DEPTHGATE: DEPTH-AWARE GATED STEERING FOR INSTRUCTION-DATA SEPARATION IN LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models often process **trusted instructions** that specify the task and **untrusted data** that supply task-relevant content within the same context. Prompt injection arises when adversarial instructions embedded in untrusted data override the trusted task. Instruction-data separation makes this role boundary explicit, but existing approaches enforce it through system-level controls or input-level role encoding, leaving its treatment during depth-wise model computation largely unspecified. We introduce **DepthGate**, a depth-aware framework that extends instruction-data separation from input formatting to hidden-state computation, carrying the trusted role boundary into depth-wise model computation. DepthGate achieves this through three complementary mechanisms. **Role-Conditioned Steering** introduces continuous role-aware gates to control the redirection of data representations, while keeping instruction representations unchanged and preserving the intrinsic geometry of data. **Depth-Aware Paths** organize these gates across model layers, providing model-specific control over where and how strongly the separation boundary is enforced during forward computation. **Schedule Discovery** further identifies steering schedules under schedule-matched adaptation and selects a model-specific schedule using held-out data before final post-training. Experiments across five Llama and Qwen models show that DepthGate consistently improves instruction-data separation, with its most consistent reductions in marked-data attack success. Different schedule variants further provide flexible robustness–utility trade-offs. On Qwen2.5 7B, the high-separation configuration raises the separation score from 51.0% to 92.1% and reduces StruQ-ID attack success from 72.4% to 11.9%.

1 INTRODUCTION

Large language models (LLMs) often process text with different functional roles in a shared context. An **instruction** specifies the task, whereas **data** provides content that the model should read, summarize, transform, or use as evidence. For an email assistant, a request to extract a date and an invoice amount is the instruction, while the email itself is the data. Prompt injection occurs when the model follows instruction-like text embedded in such data and redirects its behavior away from the trusted task (Perez & Ribeiro, 2022; Greshake et al., 2023; Schulhoff et al., 2023; Liu et al., 2024; Li et al., 2024). We study **instruction-data separation**: an LLM should follow trusted instructions while using untrusted data only as task content. This problem is central to retrieval, document processing, and tool-mediated systems, where task-relevant external content may also contain adversarial instructions (Lewis et al., 2020; Yao et al., 2022; Schick et al., 2023; Zhan et al., 2024; Debenedetti et al., 2024).

Existing methods expose role hierarchy through structured inputs, runtime filters, or safety training (Chen et al., 2025a; Hines et al., 2024; Chen et al., 2025b; Inan et al., 2023). Representation methods encode roles through segment embeddings or data-only input transformations (Chen et al., 2025a; Wu et al., 2025; Zverev et al., 2025). These methods establish the boundary before internal computation but do not control how it is maintained as Transformer updates reshape hidden states. Extending separation into the model requires role-dependent processing across depth that preserves normal data use and is matched to model adaptation.

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

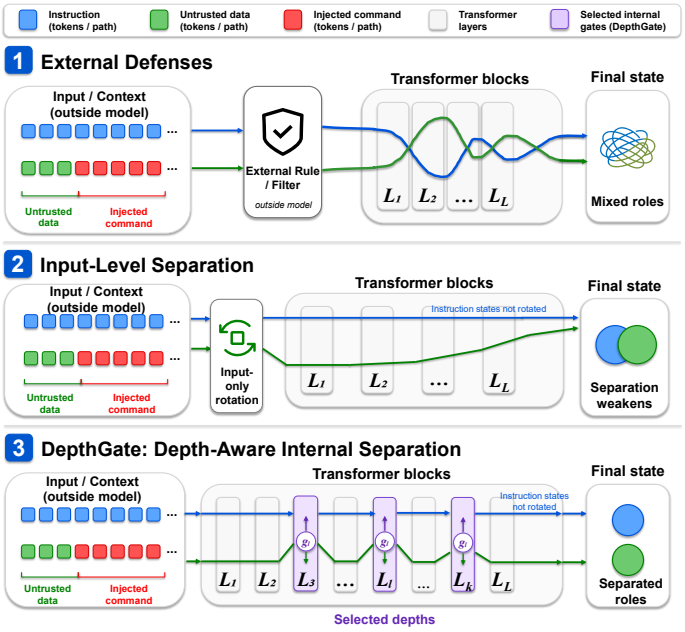


Figure 1: **DepthGate motivation. External defenses** enforce the instruction-data boundary outside the model. **Input-level separation** establishes it once but does not maintain its effect as hidden states evolve. **DepthGate** reinforces the boundary at selected depths, forming a Depth-Aware Path that carries it into depth-wise hidden-state computation.

We propose **DepthGate**, a framework that extends the trusted instruction-data boundary from input formatting into depth-wise hidden-state computation. Input role marking establishes the boundary, and DepthGate reinforces it at selected depths. Role-Conditioned Steering applies continuous-strength operations to data states while preserving their internal geometry. Their composition with Transformer blocks defines a model-specific Depth-Aware Path. Schedule Discovery selects this path under schedule-matched adaptation.

DepthGate has three components. **Role-Conditioned Steering** leaves instruction states unchanged and continuously redirects data states while preserving their internal geometry. **Depth-Aware Paths** compose these operations with Transformer updates at selected depths. **Schedule Discovery** constructs candidate paths from held-out role-marked data, evaluates them under schedule-matched short adaptation, and selects a model-specific schedule before final post-training. Together, the components maintain a trusted input boundary during internal computation.

DepthGate extends instruction-data separation from input encoding to depth-wise boundary maintenance. It combines structure-preserving Role-Conditioned Steering with model-specific Depth-Aware Paths selected through Schedule Discovery. Across five models, DepthGate improves separation and delivers its most consistent robustness gains against attacks carried by marked data. On Qwen2.5 7B, DepthGate-Continuous raises SEP from 51.0% to 92.1% and reduces StruQ-ID attack success from 72.4% to 11.9%. Its schedules provide complementary separation–utility operating points.

2 Related Work

Instruction-Data Separation. StruQ exposes the instruction-data boundary through structured fields, while ISE encodes the roles with segment embeddings (Chen et al., 2025a; Wu et al., 2025). ASIDE applies a fixed orthogonal rotation to data-token embeddings and leaves instruction embeddings unchanged (Zverev et al., 2025). These methods establish the distinction at input but do not maintain it as hidden states are repeatedly updated across depth. DepthGate extends separation into depth-wise computation through selected role-conditioned interventions.

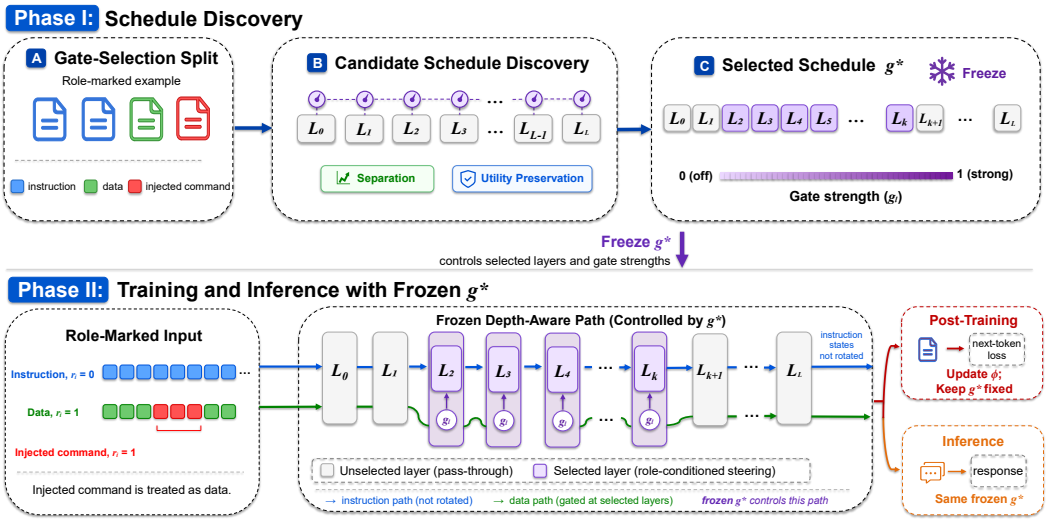


Figure 2: **DepthGate pipeline.** Input role marking establishes the trusted instruction–data boundary, and DepthGate maintains it through a model-specific path. **Phase I: Schedule Discovery** scores model depths, constructs candidate schedules, and selects $\mathbf{g}^\star$ after schedule-matched short adaptation. **Phase II: Path-Matched Post-Training** freezes $\mathbf{g}^\star$. Selected Role-Conditioned Steering operations reinforce the boundary, and their ordered composition defines a Depth-Aware Path for final post-training and inference.

Prompt-Injection Defenses. Prompt-injection defenses use structured interfaces, transformations of untrusted content, security alignment, and system-level controls. StruQ separates fields, Spotlighting transforms untrusted text, and SecAlign trains models to prioritize trusted instructions (Chen et al., 2025a; Hines et al., 2024; Chen et al., 2025b). Alignment, jailbreak defenses, and system-level designs further shape behavior or restrict control and data flow (Ouyang et al., 2022; Bai et al., 2022a;b; Rafailov et al., 2023; Zou et al., 2023b; Wei et al., 2023; Chao et al., 2024; Robey et al., 2023; Debenedetti et al., 2025). DepthGate complements these safeguards by maintaining the trusted role boundary inside the model.

Hidden-State Interventions. Transformer analyses show that linguistic, factual, and behavioral information varies across depth (Tenney et al., 2019; Jawahar et al., 2019; Rogers et al., 2020; Belrose et al., 2023; Meng et al., 2022a). Model editing, representation steering, and jailbreak interventions use layer-specific parameter or activation changes (De Cao et al., 2021; Mitchell et al., 2021; Meng et al., 2022b; Dai et al., 2022; Li et al., 2023a; Zou et al., 2023a; Turner et al., 2024; Panickssery et al., 2023; Todd et al., 2024; Wu et al., 2024; Zhao et al., 2024). Their objectives are behavioral or task-specific rather than trusted role labels. DepthGate conditions steering on roles and selects ordered paths that maintain the boundary across depth.

3 Challenges and Motivation

Instruction-data separation poses three connected challenges: role labels must induce distinct processing without limiting data use; the boundary must remain effective as hidden states evolve; and its maintenance path must be selected under matched adaptation.

Role Separation Requirements

Let I denote the trusted instruction, D the task-relevant data, and $D \oplus a$ the result of inserting a command-like span a into the data. For a model with parameters ϕ and output distribution p_ϕ, let $y^\star(I, D)$ be the target response determined by the original task. Effective separation should retain a

high probability for this target under the insertion:

$$p_\phi(y^\star(I, D) \mid I, D \oplus a) \approx p_\phi(y^\star(I, D) \mid I, D)\,. \tag{1}$$

This objective requires distinct processing without blocking data use, continuous rather than maximal separation, and preservation of relations among data states. Zero steering provides no internal reinforcement, whereas repeated fixed 90° rotations can weaken necessary cross-role interaction. These requirements motivate asymmetric, continuously controlled, and structure-preserving **Role-Conditioned Steering**.

Depth Dependence

The input role boundary is not static under model computation: attention, feed-forward, and residual updates reshape hidden states after each transformation. Let $H^{(l)}$ be the hidden-state sequence at layer l, F_{l+1} the next Transformer block, and $\mathcal{T}_{g_l}$ the local transformation with gate g_l. In general,

$$F_{l+1}\Big(\mathcal{T}_{g_l}(H^{(l)})\Big) \neq \mathcal{T}_{g_l}\Big(F_{l+1}(H^{(l)})\Big)\,. \tag{2}$$

The intervention layer changes subsequent updates. Input-only separation establishes the boundary once, whereas uniform intervention can overconstrain data use. Maintaining it requires selected operations whose placement, strength, and order are controlled, motivating **Depth-Aware Paths**.

Path Selection

Candidate paths must be compared under the adaptation they induce. For initial parameters ϕ_0 and schedule $\mathbf{g}$, each candidate receives K-step adaptation, yielding $\phi_{\mathbf{g}} = \mathrm{Adapt}_K(\phi_0; \mathbf{g})$, before held-out selection. Because useful patterns vary and exhaustive search is impractical, **Schedule Discovery** constructs a compact set and compares matched schedule–parameter pairs.

4 Method

DepthGate carries the instruction-data boundary into depth-wise hidden-state computation. Input role labels establish it, selected **Role-Conditioned Steering** operations reinforce it locally, and **Depth-Aware Paths** organize their composition with Transformer updates. **Schedule Discovery** selects the model-specific path under schedule-matched adaptation. Figure 2 summarizes the method.

Role-Conditioned Steering

Role-Conditioned Steering is the local mechanism for reinforcing the trusted role boundary at an active depth without removing data from shared computation. Instruction states remain unchanged at the steering operation, whereas data states receive a gate-controlled transformation whose strength can vary continuously. Let $h_i^{(l)} \in \mathbb{R}^d$ be the row-vector hidden state of token i at layer l, where d is the hidden dimension. The input schema supplies a fixed role indicator $r_i \in \{0, 1\}$, where $r_i = 0$ denotes instruction and $r_i = 1$ denotes data. This indicator records the known functional role of each token, not whether that token is malicious.

To reinforce this boundary without learning an additional steering direction, DepthGate uses a fixed quarter-turn matrix $R \in \mathbb{R}^{d\times d}$ to generate a continuous family of rotations. For the even hidden dimensions considered here, R is block diagonal with repeated blocks $J = \left[\begin{smallmatrix} 0 & -1 \\ 1 & 0 \end{smallmatrix}\right]$. Hence $R^\top R = \mathbf{I}_d$, $R^\top = -R$, and $R^2 = -\mathbf{I}_d$, where $\mathbf{I}_d$ is the identity matrix. A local gate $g_l \in [0, 1]$ interpolates from the identity mapping to this quarter-turn:

$$\theta_l = \frac{\pi}{2} g_l, \qquad Q(g_l) = \cos(\theta_l)\mathbf{I}_d + \sin(\theta_l)R. \tag{3}$$

The role indicator applies this rotation only to data states:

$$\tilde{h}_i^{(l)} = h_i^{(l)} + r_i\Big(h_i^{(l)} Q(g_l) - h_i^{(l)}\Big)\,. \tag{4}$$

Equation 4 leaves instruction states unchanged for $r_i = 0$ and applies $Q(g_l)$ to data states for $r_i = 1$. The endpoints $g_l = 0$ and $g_l = 1$ give no rotation and a quarter-turn, respectively, while intermediate

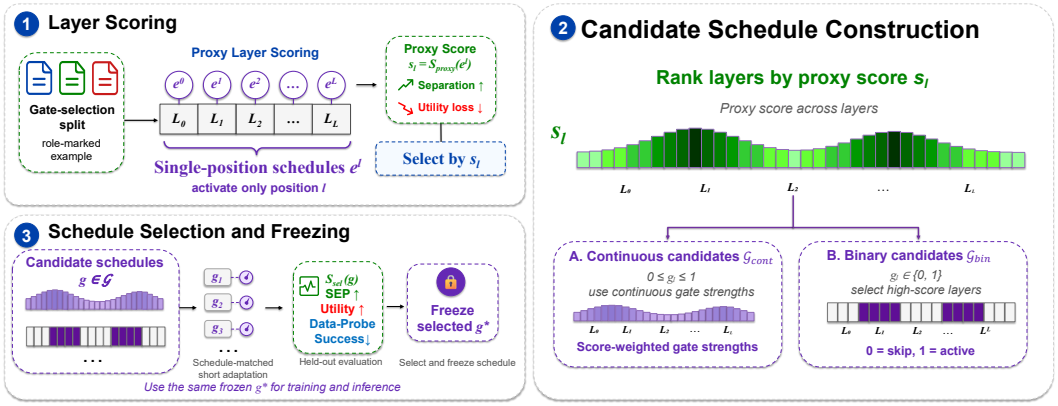


Figure 3: **Schedule Discovery selects a model-specific boundary-maintenance path.** Single-layer evaluations rank model depths on held-out role-marked data. This profile defines continuous schedules with score-weighted strengths and binary schedules with selected active layers. Each candidate is evaluated after schedule-matched short adaptation, and the best valid schedule $\mathbf{g}^\star$ is frozen for final post-training and inference.

values control rotation strength. For any pre-transform state h, the cosine similarity between h and $hQ(g_l)$ is $\cos(\theta_l)$. Thus, g_l controls the local angular offset while subsequent shared computation can still combine both roles.

The continuous control does not rely on distorting the local geometry of data states. From the properties of R, Eq. 3 satisfies $Q(g_l)^\top Q(g_l) = \mathbf{I}_d$ for every $g_l \in [0, 1]$. For any two data states h_a and h_b transformed at the same layer,

$$\begin{aligned} \|h_a Q(g_l)\|_2 &= \|h_a\|_2 , \\ (h_a Q(g_l))\,(h_b Q(g_l))^\top &= h_a h_b^\top . \end{aligned} \tag{5}$$

Equation 5 shows that the operation preserves norms and pairwise inner products among data states at the intervention layer. Their orientation relative to unchanged instruction states can vary, but their local relational geometry remains fixed. Role-Conditioned Steering therefore reinforces the role boundary with controllable strength without collapsing or arbitrarily distorting data representations.

Depth-Aware Paths

Depth-Aware Paths organize local steering across the full hidden-state computation. A schedule specifies selected depths and strengths, while its composition with Transformer updates defines the ordered trajectory that maintains the boundary. For intervention indexing, we use layers $l = 0, \ldots, L$: layer 0 is the embedding output, and layer $l \geq 1$ is the output of Transformer block l. A layerwise schedule $\mathbf{g} = (g_0, \ldots, g_L) \in [0, 1]^{L+1}$ specifies the rotation strength at every layer; $g_l = 0$ leaves layer l inactive.

Let F_l denote Transformer block l, and let $\mathcal{T}_{g_l}$ apply Eq. 4 to the hidden-state sequence at layer l. Starting from the embedding output, the schedule defines the transformation path

$$\mathcal{P}_{\mathbf{g}} = \mathcal{T}_{g_L} \circ F_L \circ \mathcal{T}_{g_{L-1}} \circ \cdots \circ F_1 \circ \mathcal{T}_{g_0}, \tag{6}$$

In Eq. 6, each gate determines local strength, its index determines placement, and composition with Transformer blocks fixes the order of intervention. The schedule records the gate values, while the path is the full hidden-state computation induced by their ordered composition with Transformer updates.

Input-only separation is the special case $g_0 = 1$ and $g_l = 0$ for $l > 0$, which establishes the boundary once without later reinforcement. A full-rotation uniform path sets $g_l = 1$ at every layer and can impose stronger separation than normal data use requires. Continuous schedules allocate intermediate rotation strengths, whereas binary schedules select an explicit set of full-rotation layers.

Their selected operations maintain the boundary through different ordered computations, allowing discovery to determine where and how strongly it should be reinforced for each model.

SCHEDULE DISCOVERY

Schedule Discovery selects the model-specific schedule that defines a Depth-Aware Path while keeping selection separate from final benchmarks. Since useful reinforcement depths and strengths vary across architectures and exhaustive gate search is impractical, it builds a compact candidate set from held-out role-marked data. It estimates a depth profile, constructs continuous and binary schedules, and evaluates each after schedule-matched short adaptation. Exact scoring functions, candidate grids, and selection hyperparameters are reported in the supplementary material.

Layer Scoring. Layer scoring identifies depths at which local reinforcement is most useful and builds the profile used to guide candidate generation. Let $\mathbf{e}^{(l)}$ be a single-layer schedule with $e_k^{(l)} = \mathbb{1}[k = l]$, where $\mathbb{1}$ is the indicator function. Its proxy score is

$$s_l = S_{\text{proxy}}\left(\mathbf{e}^{(l)}\right). \tag{7}$$

The proxy score combines behavioral separation, instruction-side probe success, data-side probe execution, and Same-Output Rate, a collapse diagnostic for identical outputs across role placements. It ranks layers and supplies relative weights for candidate construction but does not select the final schedule.

Candidate Construction. Candidate construction compresses the combinatorial schedule space into a continuous family $\mathcal{G}_{\text{cont}}$ and a binary family $\mathcal{G}_{\text{bin}}$, with candidate set $\mathcal{G} = \mathcal{G}_{\text{cont}} \cup \mathcal{G}_{\text{bin}}$. Continuous schedules assign score-normalized strengths to selected layers, allowing intermediate rotations. Binary schedules activate selected layers with full local rotations. These families produce DepthGate-Continuous and DepthGate-Binary and define alternative boundary-maintenance paths. Candidate construction uses only the proxy ranking, and each candidate remains fixed throughout its own short adaptation.

Schedule-Matched Selection. To compare candidates under the computations they induce, each candidate $\mathbf{g}$ receives a fresh short LoRA adaptation, producing matched parameters $\phi_{\mathbf{g}} = \text{Adapt}_K(\phi_0; \mathbf{g})$. The candidate is then evaluated on held-out candidate-selection examples. For either family $\mathcal{F}$, let $\mathcal{F}_{\text{valid}}$ contain the candidates that satisfy the instruction-following, data-side execution, and same-output checks. The selected schedule is

$$\begin{aligned} \mathbf{g}^\star_{\mathcal{F}} &= \arg \max_{\mathbf{g} \in \mathcal{F}_{\text{valid}}} S_{\text{sel}}(\mathbf{g}, \phi_{\mathbf{g}}), \\ \mathcal{F} &\in \{\mathcal{G}_{\text{cont}}, \mathcal{G}_{\text{bin}}\}. \end{aligned} \tag{8}$$

Here S_{sel} is the held-out selection score. Equation 8 compares schedules only with their matching adapted parameters. The highest-scoring valid schedule $\mathbf{g}^\star_{\mathcal{F}}$ for each family is frozen before final post-training.

Post-Training and Inference. After discovery, path-matched post-training holds each selected schedule $\mathbf{g}^\star_{\mathcal{F}}$ fixed during final LoRA-based supervised training on role-marked examples. This process adapts the model parameters to the selected boundary-maintenance path. At inference, DepthGate reuses the trusted role indicators and selected path without online schedule search.

5 EXPERIMENTS

We evaluate whether DepthGate keeps the trusted role boundary behaviorally effective after full hidden-state computation, reduces prompt-injection success, and retains instruction-following utility.

EXPERIMENTAL SETUP

Models and Baselines We evaluate five decoder-only large language models: Llama 2 7B and 13B, Llama 3.1 8B, Qwen2.5 7B, and Qwen3 8B (Touvron et al., 2023; Grattafiori et al., 2024;

Model	Method	Attack Success Rate [%]↓							
		Direct Attacks				Indirect Attacks			
		TensorTrust	Gandalf	Purple	RuLES	BIPIA-Text	BIPIA-Code	StruQ-ID	StruQ-OOD
Llama 2 7B	Original	47.5$_{\pm 5.1}$	51.9$_{\pm 0.3}$	74.7$_{\pm 0.3}$	75.2$_{\pm 0.9}$	17.9$_{\pm 0.3}$	22.1$_{\pm 0.9}$	47.2$_{\pm 0.3}$	51.6$_{\pm 0.9}$
	ISE	42.6$_{\pm 3.4}$	55.0$_{\pm 2.2}$	73.6$_{\pm 1.3}$	79.1$_{\pm 0.9}$	16.8$_{\pm 0.4}$	20.5$_{\pm 0.9}$	48.4$_{\pm 0.8}$	52.8$_{\pm 0.9}$
	ASIDE	39.4$_{\pm 4.0}$	<u>28.1</u>$_{\pm 3.0}$	72.3$_{\pm 1.8}$	77.5$_{\pm 0.8}$	1.4$_{\pm 0.4}$	<u>17.5</u>$_{\pm 1.6}$	33.5$_{\pm 0.7}$	<u>43.5</u>$_{\pm 0.2}$
	DepthGate-Continuous	**30.9**$_{\pm 2.6}$	**27.4**$_{\pm 2.2}$	<u>70.5</u>$_{\pm 1.7}$	**62.7**$_{\pm 1.0}$	**0.4**$_{\pm 0.1}$	**16.4**$_{\pm 0.9}$	**20.5**$_{\pm 0.8}$	**40.8**$_{\pm 0.8}$
	DepthGate-Binary	<u>37.6</u>$_{\pm 2.8}$	34.0$_{\pm 2.3}$	**68.1**$_{\pm 1.6}$	<u>66.8</u>$_{\pm 1.0}$	<u>1.1</u>$_{\pm 0.2}$	18.2$_{\pm 1.0}$	<u>30.8</u>$_{\pm 0.7}$	45.0$_{\pm 0.9}$
Llama 2 13B	Original	51.1$_{\pm 4.2}$	52.4$_{\pm 2.7}$	72.4$_{\pm 1.5}$	76.0$_{\pm 1.2}$	16.8$_{\pm 0.8}$	**17.1**$_{\pm 1.9}$	40.5$_{\pm 0.2}$	50.9$_{\pm 0.7}$
	ISE	53.4$_{\pm 2.0}$	9.8$_{\pm 1.0}$	73.1$_{\pm 1.8}$	77.2$_{\pm 1.3}$	16.2$_{\pm 0.6}$	<u>18.6</u>$_{\pm 1.5}$	40.0$_{\pm 0.6}$	51.2$_{\pm 0.8}$
	ASIDE	43.8$_{\pm 1.5}$	**0.5**$_{\pm 0.7}$	70.3$_{\pm 2.2}$	71.1$_{\pm 1.4}$	3.1$_{\pm 0.5}$	24.2$_{\pm 1.4}$	38.9$_{\pm 0.4}$	<u>49.4</u>$_{\pm 1.1}$
	DepthGate-Continuous	**35.8**$_{\pm 1.6}$	<u>1.3</u>$_{\pm 0.8}$	<u>68.5</u>$_{\pm 1.9}$	**55.1**$_{\pm 1.3}$	**0.7**$_{\pm 0.1}$	22.7$_{\pm 1.1}$	**25.8**$_{\pm 0.6}$	**47.8**$_{\pm 0.9}$
	DepthGate-Binary	<u>41.7</u>$_{\pm 1.6}$	5.9$_{\pm 1.0}$	**66.7**$_{\pm 1.8}$	<u>59.3</u>$_{\pm 1.3}$	<u>2.5</u>$_{\pm 0.4}$	24.8$_{\pm 1.3}$	<u>35.1</u>$_{\pm 0.6}$	52.0$_{\pm 1.0}$
Llama 3.1 8B	Original	57.0$_{\pm 1.3}$	61.2$_{\pm 0.7}$	74.9$_{\pm 1.2}$	66.0$_{\pm 2.9}$	20.1$_{\pm 0.2}$	20.1$_{\pm 0.6}$	45.8$_{\pm 1.0}$	46.3$_{\pm 0.6}$
	ISE	59.4$_{\pm 1.5}$	52.8$_{\pm 2.4}$	78.6$_{\pm 1.4}$	67.8$_{\pm 1.7}$	15.4$_{\pm 0.4}$	17.2$_{\pm 0.8}$	43.9$_{\pm 0.7}$	48.0$_{\pm 1.0}$
	ASIDE	<u>32.9</u>$_{\pm 0.8}$	41.4$_{\pm 4.7}$	**67.5**$_{\pm 0.7}$	<u>53.8</u>$_{\pm 1.6}$	<u>3.3</u>$_{\pm 0.0}$	**5.8**$_{\pm 0.5}$	<u>21.3</u>$_{\pm 0.6}$	<u>36.8</u>$_{\pm 0.7}$
	DepthGate-Continuous	**29.8**$_{\pm 1.4}$	**36.7**$_{\pm 1.5}$	70.2$_{\pm 1.5}$	**45.6**$_{\pm 1.5}$	**0.8**$_{\pm 0.2}$	<u>6.2</u>$_{\pm 0.6}$	**8.7**$_{\pm 0.5}$	**35.6**$_{\pm 0.7}$
	DepthGate-Binary	38.6$_{\pm 1.6}$	<u>39.2</u>$_{\pm 1.1}$	<u>68.7</u>$_{\pm 1.8}$	58.6$_{\pm 0.9}$	4.8$_{\pm 0.4}$	12.4$_{\pm 0.9}$	30.6$_{\pm 0.4}$	47.2$_{\pm 0.7}$
Qwen2.5 7B	Original	45.3$_{\pm 0.8}$	58.8$_{\pm 1.2}$	78.4$_{\pm 1.4}$	65.3$_{\pm 2.3}$	18.0$_{\pm 0.8}$	4.9$_{\pm 0.2}$	72.4$_{\pm 0.2}$	63.7$_{\pm 0.1}$
	ISE	45.0$_{\pm 1.3}$	55.3$_{\pm 2.1}$	77.2$_{\pm 1.5}$	71.6$_{\pm 1.6}$	18.5$_{\pm 0.6}$	4.3$_{\pm 0.3}$	61.5$_{\pm 1.5}$	<u>37.2</u>$_{\pm 1.7}$
	ASIDE	<u>28.3</u>$_{\pm 1.5}$	<u>43.3</u>$_{\pm 1.8}$	<u>56.6</u>$_{\pm 1.6}$	70.5$_{\pm 1.3}$	13.8$_{\pm 0.5}$	<u>1.3</u>$_{\pm 0.1}$	31.3$_{\pm 0.8}$	39.0$_{\pm 0.4}$
	DepthGate-Continuous	**19.8**$_{\pm 1.2}$	**41.4**$_{\pm 3.1}$	57.5$_{\pm 2.3}$	<u>52.5</u>$_{\pm 2.9}$	**0.1**$_{\pm 0.0}$	**0.4**$_{\pm 0.0}$	**11.9**$_{\pm 0.8}$	**35.5**$_{\pm 0.5}$
	DepthGate-Binary	29.5$_{\pm 1.1}$	53.3$_{\pm 1.9}$	**49.7**$_{\pm 2.1}$	**48.8**$_{\pm 0.0}$	<u>12.8</u>$_{\pm 0.6}$	3.4$_{\pm 0.9}$	<u>31.0</u>$_{\pm 0.4}$	50.3$_{\pm 0.6}$
Qwen3 8B	Original	56.0$_{\pm 1.9}$	63.6$_{\pm 2.7}$	80.6$_{\pm 0.8}$	59.3$_{\pm 0.3}$	19.3$_{\pm 0.3}$	11.1$_{\pm 0.8}$	60.5$_{\pm 0.8}$	60.4$_{\pm 0.9}$
	ISE	<u>35.1</u>$_{\pm 1.4}$	<u>32.7</u>$_{\pm 2.1}$	<u>68.4</u>$_{\pm 2.2}$	<u>58.6</u>$_{\pm 1.0}$	14.1$_{\pm 0.8}$	9.7$_{\pm 0.6}$	52.7$_{\pm 3.2}$	62.1$_{\pm 3.6}$
	ASIDE	37.6$_{\pm 0.0}$	34.0$_{\pm 2.4}$	70.1$_{\pm 3.7}$	60.2$_{\pm 1.1}$	11.5$_{\pm 1.7}$	<u>6.7</u>$_{\pm 0.7}$	<u>20.9</u>$_{\pm 0.8}$	<u>34.9</u>$_{\pm 0.3}$
	DepthGate-Continuous	**30.8**$_{\pm 1.7}$	**31.9**$_{\pm 2.0}$	68.8$_{\pm 2.1}$	**50.6**$_{\pm 1.2}$	**0.5**$_{\pm 0.2}$	**5.8**$_{\pm 0.7}$	**3.0**$_{\pm 0.5}$	**31.6**$_{\pm 0.7}$
	DepthGate-Binary	41.5$_{\pm 1.8}$	40.2$_{\pm 1.0}$	**64.5**$_{\pm 2.5}$	61.8$_{\pm 1.3}$	<u>7.3</u>$_{\pm 0.3}$	9.6$_{\pm 0.3}$	28.1$_{\pm 0.4}$	45.8$_{\pm 0.6}$

Table 1: **Prompt-injection robustness across five models.** Direct benchmarks place attacks in lower-trust prompt content, whereas indirect benchmarks embed attacks in external or structured data that the model must still use. Entries report mean ASR in percent with standard deviation over three seeds; lower is better. Boldface and underlining mark the best and second-best results within each model block.

Bai et al., 2023; Qwen et al., 2024; Yang et al., 2025). We compare Original, Instructional Segment Embedding (ISE), and ASIDE (Wu et al., 2025; Zverev et al., 2025). All methods use the same LoRA-based supervised post-training on 46,581 role-marked Alpaca-style examples (Hu et al., 2022); Original applies no role-specific representation transformation, while ISE and ASIDE use segment embeddings and an input-level data rotation, respectively. **DepthGate-Continuous** and **DepthGate-Binary** use real-valued and binary schedules selected by Schedule Discovery. Complete dataset statistics, optimization settings, and compute details are provided in the supplementary material.

Benchmarks and Metrics SEP uses paired prompts that place the same probe in either the instruction field or the data field. It reports how often the data-side probe is ignored among examples for which the instruction-side probe is followed; higher is better (Zverev et al., 2025). SEP Utility reports the corresponding instruction-side success rate, so the two metrics jointly characterize separation and retained instruction following. AlpacaEval evaluates general instruction following (Li et al., 2023b; Zheng et al., 2023). TensorTrust, Gandalf, Purple, and RuLES provide direct prompt-injection stress tests (Toyer et al., 2024; Pfister et al., 2025; Kim et al., 2024; Mu et al., 2023). BIPIA-Text, BIPIA-Code, StruQ-ID, and StruQ-OOD place attacks in external or structured data (Yi et al., 2025; Chen et al., 2025a). Prompt-injection robustness is measured by attack success rate (ASR), where lower is better.

Role Marking and Protocol Role indicators follow benchmark fields or the trusted channel hierarchy. In SEP, BIPIA, and StruQ, documents, code, and structured inputs form the data channel. In direct benchmarks, the protected rule or task is instruction and lower-trust prompt content is data. AlpacaEval has no data field, so all tokens are marked as instruction and steering is inactive; its score measures utility. The selection split is disjoint from final evaluation. SEP, SEP Utility, and AlpacaEval report standard error, while prompt-injection results report mean and standard deviation over three seeds.

Method	SEP↑	SEP Utility↑	AlpacaEval↑
Llama 2 7B			
Original	$71.0_{\pm 2.3}$	$\mathbf{37.9}_{\pm 1.5}$	$\underline{76.2}_{\pm 1.5}$
ISE	$70.4_{\pm 1.6}$	$\underline{36.2}_{\pm 1.4}$	$\mathbf{78.7}_{\pm 1.5}$
ASIDE	$84.7_{\pm 2.0}$	$32.0_{\pm 1.5}$	$59.3_{\pm 1.5}$
DepthGate-Continuous	$\mathbf{91.5}_{\pm 1.8}$	$28.6_{\pm 1.7}$	$56.6_{\pm 1.5}$
DepthGate-Binary	$\underline{88.9}_{\pm 1.7}$	$30.4_{\pm 1.5}$	$57.1_{\pm 1.5}$
Llama 2 13B			
Original	$55.5_{\pm 2.2}$	$\mathbf{59.3}_{\pm 1.6}$	$\underline{77.8}_{\pm 1.5}$
ISE	$53.2_{\pm 1.7}$	$52.8_{\pm 1.5}$	$\mathbf{80.4}_{\pm 1.5}$
ASIDE	$\underline{73.9}_{\pm 1.8}$	$\underline{57.1}_{\pm 1.6}$	$68.8_{\pm 1.5}$
DepthGate-Continuous	$\mathbf{88.7}_{\pm 1.9}$	$50.2_{\pm 1.7}$	$60.8_{\pm 1.5}$
DepthGate-Binary	$\underline{82.5}_{\pm 1.7}$	$53.0_{\pm 1.5}$	$66.0_{\pm 1.5}$
Llama 3.1 8B			
Original	$66.2_{\pm 1.8}$	$\mathbf{71.2}_{\pm 1.4}$	$\underline{84.1}_{\pm 1.2}$
ISE	$71.4_{\pm 1.4}$	$\underline{66.4}_{\pm 1.4}$	$\mathbf{86.0}_{\pm 1.2}$
ASIDE	$78.1_{\pm 1.7}$	$57.9_{\pm 1.6}$	$60.6_{\pm 1.2}$
DepthGate-Continuous	$\mathbf{89.6}_{\pm 1.8}$	$54.6_{\pm 1.6}$	$60.7_{\pm 1.2}$
DepthGate-Binary	$\underline{85.7}_{\pm 1.7}$	$55.2_{\pm 1.5}$	$64.1_{\pm 1.2}$
Qwen2.5 7B			
Original	$51.0_{\pm 2.0}$	$\mathbf{60.6}_{\pm 1.5}$	$\underline{83.3}_{\pm 1.3}$
ISE	$51.6_{\pm 1.5}$	$60.1_{\pm 1.4}$	$\mathbf{86.4}_{\pm 1.3}$
ASIDE	$67.4_{\pm 1.9}$	$\underline{60.4}_{\pm 1.5}$	$74.6_{\pm 1.3}$
DepthGate-Continuous	$\mathbf{92.1}_{\pm 1.3}$	$56.7_{\pm 1.6}$	$71.9_{\pm 1.3}$
DepthGate-Binary	$\underline{83.4}_{\pm 1.5}$	$58.5_{\pm 1.6}$	$78.9_{\pm 1.3}$
Qwen3 8B			
Original	$41.2_{\pm 1.8}$	$\mathbf{77.9}_{\pm 1.3}$	$\underline{79.6}_{\pm 1.4}$
ISE	$38.6_{\pm 1.6}$	$\underline{76.4}_{\pm 1.3}$	$\mathbf{82.0}_{\pm 1.4}$
ASIDE	$67.1_{\pm 1.7}$	$72.9_{\pm 1.4}$	$70.4_{\pm 1.4}$
DepthGate-Continuous	$\mathbf{86.2}_{\pm 1.8}$	$67.8_{\pm 1.5}$	$68.2_{\pm 1.4}$
DepthGate-Binary	$\underline{78.8}_{\pm 1.7}$	$70.4_{\pm 1.4}$	$73.1_{\pm 1.4}$

Table 2: **Instruction-data separation and utility.** SEP measures following an instruction-field probe while ignoring the same probe in data. SEP Utility and AlpacaEval measure retained utility. Values are percentages with standard error; bold and underlined values mark the best and second-best results per model.

Layer Setting	SEP↑	SEP Utility↑	Data-Probe Success ↓
Original	51.0	**60.6**	40.5
ASIDE (embedding only)	67.4	$\underline{60.4}$	23.8
Layer 10 only	80.3	58.8	14.5
Uniform ($g = 1$)	**97.4**	30.4	**1.2**
Uniform ($g = 0.5$)	$\underline{95.5}$	26.4	$\underline{3.2}$
Uniform ($g = 0.25$)	82.2	40.4	16.0
DepthGate-Continuous	92.1	56.7	5.5
DepthGate-Binary	83.4	58.5	13.4

Table 3: **Layer ablation on Qwen2.5 7B.** Uniform settings apply a fixed gate at every layer. Data-Probe Success measures probe execution in the data field. Values are percentages on the fixed SEP subset; bold and underlined values mark the best and second-best results.

Setting	SEP↑	SEP Utility↑	Data-Probe Success ↓
Original	51.0	**60.6**	40.5
DepthGate-Binary (matched)	$\underline{83.4}$	$\underline{58.5}$	13.4
DepthGate-Continuous (matched)	**92.1**	56.7	5.5
Original + post-hoc Binary	42.6	2.4	2.1
Original + post-hoc Continuous	64.2	1.2	**0.8**
ASIDE + post-hoc Binary	68.4	1.8	1.3
ASIDE + post-hoc Continuous	73.1	1.4	$\underline{1.1}$

Table 4: **Schedule matching on Qwen2.5 7B.** Matched schedules are used during both post-training and evaluation; post-hoc schedules are applied only during evaluation. Values are percentages on the fixed SEP subset; boldface and underlining mark the best and second-best values in each metric column.

MAIN RESULTS

Indirect Prompt-Injection Robustness The indirect attacks in Table 1 test malicious instructions carried by marked data fields. DepthGate-Continuous achieves the lowest ASR on BIPIA-Text, StruQ-ID, and StruQ-OOD for all five models, while DepthGate-Binary is generally the next strongest variant. These results show that the selected paths preserve the trusted role boundary when attacks occur in data that the model must still process. This directly evaluates the setting in which data must remain usable while embedded instructions are rejected.

Direct Prompt-Injection Robustness The direct attacks provide a complementary stress test on lower-trust prompt content. DepthGate-Continuous gives the lowest TensorTrust ASR across all five models and usually leads on Gandalf and RuLES, while DepthGate-Binary is strongest on Purple for most models. The gains are most consistent for attacks placed in explicitly marked data fields, the setting targeted by the role-conditioned mechanism. Together, the direct and indirect results distinguish general attack resistance from preservation of the marked data boundary.

Separation and Utility Across all five models, DepthGate improves conditional separation while providing two instruction-following operating points. Table 2 shows that DepthGate-Continuous ranks first in SEP and DepthGate-Binary ranks second for every model. Binary schedules retain higher SEP Utility and AlpacaEval, whereas continuous schedules prioritize separation. The two variants therefore expose the intended separation–utility operating range rather than a single fixed intervention strength.

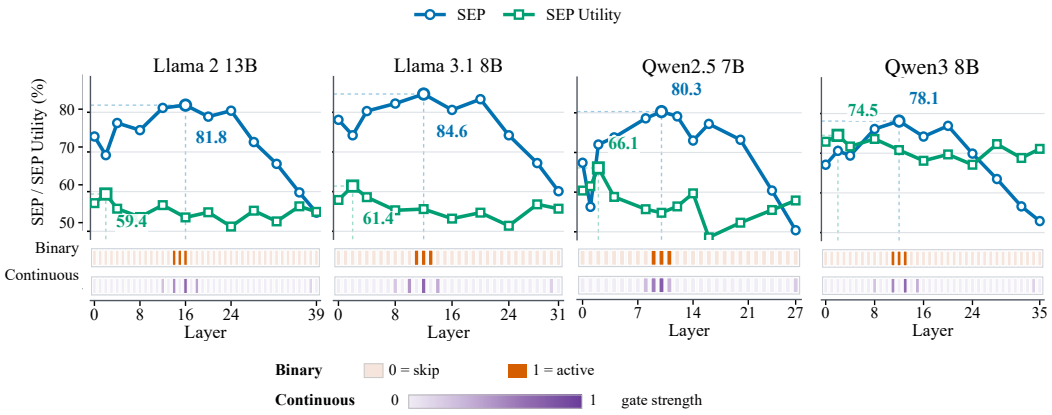


Figure 4: **Depth dependence and selected boundary-maintenance paths.** Each panel shows one model: the curves give the SEP and SEP Utility of a single full data-role rotation at each layer, with layer 0 denoting the embedding output. The binary and continuous strips below each plot show the model-specific paths selected by Schedule Discovery. The paths combine multiple depths instead of copying a single-layer optimum.

Analysis

Selected Paths Figure 4 compares single-layer profiles with the paths selected by Schedule Discovery. The profiles have different SEP and utility peaks across depth; on Qwen2.5 7B, layer 10 reaches 80.3% SEP versus 67.4% at the embedding output. The selected gates therefore span multiple depths rather than copying one layer. This depth variation motivates selecting a path instead of applying the same intervention uniformly.

Layer Ablation Table 3 compares embedding-only, single-layer, uniform, and selected paths. The uniform rows expose a separation–utility trade-off: stronger gates suppress data-side probe execution but reduce SEP Utility. The selected paths improve separation over the input-only baseline while retaining more utility than uniform intervention. The comparison isolates intervention location and strength under a common Qwen2.5 protocol.

Schedule Matching Table 4 tests whether the selected path is included during post-training or added only at evaluation. Matched schedules use the same path in both stages, whereas post-hoc controls apply it to adapters trained without that path. The matched variants retain useful SEP Utility; post-hoc insertion sharply lowers utility despite reducing Data-Probe Success, supporting path-matched adaptation. This control separates the benefit of the path from the benefit of adapting parameters to that path.

Cross-Model Consistency Across all five models, continuous schedules favor SEP while binary schedules retain more utility. This ordering is preserved across the Llama and Qwen families, indicating that Schedule Discovery selects model-specific placements and strengths.

6 Conclusion

DepthGate carries the trusted instruction-data boundary from input encoding into depth-wise hidden-state computation. Role-Conditioned Steering reinforces it, and Schedule Discovery identifies each model's Depth-Aware Path. Across five models, DepthGate improves SEP and marked-data robustness while providing complementary separation–utility operating points. The selected paths reinforce the boundary at later depths, and schedule matching confirms that the adapted path can be reused without online search at inference.

AI USE STATEMENT

Generative AI tools were used to assist with research ideation and execution, including developing and revising research code, supporting experimental implementation, and assisting with the creation or modification of scientific figures. Generative AI tools were also used to assist with drafting sections of the manuscript, improving language and readability, organizing the manuscript, and formatting the paper. The authors reviewed, tested, and revised all AI-assisted code, experimental procedures, figures, and text, verified the reported results, and take responsibility for the final content of this work, including its methods, experiments, claims, and artifacts.

REFERENCES

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022a.

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*, 2022b.

Nora Belrose, Igor Ostrovsky, Lev McKinney, Zach Furman, Logan Smith, Danny Halawi, Stella Biderman, and Jacob Steinhardt. Eliciting latent predictions from transformers with the tuned lens. *arXiv preprint arXiv:2303.08112*, 2023.

Patrick Chao, Edoardo Debenedetti, Alexander Robey, Maksym Andriushchenko, Francesco Croce, Vikash Sehwag, Edgar Dobriban, Nicolas Flammarion, George J Pappas, Florian Tramer, et al. Jailbreakbench: An open robustness benchmark for jailbreaking large language models. *Advances in Neural Information Processing Systems*, 37:55005–55029, 2024.

Sizhe Chen, Julien Piet, Chawin Sitawarin, and David Wagner. {StruQ}: Defending against prompt injection with structured queries. In *34th USENIX Security Symposium (USENIX Security 25)*, pp. 2383–2400, 2025a.

Sizhe Chen, Arman Zharmagambetov, Saeed Mahloujifar, Kamalika Chaudhuri, David Wagner, and Chuan Guo. Secalign: Defending against prompt injection with preference optimization. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*, pp. 2833–2847, 2025b.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8493–8502, 2022.

Nicola De Cao, Wilker Aziz, and Ivan Titov. Editing factual knowledge in language models. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pp. 6491–6506, 2021.

Edoardo Debenedetti, Jie Zhang, Mislav Balunovic, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. Agentdojo: A dynamic environment to evaluate prompt injection attacks and defenses for llm agents. *Advances in Neural Information Processing Systems*, 37:82895–82920, 2024.

Edoardo Debenedetti, Ilia Shumailov, Tianqi Fan, Jamie Hayes, Nicholas Carlini, Daniel Fabian, Christoph Kern, Chongyang Shi, Andreas Terzis, and Florian Tramèr. Defeating prompt injections by design. *arXiv preprint arXiv:2503.18813*, 2025.

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

Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not what you've signed up for: Compromising real-world llm-integrated applications with indirect prompt injection. In *Proceedings of the 16th ACM workshop on artificial intelligence and security*, pp. 79–90, 2023.

Keegan Hines, Gary Lopez, Matthew Hall, Federico Zarfati, Yonatan Zunger, and Emre Kiciman. Defending against indirect prompt injection attacks with spotlighting. *arXiv preprint arXiv:2403.14720*, 2024.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3, 2022.

Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, et al. Llama guard: Llm-based input-output safeguard for human-ai conversations. *arXiv preprint arXiv:2312.06674*, 2023.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. What does bert learn about the structure of language? In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 3651–3657, 2019.

Taeyoun Kim, Suhas Kotha, and Aditi Raghunathan. Testing the limits of jailbreaking defenses with the purple problem. *arXiv preprint arXiv:2403.14725*, 2024.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33: 9459–9474, 2020.

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

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. Alpacaeval: An automatic evaluator of instruction-following models, 2023b.

Zekun Li, Baolin Peng, Pengcheng He, and Xifeng Yan. Evaluating the instruction-following robustness of large language models to prompt injection. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 557–568, 2024.

Yupei Liu, Yuqi Jia, Runpeng Geng, Jinyuan Jia, and Neil Zhenqiang Gong. Formalizing and benchmarking prompt injection attacks and defenses. In *33rd USENIX Security Symposium (USENIX Security 24)*, pp. 1831–1847, 2024.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *Advances in neural information processing systems*, 35:17359–17372, 2022a.

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. *arXiv preprint arXiv:2210.07229*, 2022b.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. Fast model editing at scale. *arXiv preprint arXiv:2110.11309*, 2021.

Norman Mu, Sarah Chen, Zifan Wang, Sizhe Chen, David Karamardian, Lulwa Aljeraisy, Basel Alomair, Dan Hendrycks, and David Wagner. Can llms follow simple rules? *arXiv preprint arXiv:2311.04235*, 2023.

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.

Nina Panickssery, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Matt Turner. Steering llama 2 via contrastive activation addition. *arXiv preprint arXiv:2312.06681*, 2023.

Fábio Perez and Ian Ribeiro. Ignore previous prompt: Attack techniques for language models. *arXiv preprint arXiv:2211.09527*, 2022.

Niklas Pfister, Václav Volhejn, Manuel Knott, Santiago Arias, Julia Bazińska, Mykhailo Bichurin, Alan Commike, Janet Darling, Peter Dienes, Matthew Fiedler, et al. Gandalf the red: Adaptive security for llms. *arXiv preprint arXiv:2501.07927*, 2025.

Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, and Fei Huang. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

Alexander Robey, Eric Wong, Hamed Hassani, and George J Pappas. Smoothllm: Defending large language models against jailbreaking attacks. *arXiv preprint arXiv:2310.03684*, 2023.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. A primer in bertology: What we know about how bert works. *Transactions of the association for computational linguistics*, 8:842–866, 2020.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551, 2023.

Sander Schulhoff, Jeremy Pinto, Anaum Khan, Louis-François Bouchard, Chenglei Si, Svetlina Anati, Valen Tagliabue, Anson Kost, Christopher Carnahan, and Jordan Lee Boyd-Graber. Ignore this title and hackaprompt: Exposing systemic vulnerabilities of llms through a global prompt hacking competition. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 4945–4977, 2023.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. Bert rediscovers the classical nlp pipeline. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 4593–4601, 2019.

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

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.

Sam Toyer, Olivia Watkins, Ethan Mendes, Justin Svegliato, Luke Bailey, Tiffany Wang, Isaac Ong, Karim Elmaaroufi, Pieter Abbeel, Alan Ritter, et al. Tensor trust: Interpretable prompt injection attacks from an online game. In *International Conference on Learning Representations*, volume 2024, pp. 18714–18746, 2024.

Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Ulisse Mini, and Monte MacDiarmid. Activation addition: Steering language models without optimization. *arXiv preprint arXiv:2308.10248*, 2024.

Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. Jailbroken: How does llm safety training fail? *Advances in neural information processing systems*, 36:80079–80110, 2023.

Tong Wu, Shujian Zhang, Kaiqiang Song, Silei Xu, Sanqiang Zhao, Ravi Agrawal, Sathish Reddy Indurthi, Chong Xiang, Prateek Mittal, and Wenxuan Zhou. Instructional segment embedding: Improving llm safety with instruction hierarchy. In *International Conference on Learning Representations*, volume 2025, pp. 93728–93758, 2025.

Zhengxuan Wu, Aryaman Arora, Zheng Wang, Atticus Geiger, Dan Jurafsky, Christopher D Manning, and Christopher Potts. Reft: Representation finetuning for language models. *Advances in Neural Information Processing Systems*, 37:63908–63962, 2024.

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.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Jingwei Yi, Yueqi Xie, Bin Zhu, Emre Kiciman, Guangzhong Sun, Xing Xie, and Fangzhao Wu. Benchmarking and defending against indirect prompt injection attacks on large language models. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*, pp. 1809–1820, 2025.

Qiusi Zhan, Zhixiang Liang, Zifan Ying, and Daniel Kang. Injecagent: Benchmarking indirect prompt injections in tool-integrated large language model agents. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 10471–10506, 2024.

Wei Zhao, Zhe Li, Yige Li, Ye Zhang, and Jun Sun. Defending large language models against jailbreak attacks via layer-specific editing. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 5094–5109, 2024.

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.

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

Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*, 2023b.

Egor Zverev, Evgenii Kortukov, Alexander Panfilov, Alexandra Volkova, Soroush Tabesh, Sebastian Lapuschkin, Wojciech Samek, and Christoph H Lampert. Aside: Architectural separation of instructions and data in language models. *arXiv preprint arXiv:2503.10566*, 2025.

A Additional Experiments and Analysis

This section complements the main analysis with controls over depth placement, schedule matching, and role conditioning. Unless stated otherwise, all experiments use Qwen2.5 7B and the SEP protocol from the main paper.

Depth Placement and Scope

Table 5 extends the main depth ablation with fixed-strength uniform settings and the Same-Output Rate diagnostic.

Layer Setting	SEP↑	SEP Utility↑	Data Probe Success↓	Same Output Rate↓
Original	51.0	**60.6**	40.5	58.3
ASIDE (embedding only)	67.4	<u>60.4</u>	23.8	55.2
Layer 10 only	80.3	58.8	14.5	<u>52.3</u>
Uniform ($g = 1$)	**97.4**	30.4	**1.2**	70.0
Uniform ($g = 0.5$)	<u>95.5</u>	26.4	<u>3.2</u>	68.0
Uniform ($g = 0.25$)	82.2	40.4	16.0	56.5
DepthGate-Continuous	92.1	56.7	5.5	56.0
DepthGate-Binary	83.4	58.5	13.4	**46.9**

Table 5: Extended layer ablation on Qwen2.5 7B. Uniform settings apply the listed fixed gate at every layer. Data-Probe Success is the execution rate of a probe placed in the data field. Same-Output Rate is the percentage of paired SEP examples for which the model generates exactly the same decoded response after the probe is moved between the instruction and data fields. All settings use the same data, optimization, and evaluation protocol, with each adapter trained under its corresponding layerwise intervention configuration. Values are percentages on the fixed SEP evaluation subset; boldface and underlining mark the best and second-best values in each metric column.

Relative to uniform $g = 0.25$, the stronger settings reduce Data-Probe Success from 16.0% to 3.2% and 1.2%, but also reduce SEP Utility from 40.4% to 26.4% and 30.4%, and raise Same-Output Rate from 56.5% to 68.0% and 70.0%. The selected schedules provide complementary operating points: DepthGate-Binary retains 58.5% utility, while DepthGate-Continuous reaches 56.7% utility with stronger separation. This comparison highlights the value of selecting a depth-wise path rather than applying one fixed gate uniformly.

Schedule Matching

Schedule matching tests whether a selected path remains effective without path-matched adaptation. Tables 6 and 7 compare matched DepthGate schedule–adapter pairs with post-hoc variants that apply the selected binary or continuous schedule to Original or ASIDE adapters only during evaluation.

Setting	SEP Utility↑	TensorTrust↓	Gandalf↓	Purple↓	RuLES↓	BIPIA Text ↓	BIPIA Code ↓
Original	**60.6**	45.3	58.8	78.4	65.3	18.0	4.9
DepthGate-Binary (matched)	<u>58.5</u>	29.5	53.3	49.7	**48.8**	12.8	3.4
DepthGate-Continuous (matched)	56.7	19.8	41.4	57.5	52.5	**0.1**	<u>0.4</u>
Original + post-hoc DepthGate-Binary	2.4	0.8	1.1	**0.7**	<u>52.4</u>	0.6	<u>0.2</u>
Original + post-hoc DepthGate-Continuous	1.2	<u>0.5</u>	<u>0.7</u>	1.0	53.1	<u>0.4</u>	**0.1**
ASIDE + post-hoc DepthGate-Binary	1.8	0.6	0.9	1.8	52.8	0.9	<u>0.2</u>
ASIDE + post-hoc DepthGate-Continuous	1.4	**0.4**	**0.4**	<u>0.8</u>	52.5	0.5	<u>0.2</u>

Table 7: **Utility-aware prompt-injection controls.** SEP Utility is shown before ASR to distinguish usable defenses from low-ASR settings that no longer follow trusted instructions. Post-hoc variants reduce ASR on most benchmarks but retain only **1.2–2.4% SEP Utility**, whereas matched schedule–adapter pairs retain substantially higher utility. All values are percentages; boldface and underlining mark the best and second-best values in each metric column.

Setting	SEP↑	Data-Probe Success ↓	SEP Utility↑
Original	51.0	40.5	**60.6**
DepthGate (matched)			
Binary	<u>83.4</u>	13.4	<u>58.5</u>
Continuous	**92.1**	5.5	56.7
Post-hoc on Original			
Binary	42.6	2.1	2.4
Continuous	64.2	**0.8**	1.2
Post-hoc on ASIDE			
Binary	68.4	1.3	1.8
Continuous	73.1	<u>1.1</u>	1.4

Table 6: Schedule-matching controls on Qwen2.5 7B. Post-hoc schedules are applied only during evaluation to adapters trained under a different intervention configuration. Matched settings use the same fixed schedule throughout post-training and evaluation. Values are percentages on the fixed SEP evaluation subset; boldface and underlining mark the best and second-best values in each metric column.

Utility distinguishes effective defense from degenerate behavior. Post-hoc insertion reduces SEP Utility to 1.2–2.4%, whereas matched DepthGate-Binary and DepthGate-Continuous retain 58.5% and 56.7% SEP Utility. These controls support matching model adaptation to the schedule used at evaluation.

Role-Conditioned Steering Controls

Role-Conditioned Steering is asymmetric by design: each local operation leaves instruction states unchanged and redirects only marked data states. Table 8 tests this design against Original, which steers neither role. Every non-Original control uses the same fixed intervention at layer 0, the embedding output, with identical post-training and evaluation settings; only the roles eligible for steering change.

Using the same embedding-only data transformation as ASIDE raises SEP from 51.0% to 67.4% and reduces Data-Probe Success from 40.5% to 23.8%. The corresponding Instruction-Probe Success and Same-Output Rate are 60.4% and 55.2%. Steering instruction states alone does not produce the same separation, while steering both roles lowers Instruction-Probe Success to 59.1%. These controls support applying Role-Conditioned Steering to marked data states while leaving instruction states unchanged.

Steering Setting	SEP↑	Data-Probe Success ↓	Instruction-Probe Success ↑	Same-Output Rate ↓
Original (neither role)	$51.0_{\pm 2.0}$	$40.5_{\pm 1.6}$	$\underline{60.6}_{\pm 1.5}$	$\underline{58.3}_{\pm 1.6}$
ASIDE (embedding only)	**67.4**	**23.8**	60.4	**55.2**
Instruction only	$35.5_{\pm 1.9}$	$53.6_{\pm 1.6}$	$\mathbf{63.4}_{\pm 1.5}$	$64.8_{\pm 1.5}$
Both roles	$\underline{53.8}_{\pm 2.1}$	$\underline{34.3}_{\pm 1.5}$	$59.1_{\pm 1.6}$	$61.2_{\pm 1.5}$

Table 8: Role-Conditioned Steering controls at layer 0, the embedding output. "ASIDE (embedding only)" applies the same data-only input transformation as the main-table ASIDE baseline; the other rows change which roles are eligible for the fixed intervention. Data-Probe and Instruction-Probe Success measure probe execution in the corresponding input field. Values are percentages; boldface and underlining mark the best and second-best values in each metric column.

B Schedule Discovery Details

Schedule Discovery runs once per reported model before final post-training and benchmark evaluation. It uses the held-out selection split described in Table 10; no final benchmark examples are used to score, construct, or select schedules. The scoring rule, candidate templates and budget, and health constraints are fixed across all reported models before final benchmark evaluation. Layer 0 is the embedding output, and layer $l \geq 1$ is the output of Transformer block l.

The search uses a fixed budget of candidate templates balanced across the continuous and binary families. The templates and construction rules are established before final benchmark evaluation. Held-out layer scores instantiate their placements and strengths, and the candidate-selection score determines the final schedules. A radius-1 window activates a top-ranked layer and its immediate neighboring layers within the valid range $0, \ldots, L$.

Procedure. Let $\mathbf{e}^{(l)}$ denote the single-layer schedule with $e_k^{(l)} = \mathbb{1}[k = l]$, where $\mathbb{1}$ is the indicator function. At every layer, the proxy stage computes

$$\begin{aligned} S_{\text{proxy}}\left(\mathbf{e}^{(l)}\right) = \text{SEP}_{\text{p}}\left(\mathbf{e}^{(l)}\right) + 0.50 U_{\text{p}}\left(\mathbf{e}^{(l)}\right) \\ - 0.35 A_{\text{data,p}}\left(\mathbf{e}^{(l)}\right) \\ - 0.15 R_{\text{same,p}}\left(\mathbf{e}^{(l)}\right). \end{aligned} \tag{9}$$

Here SEP_{p} measures proxy separation, U_{p} measures instruction-side probe success, $A_{\text{data,p}}$ measures data-side probe execution, and $R_{\text{same,p}}$ measures exact response agreement across the paired role placements. Each term is a rate in $[0, 1]$. The score gives separation the primary weight, rewards instruction-side execution, and penalizes data-side execution and role-insensitive same-output behavior. We define the layer score as $s_l = S_{\text{proxy}}\left(\mathbf{e}^{(l)}\right)$, and use the resulting depth profile only to rank and weight candidates. Let $l_{(1)}, l_{(2)}, \ldots$ be layers sorted by decreasing s_l, and let

Stage	Configuration
Layer proxy	All $L + 1$ layers; 64 held-out SEP examples per layer.
Continuous pool	Single-layer candidates and top-k schedules with $k \in \{3, 5\}$, $\gamma \in \{0.25, 0.50\}$, and minimum retained gate $\tau = 0.03$.
Binary pool	Single-layer candidates, top-k schedules with $k \in \{1, 2, 3, 4\}$, and radius-1 windows around top-ranked peaks.
Candidate cap	At most 12 deduplicated schedules, balanced across continuous and binary families.
Short adaptation	120 LoRA optimization steps for each fixed candidate schedule.
Candidate check	192 held-out SEP examples with the score in Eq. 11.
Health constraints	$U_{\text{sel}} \geq 0.30$, $A_{\text{data,sel}} \leq 0.60$, and $R_{\text{same,sel}} \leq 0.82$.

Table 9: Schedule Discovery configuration. Proxy and candidate examples are drawn only from the held-out selection split, and each candidate is evaluated after schedule-matched short adaptation.

Dataset or benchmark	Count	Use in this work
Alpaca-style training data	46,581 train / 5,175 validation	Supervised post-training for all trained variants.
SEP dataset	9,160 examples	Final separation and SEP utility evaluation; reported runs use a fixed 1,000-example evaluation cap. SEP uncertainty is standard error over evaluated examples.
AlpacaEval	805 prompts	Utility evaluation with AlpacaEval judge-reported standard error.
TensorTrust, Gandalf, Purple, RuLES	165, 140, 285, and 285 prompts	Direct prompt-injection evaluation over three seeds.
BIPIA-Text and BIPIA-Code	12,276 and 6,000 generated examples	Indirect prompt-injection evaluation over three seeds; reported runs use a fixed 1,000-example cap per attack type and seed.
StruQ	208 base examples	Indirect prompt-injection evaluation over two domains, four attacks, and three seeds; results aggregate attack success across domain-attack combinations.
Schedule Discovery	892 examples	SEP held-out selection split for model-specific schedule selection only; proxy and candidate checks use fixed subsets of 64 and 192 examples respectively, and final benchmark data is not used to tune schedules.

Table 10: Data and evaluation statistics. Counts describe the training, schedule-selection, utility, separation, and prompt-injection evaluation data used in the paper. When an evaluation is capped for computational tractability, the cap is fixed before model comparison and applied consistently across methods.

$\mathcal{I}_k = \{l_{(1)}, \ldots, l_{(k)}\}$. With $m_k = \max_{j \in \mathcal{I}_k} \max(s_j, 0)$, continuous schedules use normalized weights

$$\bar{s}_l = \begin{cases} \dfrac{\max(s_l, 0)}{m_k}, & m_k > 0, \\ 1, & m_k = 0. \end{cases} \tag{10}$$

Equation 10 preserves the relative positive proxy scores within the selected layers. If $m_k = 0$, equal unit weights provide a fallback. The retained gate is $g_l = \gamma \bar{s}_l$ when $l \in \mathcal{I}_k$ and $\gamma \bar{s}_l \geq \tau$, and is zero otherwise. Binary schedules instead set the selected gates to one. Each candidate remains fixed from the first step of a fresh short LoRA adaptation.

Selection and Reuse. For each candidate $\mathbf{g}$ and its matching short-trained parameters $\phi_{\mathbf{g}}$, selection uses

$$\begin{aligned} S_{\text{sel}}(\mathbf{g}, \phi_{\mathbf{g}}) = \text{SEP}_{\text{sel}}(\mathbf{g}, \phi_{\mathbf{g}}) &+ 0.50 U_{\text{sel}}(\mathbf{g}, \phi_{\mathbf{g}}) \\ &- 0.35 A_{\text{data,sel}}(\mathbf{g}, \phi_{\mathbf{g}}) \\ &- 0.15 R_{\text{same,sel}}(\mathbf{g}, \phi_{\mathbf{g}}). \end{aligned} \tag{11}$$

The terms are rates in $[0, 1]$ with the same roles as their proxy counterparts in Eq. 9, but are computed on the candidate-selection examples after schedule-matched adaptation. The health constraints in Table 9 first remove degenerate candidates, and the score ranks the remaining candidates. The best valid continuous and binary schedules are selected separately and frozen before final post-training. Discovery is not repeated during benchmark evaluation or inference, and final test examples do not influence the selected schedules.

C Role Marking Details

DepthGate uses source-role labels supplied by the input format or known channel hierarchy. Trusted system or task instructions receive $r_i = 0$. External content that should support the task without controlling the response receives $r_i = 1$. This category includes retrieved documents, user-provided passages, code snippets, tables, emails, and structured data fields. The label records the source role rather than predicting whether an individual token is malicious.

Item	Setting
Model sizes	Llama 2 7B, Llama 2 13B, Llama 3.1 8B, Qwen2.5 7B, and Qwen3 8B.
Post-training	Supervised post-training with LoRA using rank 16, alpha 32, dropout 0.05, no bias adaptation, 3 epochs, learning rate 2×10^{-4}, cosine schedule, no warmup, weight decay 0, maximum gradient norm 1.0, per-device batch size 1, gradient accumulation 8, and maximum sequence length 768.
Numerical precision	FP16 training and inference on V100 GPUs, with activation checkpointing enabled for memory-sensitive runs.
Role-Conditioned Steering	Fixed quarter-turn base rotation with angle $\pi/2$, right-side application, and no learned rotation. Layerwise gates control the effective rotation angle.
Schedule Discovery	The schedule-matched configuration is reported in Table 9. Selected schedules are fixed before final post-training and benchmark evaluation.
Generation	SEP, AlpacaEval, and StruQ use deterministic decoding. Direct prompt-injection and BIPIA evaluations follow the benchmark-provided stochastic decoding configuration and use three random seeds.
Software	PyTorch 2.6.0, Transformers 4.52.4, Datasets 4.8.3, PEFT 0.14.0, TRL 0.13.0, Accelerate 1.7.0, DeepSpeed 0.17.0, AlpacaEval 0.6.6, NumPy 2.2.6, pandas 2.2.3, and SciPy 1.14.1.
Hardware and budget	Experiments were run on NVIDIA Tesla V100S-PCIE-32GB GPUs. Completed final runs for the five reported models used approximately 700 single-GPU hours; this estimate excludes exploratory runs and external API judge latency.

Table 11: Implementation and compute details. The table summarizes model sizes, LoRA post-training, Role-Conditioned Steering, Schedule Discovery, generation settings, software versions, and approximate compute. All DepthGate schedules are selected before final benchmark evaluation and then reused during post-training and inference.

For SEP, BIPIA, and StruQ, role labels follow the instruction and data fields provided by each benchmark. Across TensorTrust, Gandalf, Purple, and RuLES, the benchmark-defined protected rule or task is assigned to the instruction field, while the lower-trust user or input string is assigned to the data field. This assignment uses source provenance available before inference, remains fixed across all compared methods, and does not inspect attack content or evaluation outcomes. AlpacaEval has no separate data field, so all prompt tokens are marked as instruction and Role-Conditioned Steering remains inactive during evaluation. Input role labels are applied after prompt formatting and remain fixed for the formatted prompt.

D Reproducibility Details

Data and Evaluation Statistics

Table 10 reports dataset sizes and evaluation counts for the main experiments. When an evaluation is capped for computational tractability, the cap is fixed before model comparison and applied consistently across methods.

Implementation and Compute Details

Table 11 summarizes the training and evaluation configuration. All final DepthGate schedules are selected before benchmark evaluation and loaded consistently during final post-training and inference.