research question:

Can SFT-induced behaviors be intentionally concentrated into sparse carriers that are sufficient for the behavior and, by construction, necessary for it, and that remain controllable at inference time without modifying model weights?

We study this question via two methods. We introduce **Loss-Constrained Dual Descent (LCDD)** to compress SFT behavior into a sparse carrier via learned masks on the weight delta, and **SFT-Eraser** to reverse the behaviors introduced by SFT via activation matching. SFT-Eraser treats a *reversal* as successful when it produces both behavioral suppression and distributional reversion, i.e., the output distribution of the triggered model moves back toward that of the base model. Our main contributions are as follows:

- To the best of our knowledge, we are the first to investigate the feasibility of concentrating SFT-induced behaviors into sparse substructures that are sufficient for the behavior and, by construction, necessary for it, enabling their control at inference time without modifying model weights.
- We propose **LCDD**, a framework for constructing sparse behavioral carriers by jointly optimizing routing masks and model weights under explicit utility constraints. To control the crafted carriers, we further introduce **SFT-Eraser**, an input-trigger[2] method that suppresses SFT-induced behaviors under fixed weights.
- We demonstrate that LCDD and SFT-Eraser enable inference-time reversibility across three behavior types and four model families. Our ablation study shows that triggered LCDD models are consistently closer to the base model than identically triggered standard SFT models, identifying the crafted structure as what drives reversion.

2 Related Work

Post-hoc analysis of SFT behaviors. Recent work characterizes SFT observationally. LIMA shows that a small high-quality SFT set suffices for strong assistant behavior [4], and the Superficial Safety Alignment Hypothesis frames alignment as a lightweight "fulfill vs. refuse" controller with sparse safety-critical components [9]. Depth-wise analyses find that alignment updates concentrate in specific layers [10, 11], and other work addresses catastrophic forgetting and objective mismatch [12, 13, 14].

[2]While we borrow the term *trigger* from the backdoor literature [23], our trigger steers the model back toward its behavior before SFT, whereas a backdoor trigger activates a specific implanted behavior.

SafeSeek frames circuit extraction as mask optimization and demonstrates highly sparse circuits for backdoor and alignment behaviors [18], and Depth Charge shows that targeted interventions on deep attention heads can substantially alter jailbreak success [24]. These methods identify or characterize structure post-hoc. By contrast, our work constructs sparse behavioral carriers during training, so that the carrier is necessary for the behavior by construction and the location of SFT-induced computation is known in advance.

Constructive and weight-space approaches. Task arithmetic defines task vectors as $\Delta W = W_{\text{ft}} - W_{\text{base}}$ and shows that adding or negating them can compose or suppress behaviors [21, 22]. However, it operates with dense global weight edits and does not produce a sparse activation-level structure addressable through input alone under fixed weights. Parameter-efficient adaptation via task-specific binary masks [25] is closer in spirit. Our mask-only ablation (Appendix E) shows that jointly adapting the weights reaches 2.4–2.9$\times$ the sparsity of masking alone. Most directly related is Gao et al. [19], which trains weight-sparse models from scratch on narrow tasks and finds that node-level sparsity yields highly interpretable circuits. Our work shares the same design principle and applies it to fine-tuning: we impose sparsity on the SFT weight delta ΔW in already-trained chat models to construct behavioral carriers. Broader differentiable mask and circuit discovery methods [15, 16, 17, 26, 27] provide methodological precedents but do not address SFT carrier construction.

3 Method

Our goal is to develop a *sparse behavioral carrier* satisfying two properties: (1) it maintains a minimal parameter substructure sufficient to sustain the SFT-induced behavior under normal inference; and (2) there exists a fixed-weight input intervention targeting the carrier that can reliably suppress the SFT-induced behavior. Property (2) relies on SFT-induced weight changes being confined to the carrier: by construction, model components outside the carrier are reduced to their base-model state. Sparsity makes the carrier compact and identifiable, so it can be precisely targeted at inference time. We pursue this in two stages as shown in Figure 1: crafting the sparse carrier via **LCDD** (Section 3.2), and reversing the SFT-induced behavior via **SFT-Eraser** (Section 3.3). Both stages build on the delta parameterization introduced in Section 3.1.

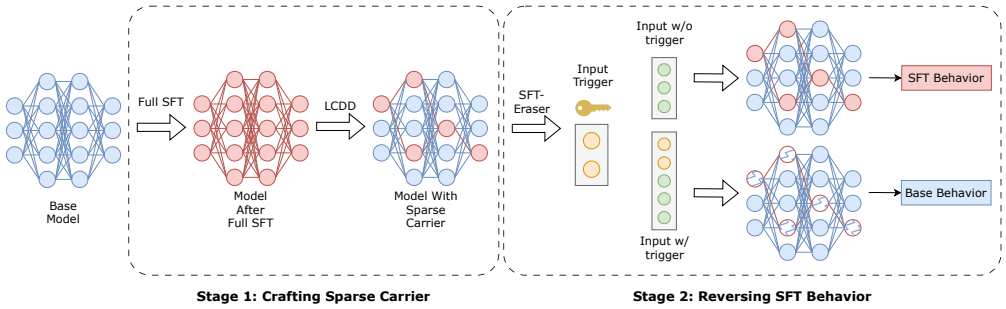

Figure 1: An overview of the LCDD + SFT-Eraser pipeline. **Stage 1**: standard SFT distributes the induced behavior broadly across model parameters (red), followed by LCDD that compresses the SFT-induced behavior into a sparse carrier. Components outside the carrier are reduced to their base-model state by construction (blue). **Stage 2**: SFT-Eraser optimizes a soft trigger. Under normal inference without the trigger, the carrier preserves SFT behavior, while with the trigger, carrier activations are driven toward the base model.

3.1 Preliminaries: Weight-Change Parameterization and Mask Implementation

Weight-change parameterization. We represent SFT-induced computation as the additive weight change from fine-tuning. Let W_{base} and W_{ft} denote the respective weights. We define the *weight delta* $\Delta W = W_{\text{ft}} - W_{\text{base}}$ as the SFT-induced parameter change. Our masked model is then parameterized as

$$W = W_{\text{base}} + M \odot \Delta W, \tag{1}$$

where $M \in \{0,1\}^{\text{shape}(W)}$ is a binary mask. Setting $M = \mathbf{0}$ recovers the base model exactly and $M = \mathbf{1}$ recovers the fully fine-tuned model. Our goal is to find the sparsest M that still preserves task behavior. Minimizing carrier size is deliberate: a sparser carrier is more identifiable and more precisely targetable at inference time, and in our utility-budget sweep sparser carriers revert more completely toward the base model (Appendix G).

Weight mask implementation. We structure sparsity at the level of activation channels (individual rows and columns of weight matrices), following the design principle that node-level weight sparsity produces identifiable and interpretable circuits in which information flow can be traced through designated channel bottlenecks [19]. Prior work has explored similar gradient-based mask optimization strategies for circuit discovery. Gao et al. [19] train weight-sparse models from scratch such that each neuron reads from and writes to only a small number of residual channels, making circuit boundaries well-defined by construction. Yu et al. [18] apply gradient-based binary mask optimization with the straight-through estimator (STE) to discover safety circuits in full models, gating unit outputs (neurons or attention heads) of the complete model weights W. Our setup differs in two respects. First, we apply structured masking specifically to the SFT weight change ΔW, isolating SFT-induced computation from the base model. Second, we use row and column gate vectors rather than unit-output masks, yielding a rank-1 structured mask that controls which activation channels carry the SFT-induced delta computation.

Formally, for each weight matrix, we introduce row and column gate vectors m_{row} and m_{col}. The contribution of the weight delta ΔW to a linear layer can be written as $[(\mathbf{x} \odot m_{\text{row}})\,\Delta W] \odot m_{\text{col}}$, where gating the input activations selects rows of ΔW and gating the output activations selects columns. This activation-gating view is equivalent to weight masking with the rank-1 structure

$$M_{jk} = m_{\text{row},j}\, m_{\text{col},k}. \tag{2}$$

Full derivations for both FFN and attention layers are given in Appendix A. Per transformer layer, we define 8 independent gate groups covering all delta-carrying weight matrices and the complete gate-to-weight-mask mapping is given in Table 5 of Appendix A.

3.2 LCDD: Utility-Budgeted Sparse Carrier Crafting

LCDD jointly optimizes the mask M and the weight change ΔW to compress SFT behavior into a sparse carrier while keeping utility loss within an explicit budget. We first describe how M is parameterized for optimization, and then present the main objective and the optimization procedure.

Parameterizing M. We cannot directly optimize M due to its discrete nature. Following Gao et al. [19], we parameterize each gate by a continuous logit $\theta_i \in \mathbb{R}$. During the forward pass, each gate is binarized via the Heaviside function as $m_i = \mathbf{1}[\theta_i > 0]$. Gradients are approximated by the straight-through estimator (STE). The sigmoid $\sigma(\theta_i)$ serves as a differentiable proxy for gate activation and defines the sparsity penalty:

$$\mathcal{L}_{\text{sparsity}} = \sum_i \sigma(\theta_i). \tag{3}$$

Here, i indexes all individual gate elements across all gate groups and all transformer layers. Minimizing $\mathcal{L}_{\text{sparsity}}$ progressively deactivates weights in the delta path. The task loss gradient via STE retains gates necessary for behavior. Full details are given in Appendix B.

Main objective. After parameterizing M via θ, the training objective is:

$$\min_{\theta,\,\Delta W} \sum_i \sigma(\theta_i) \qquad \text{s.t.} \qquad \mathcal{L}_{\text{task}}(\theta, \Delta W) \le \epsilon. \tag{4}$$

where $\mathcal{L}_{\text{task}}$ measures the model's task loss under the current mask and weight configuration, and ϵ is an explicit budget that caps the tolerated degradation in task performance. In our main experiments $\mathcal{L}_{\text{task}}$ is SFT cross-entropy. The constraint-based design provides a directly interpretable control surface: one specifies a concrete tolerance on utility degradation and lets the optimizer find the sparsest carrier within that budget.

Optimization. To solve Eq. (4), we use an adaptive penalty method: the constraint is absorbed into the objective with a multiplier λ_t updated online based on constraint violation. A linear sparsity warmup factor ρ_t gradually engages the penalty during early training, allowing the model to stabilize before compression begins. The multiplier grows when task loss is below budget (intensifying sparsification pressure) and shrinks when loss exceeds budget (allowing utility recovery). This loop continuously renegotiates the sparsity-utility trade-off without manual multiplier schedules or nested inner-loop minimization. Full details are given in Algorithm 1 (Appendix C).

To prevent behavior from routing through the token embedding or output projection layers, we freeze these components throughout training, confining compression to intermediate transformer layers. Training halts when carrier sparsity stalls across consecutive checkpoints or when utility degradation exceeds a threshold. We empirically verify that jointly optimizing masks and weight deltas is necessary for achieving high sparsity in Appendix E.

3.3 SFT-Eraser: Trigger-Based Reversal via Residual-Write Channel Matching

Given a sparse carrier crafted by LCDD, SFT-Eraser asks: can a fixed-weight input intervention suppress SFT behavior by targeting carrier activations directly? Because the carrier is the only location of SFT-induced weight changes, the trigger is optimized to bring the carrier's activations, and with them the output distribution, back to those of the base model.

Carrier channel extraction. After LCDD training, we identify active channels as those with $\theta_i > 0$. Among these, we select *residual write channels* as the trigger target:

$$\mathcal{C}_{\text{write}} = \mathcal{C}_{\text{ffn-write}} \cup \mathcal{C}_{\text{attn-write}}, \tag{5}$$

where $\mathcal{C}_{\text{ffn-write}}$ and $\mathcal{C}_{\text{attn-write}}$ are the sets of active output-gate channels of the FFN and attention delta paths, respectively (corresponding to $m_{\text{ffn-write}}$ and $m_{\text{attn-write}}$ in Section 3.1). We focus on write channels because they form the carrier's primary interface with the residual stream, where the delta computation contributes to downstream layer inputs.

Trigger objective. We optimize a soft trigger embedding sequence $t = [t_1, \ldots, t_L]$ prepended to the input prompt x, while all model weights remain frozen. The trigger loss combines three terms:

$$\mathcal{L}_{\text{trigger}} = \mathcal{L}_{\text{MSE}} + \alpha\, \mathcal{L}_{\text{KL}} + \beta\, \mathcal{L}_2. \tag{6}$$

Denoting by $\mathcal{A}_{\mathcal{C}}(\cdot)$ the residual-stream write activations at channel set $\mathcal{C}$, the primary term $\mathcal{L}_{\text{MSE}}$ drives the carrier's write activations toward those of the base model on the same input:

$$\mathcal{L}_{\text{MSE}} = \text{MSE}(\mathcal{A}_{\mathcal{C}_{\text{write}}}(\text{SFT}([t;\, x])),\ \mathcal{A}_{\mathcal{C}_{\text{write}}}(\text{Base}(x)))\,. \tag{7}$$

Since the trigger acts on the full input and can perturb activations beyond the carrier channels, $\mathcal{L}_{\text{MSE}}$ alone does not constrain the trigger's net effect on the output distribution. The auxiliary term $\mathcal{L}_{\text{KL}}$ adds output-level supervision, ensuring that the trigger produces distributional reversion toward the base model:

$$\mathcal{L}_{\text{KL}} = \text{KL}_{\text{tail-}k}(\text{Base}(x) \parallel \text{SFT}([t;\, x]))\,, \tag{8}$$

computed over the final k response positions to focus the signal on the behavioral regime. The regularizer $\mathcal{L}_2 = \frac{1}{L}\sum_{i=1}^{L} \|t_i\|_2^2$ prevents trigger tokens from drifting into degenerate high-magnitude embedding regions where activations become uninterpretable. Trigger optimization also applies PGD-style ℓ_2 projection after each gradient update (details in Appendix D).

4 Experiments

In this section, we conduct experiments to answer the following research questions:

- **RQ1:** Can LCDD craft sparse SFT carriers that preserve target behavior under normal inference?
- **RQ2:** Do LCDD-crafted carriers yield reliable trigger-based reversal under fixed weights?
- **RQ3:** Does the crafted sparse structure drive reversion toward the base model?

We describe the experimental setup in Section 4.1 and present results in Sections 4.2 and 4.3.

4.1 Experimental Setup

Tasks. We select three types of SFT tasks in increasing order of structural complexity:

- **Fixed Response.** The model is trained to respond "I don't know" on all instruction prompts from Alpaca [28]. This behavior has a discrete lexical signature and is input-unconditional, making it the simplest behavior to measure.
- **Safety Alignment.** The model is trained on WildJailbreak [29] harmful prompts interleaved with benign prompts (1:1 ratio). This behavior is input-conditional (triggered by semantic harm) and requires the carrier to encode content-sensitive routing.
- **Shakespeare Style.** The model is trained on modern-to-Shakespeare conversational pairs [30]. This is a distributional behavior with no discrete trigger condition, requiring the carrier to capture broad stylistic shifts across the output distribution, making it the most structurally demanding of the three tasks.

All main runs use 5,000 training samples per task.

Models. We evaluate on four chat LLM families: Qwen3-0.6B [31], DeepSeek-R1-Distill-Llama-8B [32], Mistral-7B-Instruct-v0.3 [33], and Vicuna-7B-v1.5 [34]. Model selection follows two criteria: architectural diversity across scales and pretraining regimes, and for the safety task, absence of built-in safety alignment so that refusal behavior is induced by SFT.

Metrics. We use the following metrics to evaluate how well the sparse carrier preserves SFT behavior before intervention (which we call *carrier fidelity*), and how completely it reverts after triggering.

- **Fixed Response:** Strict fixed-response rate, computed by keyword matching with a response length cap (Appendix H.4).
- **Safety:** WildGuard refusal rate [35] as the primary safety metric (higher is safer), WildGuard harmfulness rate [35] as a complementary output-side measure (lower is safer), and HarmBench ASR [36] as an attack-side measure (lower is safer). MMLU [37] and HellaSwag [38] track general utility.
- **Shakespeare:** LLM-judge score on a 0–5 Shakespearean authenticity scale, evaluated by Qwen3-14B [31] using the rubric in Appendix H.4. MMLU and HellaSwag track general utility.
- **KL metrics:** Computed by running each evaluated model on SFT-generated reference responses with teacher forcing (i.e., scoring on fixed reference tokens), averaged token-level over response tokens. KL(SFT‖LCDD) measures carrier fidelity before intervention (lower is better). KL(SFT‖Trig) and KL(Base‖Trig) jointly characterize distributional movement after triggering: larger KL(SFT‖Trig) indicates stronger divergence from SFT behavior, and smaller KL(Base‖Trig) indicates closer reversion to the base model.

Training setup. All runs follow the same two-phase pipeline (full SFT then LCDD), with the token embedding layer and output vocabulary projection frozen throughout LCDD training to confine compression to intermediate transformer layers. Trigger optimization targets residual write channels, uses trigger length 20, tail-k KL ($k = 8$), and norm-constrained gradient updates (ℓ_2 max-norm $= 1.0$) uniformly across all settings. Full hyperparameter details are in Appendix H.

4.2 Main Results

We present results across three behavior types. For each behavior type, we report carrier fidelity (how closely LCDD preserves SFT behavior before intervention) and reversal quality (how completely SFT-Eraser suppresses the behavior after triggering).

4.2.1 Fixed Response

As shown in Table 1, the Fixed Response task provides clear evidence that LCDD can craft a sparse carrier that is both behaviorally faithful before intervention and reliably reversible after

Table 1: **Fixed Response** results. "Trig" (LCDD+Trigger) denotes the LCDD model with SFT-Eraser trigger applied. KL values are computed on SFT-generated references.

		Fixed-Response Rate				KL Divergence		
Model	Sparsity	@Base	@SFT	@LCDD	@Trig	SFT‖LCDD	SFT‖Trig	Base‖Trig
Qwen3-0.6B	78.00%	0.0%	100.0%	97.5%	0.0%	0.040	2.839	0.633
DeepSeek-8B	73.38%	0.0%	100.0%	99.0%	1.0%	0.010	1.810	1.039
Mistral-7B	76.72%	0.0%	100.0%	100.0%	0.5%	0.002	3.527	1.437
Vicuna-7B	38.28%	0.0%	100.0%	100.0%	0.0%	0.389	3.068	0.396

triggering. LCDD preserves near-perfect fixed-response behavior across all four models (97.5–100%). KL(SFT‖LCDD) is low across all four models (0.002–0.389), confirming that the sparse carrier closely matches full SFT before intervention. Under trigger, fixed-response rate falls to near-zero in all cases (0.0–1.0%). The large post-trigger KL(SFT‖Trig) values (1.810–3.527) confirm strong divergence from SFT behavior, while KL(Base‖Trig) (0.396–1.437) quantifies distributional movement toward the base model.

4.2.2 Safety

Table 2: **Safety** results. "Trig" denotes LCDD+Trigger. All behavioral metric values in %. HB = HarmBench ASR (↓ safer); WG Ref. = WildGuard refusal rate (↑ safer); WG Harm. = WildGuard harmfulness rate (↓ safer). Sparsity is the LCDD mask sparsity. KL values are computed on SFT-generated references; the $P \,\|\, Q$ column indicates the two distributions compared.

			Safety Metrics			Utility Metrics		KL Divergence	
Model	Sparsity	Condition	HB ↓	WG Ref. ↑	WG Harm. ↓	MMLU	HellaSwag	$P \,\|\, Q$	Value
Qwen3-0.6B	74.48%	Base	43.0	32.0	52.0	40.3	37.5		
		SFT	1.5	94.0	2.0	46.0	38.7	SFT ‖ LCDD	0.069
		LCDD	2.5	88.5	3.5	42.2	38.1	SFT ‖ Trig	0.120
		Trig	32.5	37.0	40.5	45.1	37.9	Base ‖ Trig	0.196
DeepSeek-8B	81.38%	Base	19.0	47.5	38.5	54.1	55.6		
		SFT	0.0	100.0	0.0	55.3	57.1	SFT ‖ LCDD	0.254
		LCDD	1.5	96.5	2.0	54.8	56.3	SFT ‖ Trig	0.390
		Trig	13.0	56.5	30.5	49.2	54.1	Base ‖ Trig	0.305
Mistral-7B	59.34%	Base	58.5	25.5	59.5	59.8	65.8		
		SFT	0.0	100.0	0.0	50.1	53.9	SFT ‖ LCDD	0.266
		LCDD	1.0	97.5	1.0	57.0	63.6	SFT ‖ Trig	0.619
		Trig	17.0	69.0	21.5	54.1	62.3	Base ‖ Trig	1.282
Vicuna-7B	84.41%	Base	20.5	76.5	20.5	48.5	56.6		
		SFT	0.0	99.0	0.0	47.1	56.5	SFT ‖ LCDD	0.192
		LCDD	2.0	98.0	1.5	48.6	56.8	SFT ‖ Trig	1.085
		Trig	22.0	74.5	23.0	47.9	56.5	Base ‖ Trig	0.014

As shown in Table 2, all four models maintain high safety under LCDD (WG refusal 88.5–98.0%; HB ASR 1.0–2.5%), confirming that LCDD successfully compresses safety behavior into the sparse carrier. Under trigger, safety degrades along both axes for all models: refusal rates fall and attack success rates rise.

Reversion is most complete on Vicuna-7B, whose refusal rate under trigger (74.5%) returns to the base-model level (76.5%) with near-identical output distributions (KL(Base‖Trig) = 0.014). Qwen3-0.6B shows the largest behavioral suppression, with WG refusal falling by 51.5pp from 88.5% to 37.0%.

Utility is stable under LCDD for all models and shows only modest degradation under trigger (at most 5.6pp on MMLU for DeepSeek), indicating that the trigger suppresses the target behavior without broadly disrupting general language modeling.

4.2.3 Shakespeare Style

The Shakespeare task is the most demanding for LCDD: achieved sparsity is the lowest of the three tasks on every model (29.9–54.5%), and *carrier fidelity* (how closely the sparse carrier reproduces full SFT behavior before intervention, measured by KL(SFT‖LCDD) and the task-level Judge score) is correspondingly reduced: LCDD Judge scores (0.26–1.18) recover only part of the SFT-level style

Table 3: **Shakespeare style** results. "Trig" denotes LCDD+Trigger. Judge = LLM-judge score (0–5, ↑ more Shakespearean), evaluated by Qwen3-14B (rubric in Appendix H.4). MMLU and HellaSwag values in %. KL values computed on SFT-generated references; the $P \parallel Q$ column indicates the distributions compared. Sparsity is the LCDD mask sparsity.

			Style	Utility Metrics		KL Divergence	
Model	Sparsity	Condition	Judge ↑	MMLU	HellaSwag	$P \parallel Q$	Value
Qwen3-0.6B	41.20%	Base	0.03	40.3	37.5		
		SFT	0.79	45.5	38.0	SFT ‖ LCDD	0.265
		LCDD	0.44	41.8	38.6	SFT ‖ Trig	2.405
		Trig	0.04	42.7	37.6	Base ‖ Trig	0.446
DeepSeek-8B	54.54%	Base	0.01	54.1	55.6		
		SFT	1.73	54.5	56.1	SFT ‖ LCDD	1.520
		LCDD	0.98	55.4	56.8	SFT ‖ Trig	4.301
		Trig	0.005	37.4	51.5	Base ‖ Trig	0.531
Mistral-7B	50.46%	Base	0.07	59.8	65.8		
		SFT	1.75	47.9	55.0	SFT ‖ LCDD	0.853
		LCDD	1.18	55.4	62.7	SFT ‖ Trig	2.643
		Trig	0.07	42.3	53.5	Base ‖ Trig	2.052
Vicuna-7B	29.88%	Base	0.05	48.5	56.6		
		SFT	0.80	46.6	56.1	SFT ‖ LCDD	0.636
		LCDD	0.26	47.3	56.0	SFT ‖ Trig	1.702
		Trig	0.04	45.8	55.8	Base ‖ Trig	0.094

signal (0.79–1.75). This is consistent with the expectation that distributional stylistic behavior is more diffusely encoded across parameters than fixed-response or conditional refusal behaviors.

The trigger suppresses style in all four cases: Judge@Trig falls to near-base levels (0.005–0.07) regardless of the LCDD starting point. KL(SFT‖Trig) is substantially larger than KL(SFT‖LCDD) in all cases, confirming that post-trigger behavior diverges from SFT well beyond the LCDD baseline.

4.2.4 Cross-Task Observations

Three patterns emerge across the 12 model–task combinations:

- **Sparsity reflects task localizability.** On every model, the style behavior yields the lowest sparsity of the three tasks (30–55%, versus 38–84% for Fixed Response and safety). This reflects the structural demands of the behavior: distributional stylistic behavior requires broader representational support and resists aggressive compression.
- **Reversal is consistent across all combinations.** In all 12 combinations, the trigger moves the model away from SFT behavior and toward the base model.
- **Reversal does not require full carrier fidelity.** In the Shakespeare task, where carrier fidelity is lower than in Fixed Response or safety runs, the trigger still reverts the style to near-base levels in all four models.

Together, these results answer RQ1 and RQ2 affirmatively. LCDD successfully crafts sparse carriers that preserve SFT behavior under normal inference across all three behavior types and four model families. SFT-Eraser achieves reliable trigger-based reversal in all 12 model-task combinations.

4.3 Ablation Study

We investigate RQ3 by applying the same trigger optimization to LCDD models and to standard SFT models. We additionally verify the necessity of joint weight optimization in LCDD in Appendix E.

Setup. We cross two factors: model structure (LCDD vs. plain SFT) and trigger objective. The circuit-targeted objective is the full SFT-Eraser loss $\mathcal{L}_{\text{trigger}} = \mathcal{L}_{\text{MSE}} + \alpha\,\mathcal{L}_{\text{KL}} + \beta\,\mathcal{L}_2$, and the output-only objective removes $\mathcal{L}_{\text{MSE}}$, leaving $\alpha\,\mathcal{L}_{\text{KL}} + \beta\,\mathcal{L}_2$. When the circuit-targeted objective is applied to a plain SFT model, it matches the same channel set $\mathcal{C}_{\text{write}}$ extracted from the corresponding LCDD model. We run all four conditions on six model–task combinations covering all four model families and all three behavior types, and report KL(Base‖Trig), which measures how far the triggered model is from the base model (Section 4.1). Full results, including KL(SFT‖Trig), are given in Appendix F.

Table 4: Structural ablation: KL(Base‖Trig) (↓, closer to the base model) after applying the same trigger optimization to the LCDD model and to the plain SFT model. "Circuit" is the full SFT-Eraser objective; "Output" removes $\mathcal{L}_{\text{MSE}}$. Sparsity is that of the LCDD model.

			LCDD + trigger		Plain SFT + trigger	
Model	Task	Sparsity	Circuit	Output	Circuit	Output
DeepSeek-8B	Safety	81.38%	0.305	0.322	0.535	0.507
Vicuna-7B	Safety	84.41%	0.014	0.014	0.363	0.293
Qwen3-0.6B	Safety	74.48%	0.196	0.182	0.361	0.330
Mistral-7B	Safety	59.34%	1.282	1.247	1.424	1.304
DeepSeek-8B	Fixed Resp.	73.38%	1.039	0.794	4.151	2.452
DeepSeek-8B	Shakespeare	54.54%	0.531	0.604	1.092	1.094

Finding: the crafted structure drives reversion toward the base model. As shown in Table 4, in every one of the six combinations both LCDD conditions reach a lower KL(Base‖Trig) than both plain-SFT conditions. The triggered LCDD model is closer to the base model regardless of which trigger objective is used, across four model families and all three behavior types. Both models encode the same SFT behavior and differ in whether its weight changes are confined to a carrier. The crafted structure is therefore what brings triggered outputs back toward the base model.

5 Conclusion

We investigated whether SFT-induced behaviors can be deliberately compressed into sparse behavioral carriers, so that they are preserved under normal inference yet reversibly suppressible by an input trigger without weight modification. LCDD crafts such carriers through utility-budgeted joint optimization of routing masks and weight deltas, and SFT-Eraser reverts the SFT-induced behavior toward the base model via activation matching. The resulting carriers sustain the target behaviors and are, by construction, necessary for them. Our experiments and ablations establish that joint weight optimization is required to reach high carrier sparsity, and that the crafted structure is what drives reversion toward the base model.

Discussion. Beyond reversal itself, the sparse carrier framework suggests several longer-term directions. Crafted carriers provide a controlled testbed for interpretability research: unlike post-hoc circuit discovery, a carrier with known structure enables systematic study of intervention transferability and behavioral robustness. The constructability of a behavior may itself serve as a diagnostic property, since behaviors that resist LCDD compression are likely more diffusely encoded. This connects to a minimum description length interpretation [39], in which carrier sparsity is a computable proxy for the description complexity of the SFT behavioral transformation. Because LCDD operates on the weight delta $\Delta W = W_{\text{ft}} - W_{\text{base}}$ and does not depend on how that delta was produced, the same pipeline applies directly to checkpoints from preference-based or reinforcement-learning post-training such as DPO, PPO, or GRPO. Studying carriers systematically across post-training methods is a natural next step. Another direction is how pretraining scale affects the achievable sparsity–utility frontier for crafted carriers.

Limitations. Two limitations bound the current study. First, SFT-Eraser optimizes a soft prompt in continuous embedding space, and extending reversal to discrete hard-token triggers requires further investigation. Second, carrier fidelity depends on how compressible the target behavior is. Extending LCDD to more complex abilities, such as multi-step reasoning, is left to future work.

Broader impact. The same methodology could be used to weaken safety behavior in a sparsified model. This risk arises from sparsification itself: any pipeline that compresses a fine-tuned behavior into a small parameter subset also concentrates that behavior into a small attack surface. Our results also point to a concrete mitigation. Under identical trigger optimization, plain SFT models, whose weight changes are distributed across the network, remain further from their base models than LCDD models do (Section 4.3), and sparser carriers revert more completely (Appendix G). Keeping the fine-tuning delta distributed, or not releasing sparsified checkpoints, therefore avoids this exposure. All experiments use publicly released base models and public datasets (WildJailbreak, HarmBench,

and WildGuard), every model we trigger is one we fine-tuned ourselves, and no deployed system is involved. We release the training and analysis code needed to reproduce the method, but not optimized trigger embeddings. Applying a trigger requires white-box access to a sparsified checkpoint, and an attacker with such access already holds stronger capabilities.

Acknowledgments and Disclosure of Funding

Yuping Lin, Pengfei He, Yingqian Cui, and Hui Liu are supported by the National Science Foundation (NSF) under grant numbers IIS-2212032, IIS-2212144, IIS-2504089, DUE-2234015, CNS-2246050, and DRL-2405483, the Michigan Department of Agriculture and Rural Development, and the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, under Award Number DE-SC0026773. Yue Xing is supported by the National Science Foundation (NSF) under grant number DMS-2515194, the National Institutes of Health (NIH) under grant number R21AI203026, Coefficient Giving, and Tutu AI Ventures LLC.

References

[1] 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.

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

[3] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

[4] Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. Lima: Less is more for alignment. *Advances in Neural Information Processing Systems*, 36:55006–55021, 2023.

[5] Nick Mecklenburg, Yiyou Lin, Xiaoxiao Li, Daniel Holstein, Leonardo Nunes, Sara Malvar, Bruno Silva, Ranveer Chandra, Vijay Aski, Pavan Kumar Reddy Yannam, et al. Injecting new knowledge into large language models via supervised fine-tuning. *arXiv preprint arXiv:2404.00213*, 2024.

[6] Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, et al. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 1(1):12, 2021.

[7] Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 5(3):e00024–001, 2020.

[8] Lawrence Chan, Adrià Garriga-Alonso, Nicholas Goldowsky-Dill, Ryan Greenblatt, Jenny Nitishinskaya, Ansh Radhakrishnan, Buck Shlegeris, and Nate Thomas. Causal scrubbing: A method for rigorously testing interpretability hypotheses. `https://www.alignmentforum.org/posts/JvZhhzycHu2Yd57RN/causal-scrubbing-a-method-for-rigorously-testing`, Dec 2022. AI Alignment Forum / Redwood Research.

[9] Jianwei Li and Jung-Eun Kim. Superficial safety alignment hypothesis. *arXiv preprint arXiv:2410.10862*, 2024.

[10] Qinghua Zhao, Xueling Gong, Xinyu Chen, Zhongfeng Kang, and Xinlu Li. A layer-wise analysis of supervised fine-tuning. *arXiv preprint arXiv:2604.11838*, 2026.

[11] Yang Zhao, Li Du, Xiao Ding, Kai Xiong, Ting Liu, and Bing Qin. Supervised fine-tuning achieve rapid task adaption via alternating attention head activation patterns. *arXiv preprint arXiv:2409.15820*, 2024.

[12] Fei Ding and Baiqiao Wang. Improved supervised fine-tuning for large language models to mitigate catastrophic forgetting. *arXiv preprint arXiv:2506.09428*, 2025.

[13] Yutao Sun, Mingshuai Chen, Tiancheng Zhao, Phillip Miao, Zilun Zhang, Haozhan Shen, Ruizhe Zhu, and Jianwei Yin. Talking to yourself: Defying forgetting in large language models. *arXiv preprint arXiv:2602.20162*, 2026.

[14] Zhichao Wang, Bin Bi, Zixu Zhu, Xiangbo Mao, Jun Wang, and Shiyu Wang. Uft: Unifying fine-tuning of sft and rlhf/dpo/una through a generalized implicit reward function. *arXiv preprint arXiv:2410.21438*, 2024.

[15] Arthur Conmy, Augustine Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. Towards automated circuit discovery for mechanistic interpretability. *Advances in Neural Information Processing Systems*, 36:16318–16352, 2023.

[16] Aaquib Syed, Can Rager, and Arthur Conmy. Attribution patching outperforms automated circuit discovery. In *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 407–416, 2024.

[17] Samuel Marks, Can Rager, Eric J Michaud, Yonatan Belinkov, David Bau, and Aaron Mueller. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models. *arXiv preprint arXiv:2403.19647*, 2024.

[18] Miao Yu, Siyuan Fu, Moayad Aloqaily, Zhenhong Zhou, Safa Otoum, Kun Wang, Yufei Guo, Qingsong Wen, et al. Safeseek: Universal attribution of safety circuits in language models. *arXiv preprint arXiv:2603.23268*, 2026.

[19] Leo Gao, Achyuta Rajaram, Jacob Coxon, Soham V Govande, Bowen Baker, and Dan Mossing. Weight-sparse transformers have interpretable circuits. *arXiv preprint arXiv:2511.13653*, 2025.

[20] Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, et al. Toy models of superposition. *arXiv preprint arXiv:2209.10652*, 2022.

[21] Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*, 2022.

[22] Guillermo Ortiz-Jimenez, Alessandro Favero, and Pascal Frossard. Task arithmetic in the tangent space: Improved editing of pre-trained models. *Advances in Neural Information Processing Systems*, 36:66727–66754, 2023.

[23] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Evaluating backdooring attacks on deep neural networks. *Ieee Access*, 7:47230–47244, 2019.

[24] Jinman Wu, Yi Xie, Shiqian Zhao, and Xiaofeng Chen. Depth charge: Jailbreak large language models from deep safety attention heads. *arXiv preprint arXiv:2603.05772*, 2026.

[25] Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *Proceedings of the European conference on computer vision (ECCV)*, pages 67–82, 2018.

[26] Christos Louizos, Max Welling, and Diederik P Kingma. Learning sparse neural networks through l_0 regularization. *arXiv preprint arXiv:1712.01312*, 2017.

[27] Steven Cao, Victor Sanh, and Alexander M Rush. Low-complexity probing via finding subnetworks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 960–966, 2021.

[28] 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. `https://github.com/tatsu-lab/stanford_alpaca`, 2023.

[29] Liwei Jiang, Kavel Rao, Seungju Han, Allyson Ettinger, Faeze Brahman, Sachin Kumar, Niloofar Mireshghallah, Ximing Lu, Maarten Sap, Yejin Choi, et al. Wildteaming at scale: From in-the-wild jailbreaks to (adversarially) safer language models. *Advances in Neural Information Processing Systems*, 37:47094–47165, 2024.

[30] Roudranil. Shakespearean and modern english conversational dataset. `https://huggingface.co/datasets/Roudranil/shakespearean-and-modern-english-conversational-dataset`, 2023.

[31] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[32] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[33] Albert Q Jiang, A Sablayrolles, A Mensch, C Bamford, D Singh Chaplot, Ddl Casas, F Bressand, G Lengyel, G Lample, L Saulnier, et al. Mistral 7b. arxiv. *arXiv preprint arXiv:2310.06825*, 10: 3, 2023.

[34] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623, 2023.

[35] Seungju Han, Kavel Rao, Allyson Ettinger, Liwei Jiang, Bill Yuchen Lin, Nathan Lambert, Yejin Choi, and Nouha Dziri. Wildguard: Open one-stop moderation tools for safety risks, jailbreaks, and refusals of llms. *Advances in neural information processing systems*, 37:8093–8131, 2024.

[36] Mantas Mazeika, Long Phan, Xuwang Yin, Andy Zou, Zifan Wang, Norman Mu, Elham Sakhaee, Nathaniel Li, Steven Basart, Bo Li, et al. Harmbench: A standardized evaluation framework for automated red teaming and robust refusal. *arXiv preprint arXiv:2402.04249*, 2024.

[37] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

[38] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 4791–4800, 2019.

[39] Jorma Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465–471, 1978.

[40] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

A Gate–Weight Equivalence: Full Derivations

We prove that the activation-gating formulation used in LCDD is equivalent to the structured weight masking in Section 3.1. Throughout, $\Delta W = W_{\text{ft}} - W_{\text{base}}$, $M \in \{0,1\}^{\text{shape}(W)}$ is a binary mask, and $\phi(\cdot)$ denotes the layer nonlinearity.

A.1 FFN Layer

Let $\mathbf{x} \in \mathbb{R}^d$ be the layer input, $W_{\text{up}}^{\text{base}} \in \mathbb{R}^{d \times d_{\text{ffn}}}$ and $W_{\text{down}}^{\text{base}} \in \mathbb{R}^{d_{\text{ffn}} \times d}$ the base FFN weights, and $\Delta W_{\text{up}}, \Delta W_{\text{down}}$ the corresponding fine-tuning increments. Three binary gate vectors control the delta path:

$\mathbf{m}_{\text{read}} \in \{0,1\}^d$	gates input dimensions (residual read)
$\mathbf{m}_{\text{hidden}} \in \{0,1\}^{d_{\text{ffn}}}$	gates hidden (intermediate) dimensions
$\mathbf{m}_{\text{write}} \in \{0,1\}^d$	gates output dimensions (residual write)

Activation-gating forward pass.

$$\mathbf{h} = \phi\big(\mathbf{x}\, W_{\text{up}}^{\text{base}} + \big(\mathbf{x} \odot \mathbf{m}_{\text{read}}\big)\Delta W_{\text{up}} \odot \mathbf{m}_{\text{hidden}}\big) \tag{9}$$

$$\Delta\mathbf{y} = \big(\mathbf{h} \odot \mathbf{m}_{\text{hidden}}\big)\Delta W_{\text{down}} \odot \mathbf{m}_{\text{write}} \tag{10}$$

Equivalent weight masks.

$$M_{\text{up}}[i,j] = m_{\text{read}}[i] \cdot m_{\text{hidden}}[j], \qquad M_{\text{down}}[i,j] = m_{\text{hidden}}[i] \cdot m_{\text{write}}[j]. \tag{11}$$

Proof for W_{up}. The j-th pre-activation under activation gating is

$$\begin{aligned}\text{pre}_j &= \sum_i x_i\, W_{\text{up}}^{\text{base}}[i,j] + \Big(\sum_i (x_i \cdot m_{\text{read}}[i])\, \Delta W_{\text{up}}[i,j]\Big)\, m_{\text{hidden}}[j] \\ &= \sum_i x_i\, W_{\text{up}}^{\text{base}}[i,j] + \sum_i x_i \cdot m_{\text{read}}[i] \cdot m_{\text{hidden}}[j] \cdot \Delta W_{\text{up}}[i,j].\end{aligned} \tag{12}$$

Under weight masking with $M_{\text{up}}[i,j] = m_{\text{read}}[i] \cdot m_{\text{hidden}}[j]$:

$$\begin{aligned}\text{pre}_j &= \sum_i x_i \Big(W_{\text{up}}^{\text{base}}[i,j] + M_{\text{up}}[i,j] \cdot \Delta W_{\text{up}}[i,j]\Big) \\ &= \sum_i x_i\, W_{\text{up}}^{\text{base}}[i,j] + \sum_i x_i \cdot m_{\text{read}}[i] \cdot m_{\text{hidden}}[j] \cdot \Delta W_{\text{up}}[i,j]. \quad \checkmark\end{aligned} \tag{13}$$

□

Proof for W_{down}. The j-th output delta under activation gating is

$$\Delta y_j = \Big(\sum_i h_i \cdot m_{\text{hidden}}[i] \cdot \Delta W_{\text{down}}[i,j]\Big) m_{\text{write}}[j] = \sum_i h_i \cdot m_{\text{hidden}}[i] \cdot m_{\text{write}}[j] \cdot \Delta W_{\text{down}}[i,j]. \tag{14}$$

Under weight masking with $M_{\text{down}}[i,j] = m_{\text{hidden}}[i] \cdot m_{\text{write}}[j]$:

$$\Delta y_j = \sum_i h_i \cdot M_{\text{down}}[i,j] \cdot \Delta W_{\text{down}}[i,j] = \sum_i h_i \cdot m_{\text{hidden}}[i] \cdot m_{\text{write}}[j] \cdot \Delta W_{\text{down}}[i,j]. \quad \checkmark \qquad \square \tag{15}$$

□

A.2 Attention Layer

Let $\mathbf{x} \in \mathbb{R}^d$ be the layer input, $W_Q^{\text{base}}, W_K^{\text{base}}, W_V^{\text{base}} \in \mathbb{R}^{d \times d_{\text{inner}}}$ the base projection weights, and $W_O^{\text{base}} \in \mathbb{R}^{d_{\text{inner}} \times d}$ the output projection weight. Five binary gate vectors control the delta path (Figure 2):

$\mathbf{m}_{\text{read}} \in \{0,1\}^d$	gates input dimensions
$\mathbf{m}_q, \mathbf{m}_k, \mathbf{m}_v \in \{0,1\}^{d_{\text{inner}}}$	gate Q/K/V head dimensions independently
$\mathbf{m}_{\text{write}} \in \{0,1\}^d$	gates output dimensions

Activation-gating forward pass.

$$\mathbf{q} = \mathbf{x}\, W_Q^{\text{base}} + (\mathbf{x} \odot \mathbf{m}_{\text{read}})\, \Delta W_Q \odot \mathbf{m}_q \tag{16}$$

$$\mathbf{k} = \mathbf{x}\, W_K^{\text{base}} + (\mathbf{x} \odot \mathbf{m}_{\text{read}})\, \Delta W_K \odot \mathbf{m}_k \tag{17}$$

$$\mathbf{v} = \mathbf{x}\, W_V^{\text{base}} + (\mathbf{x} \odot \mathbf{m}_{\text{read}})\, \Delta W_V \odot \mathbf{m}_v \tag{18}$$

$$\mathbf{c} = \text{Attention}(\mathbf{q}, \mathbf{k}, \mathbf{v}) \tag{19}$$

$$\Delta\mathbf{y} = (\mathbf{c} \odot \mathbf{m}_v)\, \Delta W_O \odot \mathbf{m}_{\text{write}} \tag{20}$$

The row gate for ΔW_O in Eq. (20) is $\mathbf{m}_v$, not $\mathbf{m}_{\text{read}}$. This follows from data flow: W_O reads from the context vector $\mathbf{c}$, which is assembled from value projections. Zeroing value channel i (i.e. $m_v[i] = 0$) should suppress row i of ΔW_O; using $\mathbf{m}_{\text{read}}$ here would be structurally inconsistent.

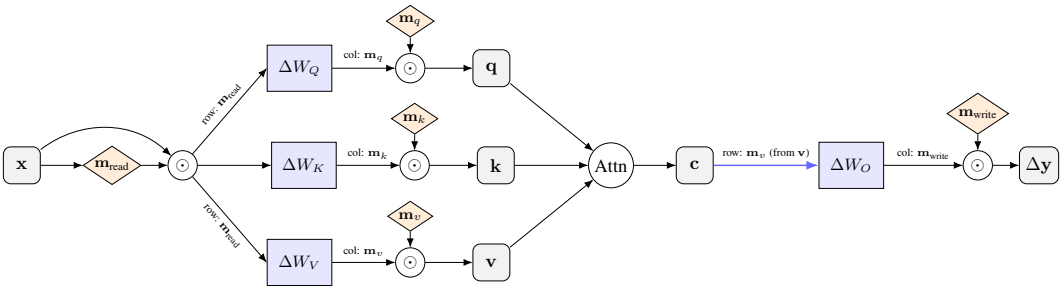

Figure 2: Delta-path computation graph for the attention layer with activation gating. Input $\mathbf{x}$ is gated by $\mathbf{m}_{\text{read}}$ before entering all projection deltas; Q/K/V outputs are independently gated by $\mathbf{m}_q, \mathbf{m}_k, \mathbf{m}_v$. The row mask of ΔW_O is $\mathbf{m}_v$ (not $\mathbf{m}_{\text{read}}$) because W_O reads from the context vector $\mathbf{c}$, which is derived from value projections.

Equivalent weight masks.

$$\begin{aligned} M_Q[i,j] &= m_{\text{read}}[i] \cdot m_q[j], \quad & M_K[i,j] &= m_{\text{read}}[i] \cdot m_k[j], \\ M_V[i,j] &= m_{\text{read}}[i] \cdot m_v[j], \quad & M_O[i,j] &= m_v[i] \cdot m_{\text{write}}[j]. \end{aligned} \tag{21}$$

Proof for W_Q, W_K, W_V. The arguments are identical; we take W_Q as representative. The j-th component of the delta query under activation gating:

$$\Delta q_j = \Big(\sum_i (x_i \cdot m_{\text{read}}[i]) \cdot \Delta W_Q[i,j]\Big) m_q[j] = \sum_i x_i \cdot m_{\text{read}}[i] \cdot m_q[j] \cdot \Delta W_Q[i,j]. \tag{22}$$

Under weight masking with $M_Q[i,j] = m_{\text{read}}[i] \cdot m_q[j]$:

$$\Delta q_j = \sum_i x_i \cdot M_Q[i,j] \cdot \Delta W_Q[i,j] = \sum_i x_i \cdot m_{\text{read}}[i] \cdot m_q[j] \cdot \Delta W_Q[i,j]. \quad \checkmark \tag{23}$$

□

Proof for W_O. Let $\mathbf{c} \in \mathbb{R}^{d_{\text{inner}}}$ be the context vector. The j-th output delta under activation gating:

$$\Delta y_j = \Big(\sum_k c_k \cdot m_v[k] \cdot \Delta W_O[k,j]\Big) m_{\text{write}}[j] = \sum_k c_k \cdot m_v[k] \cdot m_{\text{write}}[j] \cdot \Delta W_O[k,j]. \tag{24}$$

Under weight masking with $M_O[k,j] = m_v[k] \cdot m_{\text{write}}[j]$:

$$\Delta y_j = \sum_k c_k \cdot M_O[k,j] \cdot \Delta W_O[k,j] = \sum_k c_k \cdot m_v[k] \cdot m_{\text{write}}[j] \cdot \Delta W_O[k,j]. \quad \checkmark \quad \square \tag{25}$$

□

A.3 Summary

Table 5 collects the row and column gate assignments for all six delta weight matrices per transformer layer. Two structural observations follow directly from the assignment. First, $\mathbf{m}_{\text{read}}$ and $\mathbf{m}_{\text{write}}$ act as global input/output interfaces: any gate-group optimisation that suppresses $\mathbf{m}_{\text{read}}$ simultaneously prunes the row dimension of all four attention projection deltas and the row of ΔW_{up}. Second, $\mathbf{m}_v$ is shared between W_V (column) and W_O (row), enforcing structural consistency: a value channel that is pruned in the projection stage is also pruned in the readout stage, preventing information leakage through the output projection.

B Gate Learning: Heaviside Binarization and STE

Each gate logit $\theta_i \in \mathbb{R}$ is binarized during the forward pass via the Heaviside function:

$$m_i = \mathbf{1}[\theta_i > 0] \ \in \ \{0, 1\}. \tag{26}$$

Table 5: Gate-to-weight-mask assignment per transformer layer. The 8 gate vectors total 3 (FFN) + 5 (Attention). d = residual-stream dimension; d_{ffn} = FFN hidden dimension; d_{inner} = attention head dimension.

Module	Weight matrix	Shape	Row gate	Col gate
FFN	W_{up}	$d \times d_{\text{ffn}}$	$\mathbf{m}_{\text{read}}$	$\mathbf{m}_{\text{hidden}}$
	W_{down}	$d_{\text{ffn}} \times d$	$\mathbf{m}_{\text{hidden}}$	$\mathbf{m}_{\text{write}}$
Attention	W_Q	$d \times d_{\text{inner}}$	$\mathbf{m}_{\text{read}}$	$\mathbf{m}_q$
	W_K	$d \times d_{\text{inner}}$	$\mathbf{m}_{\text{read}}$	$\mathbf{m}_k$
	W_V	$d \times d_{\text{inner}}$	$\mathbf{m}_{\text{read}}$	$\mathbf{m}_v$
	W_O	$d_{\text{inner}} \times d$	$\mathbf{m}_v$	$\mathbf{m}_{\text{write}}$

Since $\mathbf{1}[\cdot]$ has zero gradient almost everywhere, backpropagation is infeasible directly. We apply the straight-through estimator (STE) [40]: during the backward pass, the gradient with respect to θ_i is approximated as

$$\frac{\partial \mathcal{L}}{\partial \theta_i} \approx \frac{\partial \mathcal{L}}{\partial m_i} \cdot \sigma'(\theta_i), \qquad \sigma'(\theta_i) = \sigma(\theta_i)\big(1 - \sigma(\theta_i)\big), \tag{27}$$

while the forward pass retains the exact binary value. We use Heaviside + STE rather than HardConcrete-based L_0 relaxation [26], which introduces a training–inference inconsistency via continuous masks; [19] report that Heaviside-based approaches consistently outperform the HardConcrete variant in sparse training settings.

C LCDD Controller: Full Algorithm

We give here the complete pseudocode for the LCDD adaptive dual-inspired controller described in Section 3.2. The algorithm instantiates the adaptive penalty method for the constrained objective (Equation 4), combining a linear sparsity warmup, an exponential moving average (EMA)-based budget threshold, and a multiplicative multiplier update driven by the normalized constraint violation ratio.

D Trigger Optimization Details

After each gradient update on $\mathcal{L}_{\text{trigger}}$, we apply PGD-style ℓ_2 projection to keep each trigger token within a bounded embedding region:

$$t_i \leftarrow \Pi_{\|\cdot\|_2 \le R}(t_i - \eta_t \nabla_{t_i} \mathcal{L}_{\text{trigger}}) \,. \tag{28}$$

This prevents the optimizer from exploiting unbounded embedding directions to satisfy the activation matching objective trivially. Full trigger hyperparameters are listed in Appendix H.3.

E Ablation Study: Necessity of Joint Weight Optimization

We test whether jointly optimizing masks and weight deltas is strictly necessary. As an alternative, we fix W_{ft} and optimize only the gate parameters θ. This follows the fixed-weight masking approach used in mask-based model adaptation [25] and safety circuit attribution [18], applied here to the SFT delta path. We evaluate whether this simpler setup achieves comparable sparsity, and whether the resulting carriers remain reversible.

Setup. We construct two mask-only variants under the same LCDD controller, budget ratio (0.30), and early-stop criterion as the main experiment. Two loss variants are tested: (1) CE-driven mask-only and (2) KL-distillation mask-only. The same trigger optimization is then applied to each checkpoint. Both variants are evaluated on DeepSeek-R1-Distill-Llama-8B under the safety task.

Finding 1: mask-only cannot achieve comparable sparsity. Both mask-only variants trigger early stopping by epoch ≈3.8. They reach only 28–34% sparsity. This is 2.4–2.9× lower than the 81.38% achieved by joint optimization. Without weight adaptation, mask deletions directly degrade

Algorithm 1 LCDD: Utility-Budgeted Sparse Carrier Crafting

Require: Base model W_{base}, fine-tuned model W_{ft}, budget ratio r, warmup steps T_w, multiplier bounds $\lambda_{\min}, \lambda_{\max}$, multiplier step size η_λ, EMA decay β
Ensure: Sparse mask M^*, adapted weight increment ΔW^*

Initialize $\Delta W \leftarrow W_{\text{ft}} - W_{\text{base}}$, gate logits $\theta_i \leftarrow 0\ \forall i$, multiplier λ_0, EMA estimate $\widehat{\mathcal{L}}_0 \leftarrow \mathcal{L}_{\text{task}}$
for each training step $t = 1, 2, \ldots$ **do**
  // Sparsity warmup
  $\rho_t \leftarrow \min(1,\ t/T_w)$
  // Forward pass with binary gates
  $m_i \leftarrow \mathbf{1}[\theta_i > 0]\ \forall i$
  Compute $\mathcal{L}_{\text{task}}$ and $\mathcal{L}_{\text{sparsity}} = \sum_i \sigma(\theta_i)$
  // Combined loss
  $\mathcal{L} \leftarrow \mathcal{L}_{\text{task}} + \lambda_t\, \rho_t\, \mathcal{L}_{\text{sparsity}}$
  // Update θ via STE, update ΔW via standard grad
  $\theta,\ \Delta W \leftarrow \textsc{BackwardStep}(\mathcal{L})$
  // Update EMA of task loss
  $\widehat{\mathcal{L}}_t \leftarrow \beta\, \widehat{\mathcal{L}}_{t-1} + (1-\beta)\, \mathcal{L}_{\text{task}}$
  if $t = T_w$ **then** ▷ Set budget after warmup stabilizes
    $\epsilon \leftarrow \widehat{\mathcal{L}}_{T_w} \cdot (1 + r)$
  end if
  // Update multiplier via normalized constraint violation ratio
  $v_t \leftarrow (\widehat{\mathcal{L}}_t - \epsilon)\ /\ \epsilon$
  $\lambda_{t+1} \leftarrow \text{clip}(\lambda_t \exp(-\eta_\lambda\, v_t),\ \lambda_{\min},\ \lambda_{\max})$
  if early-stop criterion met **then** ▷ Sparsity stall or budget breach
    break
  end if
end for
return $M^* = \{m_i\},\quad \Delta W^*$

Table 6: Ablation: sparsity and trigger-based reversal on DeepSeek-R1-Distill-Llama-8B (safety task). KL metrics computed on SFT-generated references. Base WG refusal: 47.5%; SFT WG refusal: 100.0%.

			WG Refusal		KL Divergence		
Method	Stop Epoch	Sparsity	LCDD	+Trig	SFT‖LCDD	SFT‖Trig	Base‖Trig
LCDD joint (main)	—	81.38%	96.5%	56.5%	0.254	0.390	0.305
Mask-only (CE)	3.70	28.40%	99.0%	64.0%	0.121	0.262	0.254
Mask-only (KL)	3.83	33.71%	98.0%	60.0%	0.169	0.318	0.196

safety behavior. The optimizer hits a hard constraint and stops within the first four epochs. Joint optimization avoids this. Weight deltas reorganize around pruned connections, keeping behavior within the utility budget.

Finding 2: joint optimization compresses the carrier while keeping it reversible. All three carriers remain reversible under the same trigger optimization: WG refusal falls from 96.5% to 56.5% for the joint carrier, and from 99.0% to 64.0% (CE) and from 98.0% to 60.0% (KL) for the mask-only carriers. Joint optimization therefore yields a carrier 2.4–2.9× sparser than masking alone while retaining reversal of comparable magnitude.

Synthesis. Joint weight optimization is necessary to reach the sparsity regime of LCDD. Masking the fixed SFT delta stalls at 28–34% sparsity, whereas jointly adapting the weights lets the delta reorganize around pruned connections and reach 81.38% sparsity, with the behavior kept within the utility budget and the resulting carrier reversible.

F Structural Ablation: Full Results

Table 7 extends Table 4 with KL(SFT‖Trig), which measures how far the trigger moves each model away from its own fine-tuned behavior. In all six combinations and under both trigger objectives, the trigger moves the LCDD model further from SFT behavior than the plain SFT model, and brings it closer to the base model. When the circuit-targeted objective is applied to a plain SFT model, it matches the channel set $\mathcal{C}_{\text{write}}$ extracted from the LCDD model of the same model–task combination. KL values are computed with SFT-generated references as in Appendix H.4.

Table 7: Full structural ablation results. "Circuit" is the full SFT-Eraser objective; "Output" removes $\mathcal{L}_{\text{MSE}}$.

		KL(Base‖Trig) ↓				KL(SFT‖Trig) ↑			
		LCDD		Plain SFT		LCDD		Plain SFT	
Model	Task	Circuit	Output	Circuit	Output	Circuit	Output	Circuit	Output
DeepSeek-8B	Safety	0.305	0.322	0.535	0.507	0.390	0.438	0.222	0.209
Vicuna-7B	Safety	0.014	0.014	0.363	0.293	1.085	1.062	0.403	0.409
Qwen3-0.6B	Safety	0.196	0.182	0.361	0.330	0.120	0.133	0.078	0.070
Mistral-7B	Safety	1.282	1.247	1.424	1.304	0.619	0.619	0.416	0.467
DeepSeek-8B	Fixed Resp.	1.039	0.794	4.151	2.452	1.810	2.116	0.736	1.017
DeepSeek-8B	Shakespeare	0.531	0.604	1.092	1.094	4.301	4.171	3.486	3.294

G Sensitivity Analysis

We vary one setting at a time on Qwen3-0.6B (safety), holding all others at the values of the main experiment. KL is computed as in Appendix H.4.

Utility budget. The budget ratio r sets $\epsilon = \widehat{\mathcal{L}}\,(1 + r)$ in Algorithm 1, where $\widehat{\mathcal{L}}$ is the EMA of the task loss at the end of warmup. As shown in Table 8, a larger budget yields a sparser carrier and a more complete reversion: from $r = 0.0005$ to $r = 0.10$, sparsity rises from 72.03% to 90.61% and KL(Base‖Trig) falls from 0.195 to 0.079, while KL(SFT‖LCDD) rises from 0.063 to 0.212. The budget is thus a single interpretable control that trades carrier fidelity for completeness of reversion.

Trigger length and channel selection. Reversion is stable in both settings (Table 9). Across an eight-fold change in trigger length (5 to 40 tokens), KL(Base‖Trig) stays within 0.193–0.198. Matching all residual channels, only the FFN write channels, or only the attention write channels gives KL(Base‖Trig) within 0.192–0.203, matching the default residual write channels (0.196).

Table 8: Utility budget sweep on Qwen3-0.6B (safety). $r = 0.01$ is the main-experiment setting.

Budget ratio r	Sparsity	KL(SFT‖LCDD) ↓	KL(Base‖Trig) ↓
0.0005	72.03%	0.063	0.195
0.003	72.72%	0.063	0.201
0.01	74.48%	0.069	0.196
0.04	81.74%	0.107	0.151
0.10	90.61%	0.212	0.079

H Experimental Details

H.1 Dataset Details

Fixed Response task (training and evaluation prompts). For Fixed Response training, we use the Alpaca dataset [28] and format each sample as a chat prompt with a fixed target completion "I don't know." Main runs use the first 5,000 examples from the Alpaca `train` split (without shuffling); samples with a non-empty secondary input field have that field appended to the instruction. Evaluation prompts are drawn from a held-out pool by excluding those first 5,000 examples, shuffling

Table 9: Trigger length and matched channel set on Qwen3-0.6B (safety), using the main-experiment carrier (74.48% sparsity). Defaults are length 20 and residual write channels.

Setting	KL(SFT‖Trig)	KL(Base‖Trig) ↓
Length 5	0.136	0.193
Length 10	0.131	0.198
Length 20 (default)	0.120	0.196
Length 40	0.130	0.196
Residual write (default)	0.120	0.196
All residual (read + write)	0.135	0.192
FFN write only	0.120	0.203
Attention write only	0.120	0.203

the remainder with a fixed seed, and retaining only instruction-only rows (empty secondary input field).

Safety task. We use the WildJailbreak dataset [29] and separate harmful and benign subsets. Main safety training uses a 1:1 harmful/benign mixture (5,000 total), matching the intended alignment objective of retaining utility while learning refusal behavior. Safety evaluation uses HarmBench standard behaviors and reports HarmBench ASR plus WildGuard refusal and harmfulness rates.

Shakespeare task. We use the Shakespearean and Modern English Conversational Dataset [30], treating the modern English translation as input and the original Shakespearean text as target. The dataset provides paired modern / Shakespearean dialogue turns at the utterance level. Main runs use 5,000 training samples.

H.2 Training Hyperparameters

Shared settings. All LCDD runs follow a two-phase pipeline: the SFT checkpoint provides the fine-tuned weights W_{ft}, and the original pretrained model provides the base weights W_{base}. Mask parameters are optimized with SGD (momentum 0.0) at learning rate 0.1. The SFT stage uses learning rate 2×10^{-5} across all main runs. Training sample count is 5,000 per task.

Table 10 lists the LCDD hyperparameters for the 12 primary model-task runs. The column $\eta_{\lambda,\uparrow}$ denotes a separate growth learning rate used in asymmetric dual controller variants; "–" indicates the standard symmetric update is used instead.

Table 10: LCDD hyperparameters for the 12 primary model-task runs. η_λ: multiplier update step size. $\eta_{\lambda,\uparrow}$: asymmetric growth rate ("–" = symmetric update).

Setting	budget ratio	λ_0	η_λ	$\eta_{\lambda,\uparrow}$	warmup steps	epochs
Fixed Response (Qwen, Mistral, DeepSeek)	0.30	1.0	0.1	–	300	10
Fixed Response (Vicuna)	0.30	0.01	0.1	0.01	700	20
Safety (Mistral, DeepSeek)	0.30	0.01	0.1	–	300	20
Safety (Qwen)	0.01	0.01	0.1	0.01	300	20
Safety (Vicuna)	0.50	0.01	1.0	0.005	300	20
Shakespeare (Qwen, Mistral, Vicuna)	0.01	0.01	0.1	0.01	700	20
Shakespeare (DeepSeek)	0.01	0.01	0.1	0.01	300	20

Additional stabilizers. For runs with asymmetric dual control, $\lambda_{\max} \in \{0.02, 0.05\}$. For Qwen and Vicuna safety and Shakespeare runs, we apply EMA smoothing with $\beta_{\text{ema}} = 0.90$ to the task loss estimate.

Compute resources. Each experiment run (one model-task combination) is conducted on a single NVIDIA H200 GPU (80GB).

H.3 Trigger Optimization Hyperparameters

Default trigger settings. Trigger optimization targets residual write channels and uses the following settings uniformly across all runs: trigger length 20, batch size 16, learning rate 0.003, KL weight $\alpha = 0.7$, tail-k KL with $k = 8$, ℓ_2 regularization weight $\beta = 0.1$, ℓ_2 max-norm $R = 1.0$, 200 prompt pairs, 2000 optimization steps. Prompt sources are task-matched: Alpaca prompts for Fixed Response, WildJailbreak 50/50 mix for Safety, and Shakespeare dataset prompts for style.

H.4 Evaluation Protocol

Fixed-response rate. Evaluation uses keyword matching against the set {"i don't know", "i do not know", "i'm not sure", "i am not sure"}, with response length capped at 40 words to separate clean fixed responses from hedged long-form answers.

Shakespeare LLM-judge rubric. The judge prompt asks for a single integer 0–5 and is reproduced verbatim below.

```
Rate the following response on a scale of 0 to 5 for how
Shakespearean it sounds.

0 = completely modern English, no archaic features
1 = occasional archaic word but mostly modern
2 = noticeably archaic vocabulary, some Shakespearean phrasing
3 = clearly Shakespearean style throughout
4 = strong Elizabethan English with authentic idioms and syntax
5 = authentic Shakespearean English indistinguishable from original

Response:
{response}

Reply with only a single integer: 0, 1, 2, 3, 4, or 5.
```

KL benchmark definition. For each prompt, the SFT model first generates a reference response autoregressively. Each evaluated model is then teacher-forced on [prompt; reference response], and we compute token-level KL divergence over the full vocabulary (response tokens only):

$$\mathrm{KL}_t(P\|Q) = \sum_v P_t(v)\big(\log P_t(v) - \log Q_t(v)\big).$$

Reported values are averages over response tokens and then over prompts, for $\mathrm{KL}(\mathrm{SFT}\|\mathrm{LCDD})$, $\mathrm{KL}(\mathrm{SFT}\|\mathrm{LCDD{+}Trig})$, and $\mathrm{KL}(\mathrm{Base}\|\mathrm{LCDD{+}Trig})$. Every evaluation prompt that also appears in the SFT/LCDD training set or the trigger optimization set is excluded. For the safety task, KL is computed on 189 WildJailbreak harmful prompts; for Fixed Response, on 199 Alpaca prompts; Shakespeare KL prompts have no overlap with training or trigger data.

Sampling and decode defaults. Main benchmarks use 200 evaluation prompts per condition, sampled with held-out seeds. Generation uses deterministic greedy decoding; maximum generation length is 128 tokens for Fixed Response and Shakespeare evaluation and 512 tokens for safety evaluation.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction accurately state our contributions and scope. See Section 1.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations are discussed in the Limitations paragraph of Section 5.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: The paper's theoretical content consists of gate–weight equivalence proofs, which are given in full in Appendix A.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Full hyperparameters, dataset details, evaluation protocols, and compute resources are provided in Appendix H.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Code is publicly available at the repository linked in the abstract; datasets used are publicly available and described in Appendix H.1.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: Training and evaluation details are described in Section 4.1 and fully specified in Appendix H.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: All experiments are single runs; error bars are not reported due to the computational cost.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Compute resources are specified in Appendix H.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conforms with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Positive directions are discussed in the Discussion paragraph and dual-use risks and mitigations are discussed in the Broader impact paragraph of Section 5.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [Yes]

    Justification: We do not release model weights, datasets, or optimized trigger embeddings; only the training and analysis code is released (Broader impact paragraph of Section 5).

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All models and datasets used are properly cited. Model licenses: Qwen3 (Apache 2.0), DeepSeek-R1-Distill (MIT), Mistral (Apache 2.0), Vicuna (Llama 2 Community License). Dataset licenses: Alpaca (CC BY-NC 4.0), WildJailbreak (ODC-By), WildGuard (Apache 2.0), HarmBench (MIT). The Shakespearean dataset [30] does not specify a license; we use it for research purposes only.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Code is released with documentation at the repository linked in the abstract; no new datasets or models are released.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: LLMs are the subject of study; Qwen3-14B is used as an LLM judge for Shakespeare style evaluation, as described in Section 4.1.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.