Crafting Reversible SFT Behaviors in Large Language Models

Anonymous Author(s)
Affiliation
Address
email

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, such correlations do not imply *causal necessity*, limiting the ability to selectively control SFT-induced behaviors at inference time. We pursue an alternative by asking: can an SFT-induced behavior be deliberately compressed into a sparse, mechanistically necessary subnetwork, termed a *carrier*, 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 prompt optimized via activation matching on extracted carrier channels, to reverse the SFT-induced behavior. Across safety, fixed-response, and style behaviors on multiple model families, LCDD yields sparse carriers that preserve target behaviors while enabling strong reversion when triggered by SFT-Eraser. Ablations further establish that the sparse structure is the key precondition for reversal: the same trigger optimization fails on standard SFT models, confirming that structure rather than trigger design is the operative factor. These results provide direct evidence that the learned carriers are causally necessary for the behaviors, pointing to a new direction for systematically localizing and selectively suppressing SFT-induced behaviors in deployed models. Code is available at `https://anonymous.4open.science/r/sft-reverse-8150/`.

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.

This lack of structure creates two compounding problems. **First**, it prevents *causal diagnosis*. 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. As long as behavior remains diffusely encoded, no amount of post-hoc analysis can conclusively establish whether a given substructure is truly necessary or merely correlated. **Second**, it limits *causal controllability*. Existing approaches do not provide a way to *instantiate* a sparse substructure whose presence is required for an SFT-induced behavior.

Submitted to 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Do not distribute.

Behavioral and representational analyses of SFT [4, 8, 9, 10, 11, 12, 13] and circuit localization methods [14, 15, 16, 17, 18, 19] can identify structure post-hoc but do not construct it. Task-vector and model-editing methods [20, 21] 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 is both causally necessary for an SFT-induced behavior and selectively suppressible via input alone under fixed weights.

We pursue a fundamentally different approach that addresses both problems at once. Rather than attempting to discover sparse structure in models that were never designed to induce it, we ask whether such structure can be deliberately constructed during training. When structure is imposed by design, the causal role of the carrier becomes substantially more identifiable: the behavior is intentionally concentrated within the carrier, and components outside it are maintained in their base-model state (i.e., before SFT). This makes causal necessity more directly testable via input interventions that target the carrier under fixed weights, rather than relying on ablations that may be confounded by redundant pathways [22]. If an input intervention suppresses SFT behavior while all model weights remain fixed, the sparse substructure it targets is demonstrated to be the causal bottleneck, not merely a correlated subnetwork. Beyond its role as a causal diagnostic, 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 and mechanistically necessary carriers that remain controllable at inference time without modifying model weights?

We term such a parameter substructure the *sparse behavioral carrier*. We study this question via two methods. We introduce **Loss-Constrained Dual Descent (LCDD)** to compress SFT behavior into a sparse carrier via mask-based compression, 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 toward 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, mechanistically necessary substructures, enabling their precise 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 validate causal necessity under fixed weights, we further introduce **SFT-Eraser**, an input-trigger[1] protocol that tests whether targeted input interventions can selectively suppress SFT-induced behaviors.
- We demonstrate that LCDD and SFT-Eraser enable inference-time reversibility across three behavior types and four model families. These results provide strong evidence for the feasibility of concentrating SFT-induced behaviors into sparse, causally necessary carriers, and show that the constructed carriers are not merely correlated with the behavior. Ablations further confirm that the sparse structure, rather than the trigger design, is the operative factor.

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 [8]. Depth-wise analyses find that alignment updates concentrate in specific layers [9, 10], and other work addresses catastrophic forgetting and objective mismatch [11, 12, 13]. SafeSeek frames circuit extraction as mask optimization and demonstrates highly sparse circuits for backdoor and alignment behaviors [17], 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 by construction the complete locus of the behavior, enabling causal validation rather than correlational attribution.

[1]While we borrow the term *trigger* from the backdoor literature [23], our trigger recovers the full functionalities of the base model before SFT, whereas a backdoor trigger activates a specific implanted behavior.

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 [20, 21]. 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) finds it insufficient for achieving the sparsity depth required for reliable reversal. Most directly related is Gao et al. [18], 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 but differs fundamentally in goal: rather than training sparse models de novo, 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 [14, 15, 16, 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 *causal necessity*: the carrier must be the complete locus of the behavior by construction, with model components outside the carrier reduced to their base-model state. Sparsity is essential to both properties: a compact, identifiable substructure can be precisely targeted at inference time, whereas a diffusely encoded behavior offers no designated target for such intervention. We pursue this in two stages as shown in Figure 1: crafting the sparse carrier via **LCDD** (Section 3.2), and validating its necessity via **SFT-Eraser** (Section 3.3). Both stages build on the delta parameterization introduced in Section 3.1.

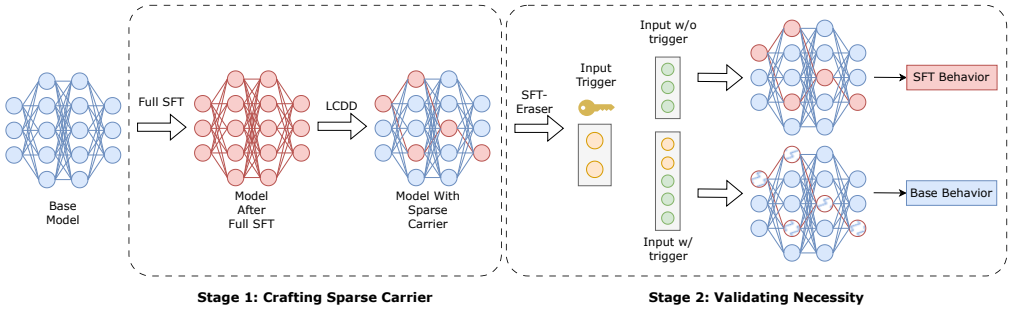

Figure 1: An overview of the LCDD + SFT-Eraser pipeline. **Stage 1**: standard SFT distributes the induced behavior broadly across model parameters (red), followed by LCDD that compresses the SFT-induced behavior into a sparse carrier. Components outside the carrier are reduced to their base-model state 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.

Weight mask implementation. We structure sparsity at the level of activation channels (individual rows and columns of weight matrices) rather than arbitrary weight elements, 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 [18]. Prior work has explored similar gradient-based mask optimization strategies for circuit discovery. Gao et al. [18] 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. [17] 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 delta contribution 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. [18], 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, rather than tuning a regularization coefficient whose effect on utility is indirect.

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 sufficient sparsity depth 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? If the trigger suppresses SFT behavior while reverting the output distribution toward the base model, the joint outcome implies that the carrier channels are mechanistically necessary rather than merely correlated with the behavior.

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. Targeting these channels is more causally precise than targeting intermediate hidden states.

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 rather than an artifact of carrier-level matching alone:

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

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). The necessity of $\mathcal{L}_{\text{MSE}}$ is validated empirically in Ablation 2 (Section 4.3).

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:** Is the crafted sparse structure a necessary precondition for reversal, rather than the trigger design itself?

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 carrier to craft and 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, placing greater demands on sparsification than Fixed Response.

- **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 genuinely induced by SFT rather than already present in the pretrained weights.

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 F.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 F.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 rather than free generation), averaged token-level over response tokens. KL(SFT‖LCDD) measures carrier fidelity before intervention (↓ better). KL(SFT‖Trig) and KL(Base‖Trig) jointly characterize distributional movement after triggering: larger KL(SFT‖Trig) indicates stronger divergence from SFT behavior; 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 F.

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

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	83.66%	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	1.811	1.037
Mistral-7B	76.72%	0.0%	100.0%	100.0%	0.5%	0.002	3.526	1.437
Vicuna-7B	38.28%	0.0%	100.0%	100.0%	0.0%	1.436	4.810	0.024

As shown in Table 1, the Fixed Response task provides the clear evidence that LCDD can craft a sparse carrier that is both behaviorally faithful before intervention and reliably reversible after triggering. LCDD preserves near-perfect fixed-response behavior across all four models (97.5–100%). For three models (Qwen, DeepSeek, Mistral), KL(SFT‖LCDD) is uniformly low (0.002–0.040), confirming that the sparse carrier is behaviorally near-indistinguishable from 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 (1.811–4.810) confirm strong divergence from SFT behavior, while KL(Base‖Trig) (0.024–1.437) 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	81.05%	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	36.5	39.5	41.5	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	0.256
		LCDD	1.5	96.5	2.0	54.8	56.3	SFT ‖ Trig	0.391
		Trig	13.0	56.5	30.5	49.2	54.1	Base ‖ Trig	0.306
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	0.269
		LCDD	1.0	97.5	1.0	57.0	63.6	SFT ‖ Trig	0.620
		Trig	17.0	69.0	21.5	54.1	62.3	Base ‖ Trig	1.282
Vicuna-7B	84.40%	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	1.081
		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, with the direction of change consistent across architectures.

Reversal magnitude varies across models, reflecting differences in carrier geometry. Notably, behavioral and distributional metrics capture partially independent aspects of reversal: Vicuna-7B achieves near-perfect distributional reversion (KL(Base‖Trig) = 0.014) despite moderate behavioral suppression (WG: -23.5pp), while Mistral-7B shows the strongest behavioral suppression (WG: -28.5pp) with a larger residual distributional gap (KL(Base‖Trig) = 1.282). Both patterns confirm that the trigger successfully targets the carrier in all cases.

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 targets carrier-specific channels without broadly disrupting general language modeling.

4.2.3 Shakespeare 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 F.4). MMLU and HellaSwag values in %. KL values computed on SFT-generated references; the $P \,\|\, Q$ column indicates the distributions compared. Sparsity is the LCDD mask sparsity.

			Style	Utility Metrics		KL Divergence	
Model	Sparsity	Condition	Judge ↑	MMLU	HellaSwag	$P \,\|\, Q$	Value
Qwen3-0.6B	56.34%	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	41.9	37.3	Base ‖ Trig	0.446
DeepSeek-8B	54.54%	Base	0.01	54.1	55.7		
		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

The Shakespeare task is the most demanding for LCDD: achieved sparsity is substantially lower (29.9–56.3%) than in Fixed Response or safety runs, 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 (0.26–0.98) recover only part of the SFT-level style 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.

Even where carrier fidelity is partial, trigger intervention suppresses style effectively in all four cases: Judge@Trig falls to near-base levels (0.005–0.07) regardless of the LCDD starting point, indicating that even a partially concentrated carrier can be targeted successfully by the trigger. 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.

A consistent side effect in Shakespeare runs is larger MMLU degradation under trigger compared to Fixed Response or safety conditions (e.g., DeepSeek: 55.4%→37.4%). We conjecture that style carrier channels overlap more with general language modeling representations than behavior-specific carrier channels, making them harder to target without collateral disruption.

4.2.4 Cross-Task Observations

Three patterns are consistent across all 12 model-task combinations:

- **Sparsity reflects task localizability.** Fixed Response compresses most aggressively (73–84%), safety to an intermediate level (59–84%), and style least so (30–56%). This ordering reflects the structural complexity of each behavior: a fixed lexical response can be concentrated in a small parameter substructure, whereas distributional stylistic behavior requires broader representational support and resists aggressive compression.
- **Reversal direction is universal; magnitude is heterogeneous.** All 12 combinations show movement in the correct direction after triggering. Variation in degree reflects model- and task-specific carrier geometry, not method failure.
- **Carrier fidelity and reversal quality are partially decoupled.** Poor carrier fidelity does not preclude effective trigger-based suppression. In the Shakespeare task, where carrier fidelity is lower than in Fixed Response or safety runs, the trigger still successfully suppresses style in all four models. This suggests that trigger effectiveness depends on the presence of the crafted bottleneck rather than on how completely the carrier captures the behavior.

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, with consistent direction of change even where magnitude varies.

4.3 Ablation Study

We investigate RQ3 along two dimensions. First, we test whether the crafted sparse structure is a necessary precondition for reversal, or whether the same trigger optimization succeeds on any SFT model. Second, we examine whether the circuit-targeted MSE term in SFT-Eraser is necessary, or whether output-level KL pressure alone suffices. Both experiments are conducted on DeepSeek-R1-Distill-Llama-8B under the safety task, using the main experiment as the reference baseline. We additionally verify the necessity of joint weight optimization in LCDD in Appendix E.

Setup. We factor the experiment along two dimensions: model structure (LCDD vs. plain SFT) and trigger objective (circuit-targeted vs. output-only), yielding four conditions. The circuit-targeted trigger uses the full loss $\mathcal{L}_{\text{trigger}} = \mathcal{L}_{\text{MSE}} + \alpha\,\mathcal{L}_{\text{KL}} + \beta\,\mathcal{L}_2$; the output-only trigger removes $\mathcal{L}_{\text{MSE}}$, leaving $\alpha\,\mathcal{L}_{\text{KL}} + \beta\,\mathcal{L}_2$. Each trigger objective is applied to both the LCDD model and the plain SFT model, giving four conditions in total: LCDD + circuit trigger (main method), LCDD + output-only trigger, SFT + circuit trigger, and SFT + output-only trigger.

Finding 1: the sparse carrier structure is the decisive factor. Both LCDD conditions substantially outperform both SFT conditions in behavioral reversal (-40pp/-31pp vs. -21pp/-24pp). The KL metrics reveal the mechanism. SFT triggers do perturb the output (KL(SFT‖Trig) $\approx$0.21). However,

Table 4: Structural dependence ablation: safety reversal and distributional reversion on DeepSeek-R1-Distill-Llama-8B. WG = WildGuard refusal rate; MMLU values are post-trigger.

		WG Refusal			KL Divergence		
Model	Condition	No Trig	+Trig	Δ	SFT‖Trig	Base‖Trig	MMLU
Base	(reference)	47.5%	–	–	–	–	54.1%
LCDD	No trigger	96.5%	–	–	–	–	54.8%
	+ circuit	96.5%	56.5%	−40pp	0.391	0.306	49.2%
	+ output-only	96.5%	65.5%	−31pp	0.440	0.322	37.2%
SFT	No trigger	100.0%	–	–	–	–	55.3%
	+ circuit	100.0%	78.5%	−21pp	0.221	0.539	42.0%
	+ output-only	100.0%	76.5%	−24pp	0.208	0.512	40.9%

the triggered SFT model cannot revert toward the base model distribution (KL(Base‖Trig) ≈0.51–0.54). LCDD conditions achieve KL(Base‖Trig) ≈0.31. Triggered outputs genuinely revert toward the base model. In LCDD models, components outside the carrier are reduced to their base-model state by construction. The carrier is therefore the only locus of SFT-induced computation. Any trigger-based reversion must be mediated through the carrier, not non-carrier channels. The trigger optimization procedure alone, without a crafted carrier, is insufficient.

Finding 2: the circuit MSE term improves targeting precision and reduces utility cost. Removing $\mathcal{L}_{\text{MSE}}$ (LCDD + output-only trigger) reduces behavioral reversal from -40pp to -31pp. It also substantially increases MMLU degradation (-17.6pp vs. -5.6pp). KL(Base‖Trig) remains similar (0.322 vs. 0.306). The MSE term provides a gradient signal focused on carrier channels. This improves optimization efficiency and preserves utility by avoiding disruption of non-carrier activations.

5 Conclusion

We investigated whether SFT-induced behaviors can be deliberately compressed into sparse behavioral carriers that are preserved under normal inference yet reversibly suppressible by an input trigger at inference time without weight modification. LCDD crafts such carriers through utility-budgeted joint optimization of routing masks and weight deltas; SFT-Eraser validates their mechanistic necessity via activation matching. Our experiments and ablations establish that joint weight optimization is required to craft a reliable sparse bottleneck, and that the bottleneck, rather than the trigger design, is the necessary precondition for trigger-based reversal.

Discussion. Beyond its role as a mechanistic diagnostic, 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 question 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. The most immediate open problem is extending reversal to discrete input tokens rather than continuous soft prompts, which would make the behavioral interface genuinely deployable; a second direction is studying how pretraining scale affects the achievable sparsity–utility frontier for crafted carriers.

Limitations. Two limitations bound the current study. First, SFT-Eraser optimizes in continuous embedding space. The trigger is a soft prompt realized as a learned token-embedding sequence rather than a discrete hard-token string. The reversal evidence is valid as a mechanistic diagnostic under controlled conditions, but extending this to discrete hard-token triggers requires further investigation. Second, the structural necessity ablation is conducted on a single model and task combination. Whether the same causal relationship holds consistently across all model families and behavior types remains to be verified more systematically. We also note that the same methodology could in principle be used to remove safety alignment from deployed models; this dual-use risk warrants careful consideration in any deployment context.

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

[9] 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.

[10] 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.

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

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[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

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

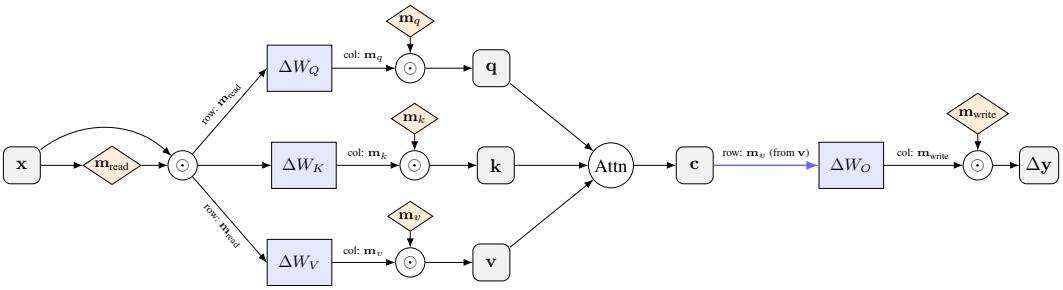

Figure 2: Delta-path computation graph for the attention layer with activation gating. Input $\mathbf{x}$ is gated by $\mathbf{m}_{\text{read}}$ before entering all projection deltas; Q/K/V outputs are independently gated by $\mathbf{m}_q, \mathbf{m}_k, \mathbf{m}_v$. The row mask of ΔW_O is $\mathbf{m}_v$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

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

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

□

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.

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.

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	$d_{\text{inner}} \times d$	$\mathbf{m}_v$	$\mathbf{m}_{\text{write}}$

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:

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

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; [18] 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.

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 $M^* = \{m_i\}, \quad \Delta W^*$

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 F.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 [17], applied here to the SFT delta path. We evaluate whether this simpler setup achieves comparable sparsity and reversal quality.

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.

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%	0.256	0.391	0.306
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	0.319	0.197

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 approximately $3\times$ less than the 81% achieved by joint optimization. Without weight adaptation, mask deletions directly degrade 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. This allows masks to compress ≈$3\times$ deeper while keeping behavior within the utility budget.

Finding 2: shallower sparsity yields a weaker bottleneck. At 28–34% sparsity, mask-only triggers produce weaker reversal. WG refusal endpoint is 64.0% (CE) and 60.0% (KL), compared to 56.5% for LCDD joint (-35pp and -38pp vs. -40pp). KL(SFT‖Trig) is lower (0.262/0.319 vs. 0.391), indicating a smaller distributional shift. KL(Base‖Trig) is also lower (0.254/0.197 vs. 0.306), meaning triggered outputs do not revert as fully toward the base model. The reason is structural. In mask-only training, W_{ft} is fixed. The masked-in deltas retain the distributed SFT representation from full fine-tuning. They are not reorganized into a concentrated carrier. Joint optimization trains the masked-in deltas to be the complete locus of SFT behavior. Disrupting the carrier then disrupts the behavior in its entirety.

Synthesis. Mask-only optimization cannot reach the sparsity regime that joint optimization achieves. Even at its achieved sparsity, the carrier is structurally incomplete. The masked-in components retain a distributed representation. The trigger has no concentrated target to disrupt. Joint weight optimization is therefore necessary both for achieving sufficient compression depth and for organizing the carrier as a targetable locus of SFT behavior. Together with the structural dependence ablation (Section 4.3), these findings establish that both joint optimization and the resulting sparse structure are necessary preconditions for reliable trigger-based reversal.

F Experimental Details

F.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 strictly held-out pool by excluding those first 5,000 examples, shuffling 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. This dataset is directly suitable for supervised sequence mapping without additional alignment construction, as it already provides paired modern / Shakespearean dialogue turns at the utterance level. Main runs use 5,000 training samples.

F.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 7 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 7: 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).

F.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.

F.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. We additionally compute AWR (archaic word rate via word-boundary regex match against a Shakespearean lexicon) as a supplementary lexical measure, though it is not reported in the main results tables. 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 $\mathrm{KL}(\mathrm{Base}\|\mathrm{LCDD{+}Trig})$.

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

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes], [No], or [N/A].
- [N/A] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for [N/A]).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will also be asked to include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While [Yes] is generally preferable to [No], it is perfectly acceptable to answer [No] provided a proper justification is given (e.g., error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering [No] or [N/A] is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

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 F.

   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 via anonymous repository; datasets used are publicly available and described in Appendix F.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 F.

   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 F.

   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 are acknowledged in the Limitations 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: [N/A]

Justification: We do not release model weights or datasets; no additional safeguards are required.

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 via anonymous repository; 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.