SAFECOMPOSE: BRANCH-PRESERVING SAFETY COMPOSITION FOR HELD-OUT TASK ADAPTERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Parameter-efficient fine-tuning delivers new capability as a small adapter that leaves the base model untouched, so the party that trains an adapter is often not the party that deploys it. The deploying party, which we call the operator, receives finished adapter weights but nothing of the training run behind them. If that run included unsafe examples, composing the adapter with an aligned base model can erode the model's refusal of unsafe requests. We ask whether a defense can restore refusal without giving up the adapter's capability, even for adapters it has never seen. We introduce SafeCompose: the aligned base, the received adapter, and a safety adapter stay frozen and separate, and small per-layer gates read the input to set how much each adapter contributes. The gates are trained once on a pool of task adapters and reused unchanged on adapters that arrive later. We test on new adapters across model families and levels of unsafe training data, including a task the gates never saw, and also over a sequence of arriving adapters and under attack. On the unseen task, SafeCompose restores refusal and keeps more task capability than weight-projection and activation-steering defenses, retaining the 9–31 points of accuracy that discarding the adapter would give up. Over successive adapter arrivals, it stays safe where steering defenses do not. When the response is forced to begin by agreeing, even by an attacker who sees the gates, SafeCompose is safer than fixed gates trained the same way. It keeps this advantage even when its gates read the prompt only once, which allows cached decoding.

1 INTRODUCTION

Parameter-efficient fine-tuning adapts a language model by attaching a small adapter and leaving the base weights fixed (Houlsby et al., 2019; Hu et al., 2022). That separation makes adapters portable. As a side effect, however, it can also let a trained adapter weaken the safety behavior of a trusted base model it never modifies (Qi et al., 2024).

We study the party that deploys an aligned model, which we call the *operator*. The operator owns an aligned base model and a safety adapter, an adapter trained to strengthen refusal. A downstream party trains a task adapter, which adds a new capability, using data and a training procedure unavailable to the operator, and hands over the finished weights. The training data may contain unsafe examples; we call the resulting adapter *contaminated*, and even a small fraction of unsafe examples can erode the model's refusal of unsafe requests (Qi et al., 2024). Yet the operator can act only on the finished weights, ruling out any defense that intervenes during fine-tuning or requires access to the training run. The operator is therefore left, by default, with two poor options: ship the adapter and risk the model responding to unsafe requests, or discard it and lose the capability it adds.

Two lines of prior work address this issue. Composition methods (Pfeiffer et al., 2021; Wang et al., 2024a; Buehler & Buehler, 2024; Wang et al., 2024c) and routing methods (Shen, 2026; Wang et al., 2026) learn to combine or gate a set of adapters at inference time. They train the router on a fixed set of adapters and cannot take in an adapter that arrives later. Post-hoc safety methods, by contrast, restore refusal after the adapter arrives, by rewriting its weights or by adding a safety module to the merged model (Arazzi et al., 2026; Li et al., 2026; Aliev et al., 2026). Either way, the operator can no longer switch the adapter off or adjust how strongly it acts. An operator that keeps receiving adapters it did not train needs both properties at once. It needs a composition policy, a rule for how much each

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.

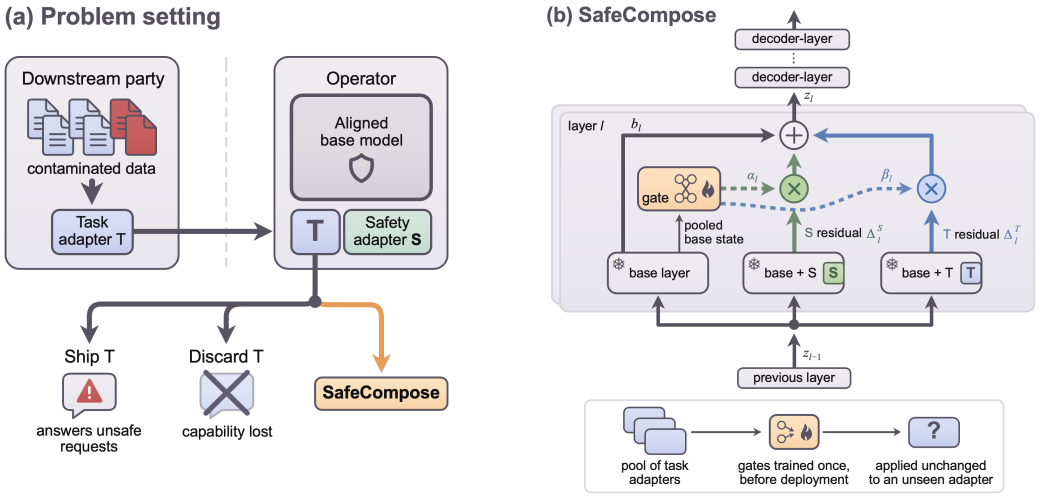

Figure 1: (a) The operator receives a finished task adapter T, possibly trained on contaminated data, by default it can only ship T or discard it. (b) SafeCompose keeps the base model, the safety adapter S, and T frozen and separate. At each layer l, a small trained gate reads the pooled base state and sets α_l and β_l, which scale the residuals Δ_l^S and Δ_l^T, each measured from the base output b_l, before they are added back to it (Eq. 1). The gates are trained once on a pool of task adapters and applied unchanged to unseen adapters.

Table 1: Structural comparison drawn from the cited primary descriptions. Adapter composition: AdapterFusion, LoRA-Flow, X-LoRA; safety routing: DR2, CLEAR. ✓ satisfied; ◐ partially or conditionally; ✗ not satisfied; – not applicable. Criteria and the reason for each ◐ are discussed in Appendix D.

	Adapter composition	Safety routing	NeWTral	TRACE	HyperSafe	**SafeCompose**
Task adapter unseen at policy training	✗	✗	✓	◐	◐	✓
No target-specific adaptation	✗	–	✓	✓	◐	✓
Received adapter unchanged	✓	–	✗	✗	◐	✓
Separate task and safety branches	✗	◐	✗	✗	✗	✓
Independent control of both branches	✗	✗	✗	✗	✗	✓

adapter contributes, that is learned in advance, applies unchanged to each new adapter, and leaves every component frozen and separate.

We propose SafeCompose, which treats an adapter as a push: given an incoming hidden state, the adapter pushes the layer's output away from what the base model alone would produce, and we call that push its activation residual. The two adapters' pushes start from the same base output but may point in different directions, one toward refusal and one toward a domain answer, and they collide at particular layers and for particular prompts. A common remedy is to merge the two adapters into one set of weights. A merge, however, sets their balance once, globally or per layer, before any input arrives, so a harmful prompt and a benign one get the same balance even though they may need different ones. SafeCompose instead sets the balance for each input (Section 3): small per-layer gates read the incoming state and scale each push before it is added to the base output.

The gates, which we call the composer, are the only part we train: the base model and both adapters stay frozen, and we train the gates once, on a pool of task adapters the operator holds before deployment. Because the gates see an adapter only through its residuals, they should learn how safety and task residuals interact in general, not features of the particular adapters they were trained on. So the same gates should also work for unseen adapters without retraining, a claim we call the transfer hypothesis.

We test the hypothesis on adapters trained after the gates were frozen, for a task excluded from the gates' training (MedMCQA (Pal et al., 2022)). On these adapters, SafeCompose restores refusal and keeps more of the task capability than existing weight-projection and activation-steering defenses across model families, retaining the 9–31 points of MedMCQA accuracy that discarding the adapter would give up (Section 4). Over five arriving adapters, it answers fewer than 1% of harmful requests while the steering defenses end at 17–30%. Without an attack, gates trained the same way that never see the prompt do just as well, but when the response is forced to begin by agreeing to the request they are clearly less safe, even against an attacker who sees the gates. The learned gate keeps this advantage even when it reads the prompt only once, which allows cached decoding (Section 5).

Our contributions are as follows.

1. **Gates learned before the adapter exists.** We introduce SafeCompose, to our knowledge the first composition policy learned before the task adapter exists and applied to it unchanged. Small per-layer gates compose the base model and both adapters, all kept frozen and separate.
2. **Transfer to unseen adapters.** On adapters for a new task, SafeCompose restores refusal and keeps more task capability than weight-projection and activation-steering defenses across model families, and, unlike the steering defenses, stays safe even as adapters keep arriving.
3. **An account of what the gate adds.** Compared to fixed gates trained the same way, reading the prompt adds nothing without an attack but makes the gate safer under attack, even when it is read only once, which is what makes cached decoding possible.

2 Related work

Composing frozen adapters. AdapterFusion (Pfeiffer et al., 2021) showed that two adapters trained apart can be combined without harming either. LoRA-Flow (Wang et al., 2024a), X-LoRA (Buehler & Buehler, 2024), and instance-level dynamic composition (Wang et al., 2024c) then made the combination learned and input-dependent. SafeCompose keeps both properties. It differs from them in two ways. They combine only the adapters present when their router was trained, whereas SafeCompose applies its gates unchanged to an adapter that arrives later. They also treat every adapter as a source of capability, whereas SafeCompose pairs the received adapter with one dedicated to safety.

Recovering safety after adaptation. Fine-tuning erodes safety (Qi et al., 2024), and existing defenses act at one of three stages: during training, on the finished weights, or at inference. Training-time methods act before or during task training: Vaccine (Huang et al., 2024) and RepNoise (Rosati et al., 2024) harden the base model beforehand, and SafeGrad (Yi et al., 2025) removes the part of the task gradient that conflicts with a safety objective; none can act on an adapter that already exists. Weight-space methods constrain or merge finished updates: SaLoRA (Li et al., 2025a), SafeMERGE (Djuhera et al., 2025), and TIES-Merging (Yadav et al., 2023) each produce one set of weights, and RESTA (Bhardwaj et al., 2024) adds to the fine-tuned weights a safety vector, the difference between the aligned model and a deliberately unaligned copy of it; in every case task and safety can no longer be adjusted separately. Inference-time methods such as CAST (Lee et al., 2025) and SafetyLock (Zhu et al., 2024) leave the weights alone, but each is configured per model, owns no separate safety adapter, and has no policy that carries over to the next adapter.

Safety composition and post-hoc repair. DR2 (Shen, 2026) and CLEAR (Wang et al., 2026) route or rescale safety LoRAs at inference time, but over modules known when the router was trained, and with no received task branch. Closest to our setting are systems built for adapters the defender did not train: NeWTral (Arazzi et al., 2026) translates the received weights, TRACE (Li et al., 2026) merges in a learned safety patch, and HyperSafe (Aliev et al., 2026) fits a side network to the one checkpoint. Each accepts an adapter that arrives after training. None keeps that adapter and the safety adapter as separate frozen branches under a policy learned before the adapter existed. Table 1 records these distinctions; Appendix D states the criteria.

Repeated adaptation and robust objectives. Multi-round red teaming in MART (Ge et al., 2024) and SC-Safety (Xu et al., 2023), and studies of safety under continual adaptation (Alssum et al.,

2025; Yun et al., 2025), show safety degrading over successive interventions. Our repeated-arrival protocol follows the same pattern with task adapters in place of attacks. Our objective adopts variance-penalized risk extrapolation from V-REx (Krueger et al., 2021), which penalizes gates that succeed on average but fail in one contamination environment; GroupDRO (Sagawa et al., 2020) is its worst-group counterpart.

3 APPROACH

3.1 PROBLEM SETTING

Let M_θ be a frozen aligned language model. Assume that the operator owns M_θ and a safety adapter S, a LoRA (Hu et al., 2022) trained to strengthen refusal of unsafe requests. The operator also holds a pool of its own task adapters, with their task prompts, and uses them for one purpose: to train the composition policy before any external adapter arrives. The adapter evaluated at deployment is never in that pool; whether its task is in the pool separates the two panels (in-pool vs held-out) in Section 4.1. A task adapter T then arrives from a downstream party. T is a LoRA trained on the party's task data with no safety term. The operator receives only its finished weights: no data, no optimizer state, no record of the training run, and no way to retrain it. We call T *contaminated* when a fraction ρ of its training examples are unsafe, and we call ρ the contamination dose.

Discard, which runs $M_\theta + S$ without T, is the operator's other option; we report it in every table for reference rather than as a baseline (Appendix A). The threat is an adapter that makes the model answer unsafe requests while keeping its task accuracy, whether through careless data collection or malicious intent. The downstream party does not know the composer's parameters and does not optimize against them. Section 5 relaxes this with attacks on the response, including one that observes the gates.

Branch preservation. We require that the three components, which we call branches, stay untouched and separate: the operator loads T as received, keeps S as its own module, and never rewrites the stored weights of M_θ. This is a constraint on how the operator handles the components, not a safety guarantee; Appendix A explains what it buys the operator and why we compose activations rather than weights.

3.2 BRANCH-PRESERVING ACTIVATION COMPOSITION

Let z_{l-1} be the composed state entering decoder layer l. Let F_l^0 be decoder layer l of the base model, and let F_l^S and F_l^T be the same layer with the frozen LoRA updates of S and T applied on top of its frozen weights; these are the base, safety, and task branches at layer l. All three read z_{l-1}, and the composed state is

$$z_l = b_l + \alpha_l\,\Delta_l^S + \beta_l\,\Delta_l^T, \;\; \text{where } b_l = F_l^0(z_{l-1}),\; \Delta_l^S = F_l^S(z_{l-1}) - b_l,\; \Delta_l^T = F_l^T(z_{l-1}) - b_l. \quad (1)$$

Layer $l+1$ receives z_l, not b_l. The gates $\alpha_l, \beta_l \in (0,1)$ depend on the text so far, $x_{1:t}$; we suppress the decoding index t throughout. The model evaluates Equation (1) at every layer for every generated token, so the composition is one forward computation whose branches interact at each depth rather than an interpolation of three finished passes. Setting $\alpha_l = 1$ and $\beta_l = 0$ for all l recovers discard. The gates therefore interpolate, layer by layer, between discarding T and incorporating its full residual.

The gate network. The gate at layer l reads a summary of the base-branch state over the text so far. With m_j the attention mask and $b_{l,j}$ the base state at position j,

$$p_l = \frac{\sum_j m_j\, b_{l,j}}{\sum_j m_j}, \qquad u_l = \mathrm{ReLU}(W_p\, p_l + c_p), \qquad (\alpha_l, \beta_l) = \sigma\big(G_l([u_l;\, c])\big). \quad (2)$$

W_p is one projection from the model's hidden state into 64 dimensions, shared across layers, and each G_l is a layer-specific $65 \to 64 \to 2$ perceptron. In all experiments, the final input channel c is fixed to 0.5 and acts as a constant bias, not a signal (Appendix B). The gates are thus two scalars per layer, one per adapter; the model applies them at every position within a forward pass and recomputes them at every decoding step as the text grows. The composer adds only on the order of 10^5 parameters

to a multi-billion-parameter model; Appendix Table 4 gives the exact count per architecture. Since the gates depend on the whole text so far, the reference implementation forgoes key–value caching; letting them read the prompt only once restores it (Section 5).

3.3 META-TRAINING OBJECTIVE

We train the composer on a pool of task adapters, and every parameter other than the gates stays frozen. Training episodes combine four kinds of prompts: task prompts from a pool adapter's domain, unsafe prompts drawn from contamination environments e, which differ in contamination dose, benign prompts that should not be refused (Röttger et al., 2024), and general-capability prompts. For each kind, the target is what the trusted branch for that kind would do on its own: the task adapter on task prompts, the safety adapter on unsafe prompts, and the base model on benign and capability prompts. We obtain each target from an independent forward pass through that frozen branch, following feature-level distillation (Hinton et al., 2015; Romero et al., 2015). Each matching loss is the element-wise mean squared error between the composed model's hidden states and those of its target branch, computed at every decoder layer and averaged equally across layers. The task loss also asks the composed model to reproduce the continuations the task branch generates. The gates thus learn to reproduce, within a composed model, the behavior the operator would want from each branch in its own domain.

Let $\mathcal{L}_{\text{task}}$ denote task-branch matching, $\mathcal{L}^e_{\text{safe}}$ safety-branch matching in environment e, and $\mathcal{L}_{\text{over}}$ and $\mathcal{L}_{\text{cap}}$ base-branch matching on benign and capability prompts. A behavioral term $\mathcal{L}_{\text{beh}}$ acts on unsafe prompts: it is the sum of the next-token divergence $\mathrm{KL}(P_S \,\|\, P_{\text{comp}})$ from the safety branch and the teacher-forced negative log-likelihood of a fixed refusal. We train the safety term under variance-penalized risk extrapolation (Krueger et al., 2021). Let μ_e and $\widehat{\mathrm{Var}}_e$ be the mean and unbiased variance of $\mathcal{L}^e_{\text{safe}}$ over the contamination environments; then $\mathcal{R}_{\text{V-REx}} = \mu_e + \gamma\,\widehat{\mathrm{Var}}_e$. A gate that refuses at one contamination dose and fails at another therefore incurs a penalty for the spread, not just the mean. A second penalty, $\mathrm{Var}_{0,e}$, measures the variance of the same loss over the clean environment (e=0) and the contamination environments together. The two variance terms differ in scope: one measures the spread across contaminated environments, the other the spread between clean and contaminated ones:

$$\mathcal{L} = \mathcal{L}_{\text{task}} + \lambda_{\text{safe}}\,\mathcal{R}_{\text{V-REx}}\big(\{\mathcal{L}^e_{\text{safe}}\}\big) + \lambda_{\text{var}}\,\mathrm{Var}_{0,e}\left(\mathcal{L}^e_{\text{safe}}\right) + \lambda_{\text{over}}\mathcal{L}_{\text{over}} + \lambda_{\text{cap}}\mathcal{L}_{\text{cap}} + \lambda_{\text{beh}}\mathcal{L}_{\text{beh}}, \quad (3)$$

with $\lambda_{\text{var}} = 0.1$, $\lambda_{\text{over}} = 0.3$, and $\lambda_{\text{cap}} = \lambda_{\text{beh}} = 0.5$; λ_{safe} and γ are listed in Appendix B. Unsafe prompts enter through a curriculum (Bengio et al., 2009) that begins with benign episodes and raises the share of unsafe prompts over training, so the gates first learn to reproduce the branches and then learn where the branches conflict. We list optimizer settings, the curriculum schedule, and the early-stopping rule in Appendix B.

4 RESULTS

4.1 EXPERIMENT DESIGN

Models, adapters, and contamination. We evaluate on three model families: Qwen2.5-7B (Yang et al., 2024), Mistral-7B (Jiang et al., 2023), and Gemma-2-9B (Gemma Team, 2024). For each family, the operator's safety adapter S is a rank-64 LoRA trained with DPO (Rafailov et al., 2023) on refusal preferences. Every task adapter T is a rank-32 LoRA trained on task data with no safety term. We contaminate a task adapter by mixing unsafe BeaverTails (Ji et al., 2023) records into its training data at a nominal dose $\rho \in \{1\%, 5\%, 10\%\}$. The nominal dose is the share of those records in the training mix; the share that BeaverTails labels unsafe is lower, about 2.9% at the 5% dose, and we report nominal values throughout. The in-pool panels and the Qwen repeated-arrival sequence use five seeds; the held-out panels, the Gemma sequence, and the attack and control experiments in Section 5 use three. A seed fixes the training of the composer, the task adapter, and every learned defense, so each arm yields one paired observation per seed (Appendix E).

Two panels: in-pool and held-out. Each family has two evaluation panels, which differ in what the composer saw during meta-training and in how much the task adapter contributes. In the *in-pool* panel, T is a GSM8K adapter (Cobbe et al., 2021): its training prompts appear in the composer's

training data, but the adapter itself and its test items do not, and the aligned base already performs well on GSM8K, so discard is at its strongest and the comparison isolates safety recovery. In the *held-out* panel, T is a MedMCQA adapter (Pal et al., 2022) trained after the composer was frozen: neither the adapter nor any MedMCQA data appears in composer training, and the base performs poorly on the task, so discarding T is costly. Only the task and the adapter differ from the training pool; how adapters are trained, how unsafe examples enter them, and the kind of unsafe request stay the same. The held-out panel tests the transfer hypothesis under the deployment setting for which the method is designed.

Metrics. Safety is measured by the attack success rate (ASR, lower is better) on 104 held-out AdvBench (Zou et al., 2023b) prompts, judged by the HarmBench (Mazeika et al., 2024) classifier. Capability in the in-pool panel is 8-shot exact-match accuracy on 520 held-out GSM8K test items. Capability in the held-out panel is exact-match accuracy on 520 MedMCQA items, a fixed random subset of the dataset's validation split that we reserve as the test set. On Qwen at the 5% dose we also report HarmBench (Mazeika et al., 2024) and JailbreakBench (Chao et al., 2024) for safety, MMLU (Hendrycks et al., 2021) and MT-Bench (Zheng et al., 2023) for general capability, and XSTest (Röttger et al., 2024) for over-refusal (Appendix I).

Compared configurations. Every panel table contains the same seven arms. SafeCompose composes M_θ, S, and T through the frozen gates. Discard runs $M_\theta + S$ without T; it is the operator's fallback and is reported for reference, not as a baseline. No defense runs $M_\theta + T$ and shows how the received adapter behaves without protection. Four baselines represent the weight-space and inference-time families: SaLoRA (adapted)[1] (Li et al., 2025a), CAST (Lee et al., 2025), SafetyLock (Zhu et al., 2024), and naive weight merge, which adds both adapters at full weight. In the panels each baseline uses one default configuration from its public implementation across all seeds, doses, and families; Appendix C details the settings.

To check that fixed settings do not understate the baselines, we swept the strength of a weighted merge, CAST, and SafetyLock, selected one strength per defense on a validation split, and evaluated it once on the test split (Appendix Table 18). Tuning does not rescue the steering defenses, which still yield 19–33% ASR at their best validation setting. It does rescue weighted merging: one weight per family, held fixed across seeds and doses, reaches SafeCompose's safety and capability at every dose on Qwen, and yields a safer but less capable point on Mistral and a slightly less capable one on Gemma (Appendix Table 24). The weight-space arms in the panels use one setting for the whole model rather than one per layer; additional baselines were evaluated separately (Appendix C). All arms share the same prompts, classifier, decoding settings, and evaluation harness.

4.2 Held-out transfer: composing outperforms discarding when the adapter adds capability

On the held-out panels (Table 2), SafeCompose keeps the capability the base lacks: it scores about 58%, 53%, and 60% on MedMCQA on Qwen, Mistral, and Gemma, 9–31 points above discard and the highest of any method at its default setting at every dose. The undefended adapter is nearly as accurate, but its ASR reaches 38.5%, 64.7%, and 48.4% at the 10% dose. The weight-space baselines keep most of the accuracy but, on Qwen and Gemma, let ASR grow with dose, and the steering defenses remove much less of the erosion.

Safety divides the families. On Qwen and Gemma the composer is as safe as discard (0.6% and 0.0% ASR), and on Gemma it is strictly better than discard: higher on both MedMCQA (by 6.3–11.0 points) and GSM8K (by 2.7–5.4) at equal ASR in every seed and dose. On Mistral it leaves 7–9% ASR, above the near-zero of discard, SaLoRA, and naive merge, the same trade-off as in the in-pool panel.

A weighted merge with one validation-selected weight matches this result on Qwen, where it is above SafeCompose on MedMCQA by 0.6–2.7 points at equal ASR, but not on the other families: on Mistral it is safer and less capable, and on Gemma it gives up 3.3–4.2 points of GSM8K (Appendix Tables 19 and 24). Without an attack, a well-chosen merge therefore matches or trades against

[1] A post-hoc adaptation, not the constrained training procedure of Li et al. (2025a).

Table 2: Held-out MedMCQA panels on three families. Mean ± SD over three seeds. MedMCQA is exact-match accuracy on 520 items, a fixed subset of the dataset's validation split; the MedMCQA adapters were trained after the composer was frozen. ASR is the HarmBench classifier on 104 held-out AdvBench prompts. Within each family, bold marks the best mean per column among SafeCompose and the four baselines, and underline the second-best; discard and the undefended adapter are listed below them for reference.

	MedMCQA accuracy (%) ↑			AdvBench ASR (%) ↓		
Method	1%	5%	10%	1%	5%	10%
Qwen2.5-7B						
SafeCompose	$\mathbf{58.3 \pm 0.9}$	$\mathbf{59.0 \pm 1.3}$	$\mathbf{59.4 \pm 0.2}$	0.6 ± 0.6	$\mathbf{0.6 \pm 0.6}$	$\mathbf{0.6 \pm 0.6}$
SaLoRA	55.7 ± 0.7	56.9 ± 0.8	56.8 ± 0.7	0.6 ± 0.6	5.1 ± 2.2	6.1 ± 3.4
Naive merge	55.7 ± 1.4	55.8 ± 1.6	56.5 ± 0.6	$\mathbf{0.3 \pm 0.6}$	7.1 ± 4.7	9.0 ± 6.5
CAST	56.0 ± 0.5	55.8 ± 1.6	57.1 ± 1.2	5.8 ± 3.5	28.5 ± 7.2	39.7 ± 19.1
SafetyLock	56.2 ± 0.7	56.0 ± 1.9	56.8 ± 0.7	5.1 ± 3.4	28.5 ± 9.8	36.9 ± 16.9
Discard T ($M_\theta + S$)	28.0 ± 2.3	28.0 ± 2.3	28.0 ± 2.3	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0
No defense	56.3 ± 0.2	56.2 ± 1.4	57.1 ± 0.9	6.7 ± 3.5	34.3 ± 10.6	38.5 ± 14.5
Mistral-7B						
SafeCompose	$\mathbf{53.2 \pm 0.5}$	$\mathbf{53.9 \pm 1.1}$	$\mathbf{53.8 \pm 1.0}$	8.7 ± 2.5	9.3 ± 2.2	7.1 ± 3.4
SaLoRA	50.0 ± 1.0	51.5 ± 1.8	49.9 ± 1.4	$\mathbf{0.3 \pm 0.6}$	$\mathbf{0.3 \pm 0.6}$	$\mathbf{0.3 \pm 0.6}$
Naive merge	49.1 ± 1.9	50.7 ± 1.6	49.0 ± 0.7	0.6 ± 1.1	0.6 ± 0.6	$\mathbf{0.3 \pm 0.6}$
CAST	52.3 ± 1.3	52.2 ± 0.6	51.6 ± 0.7	43.6 ± 8.6	52.6 ± 14.7	43.3 ± 14.9
SafetyLock	51.7 ± 1.5	52.6 ± 0.4	51.2 ± 0.5	30.1 ± 16.4	30.1 ± 6.8	28.8 ± 11.1
Discard T ($M_\theta + S$)	36.3 ± 3.2	36.3 ± 3.2	36.3 ± 3.2	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
No defense	52.2 ± 1.1	52.1 ± 0.5	51.5 ± 0.5	64.1 ± 15.8	63.1 ± 8.3	64.7 ± 5.6
Gemma-2-9B						
SafeCompose	$\mathbf{59.9 \pm 0.3}$	$\mathbf{59.9 \pm 0.3}$	$\mathbf{60.8 \pm 1.0}$	$\mathbf{0.0 \pm 0.0}$	$\mathbf{0.0 \pm 0.0}$	$\mathbf{0.0 \pm 0.0}$
SaLoRA	59.2 ± 1.1	59.5 ± 0.4	60.1 ± 1.0	$\mathbf{0.0 \pm 0.0}$	0.6 ± 1.1	1.6 ± 1.5
Naive merge	59.2 ± 0.5	58.8 ± 1.3	59.6 ± 1.0	$\mathbf{0.0 \pm 0.0}$	0.3 ± 0.6	5.1 ± 4.5
CAST	58.7 ± 1.0	58.6 ± 1.5	59.2 ± 1.3	11.9 ± 9.7	25.0 ± 8.4	46.5 ± 12.8
SafetyLock	58.7 ± 1.1	58.6 ± 0.9	59.3 ± 1.1	11.9 ± 10.5	24.0 ± 10.0	44.6 ± 14.7
Discard T ($M_\theta + S$)	50.8 ± 2.3	50.8 ± 2.3	50.8 ± 2.3	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
No defense	58.7 ± 0.8	58.8 ± 1.5	59.3 ± 1.1	11.9 ± 9.7	24.7 ± 9.7	48.4 ± 14.9

composition; the composer's advantage over it appears under attack (Section 5). Appendix G reads the panels arm by arm, and Appendix H reports the seed-level consistency of the orderings.

4.3 In-pool panel: safety recovery when the operator's alternative is strong

On the in-pool GSM8K panels (Table 3; Appendix Tables 8 and 9), the baselines at their default settings lose 11–30 points of GSM8K to SafeCompose in all three families. SafeGrad and TIES-Merging, run on Qwen, fall short in the same way: SafeGrad keeps ASR at 2–5% but scores 70–73%, and TIES-Merging scores 76–78% at 19–35% ASR (Appendix Table 20). The validation-selected merge ties SafeCompose on both axes on Qwen; it was not run on the other in-pool panels.

On Qwen and Gemma, SafeCompose stays within one prompt of discard's ASR at every dose, while the weight-space baselines let ASR grow with dose and the steering defenses stay near the undefended level. On Mistral it trades safety for capability: it is 7–8 points above discard on GSM8K at 7–13% ASR, where every arm that stays safe scores about 20 points below discard. A rerun trained through the full curriculum reproduces this trade-off, so it is not an artifact of early stopping (Appendix Table 14), and no other arm beats either discard or SafeCompose on both axes.

How much composing gains over discarding tracks how much the base already does: discard is about 2 points more accurate on Qwen, where $M_\theta + S$ scores 86.6%, while composition is 2–3 points ahead on Gemma (82.7%) and 7–8 on Mistral (37.5%), as the setup predicts (Appendix A). On HarmBench and JailbreakBench at the 5% Qwen dose, SafeCompose has 10.3% and 3.7% ASR, above discard's

Table 3: In-pool GSM8K panel on Gemma-2-9B. Mean ± SD over five seeds. GSM8K is 8-shot exact match on 520 held-out test items; ASR is the HarmBench classifier on 104 held-out AdvBench prompts. Bold marks the best mean per column among SafeCompose and the four baselines, and underline the second-best; discard and the undefended adapter are listed below them for reference. Appendix F reports Qwen2.5-7B and Mistral-7B.

	GSM8K accuracy (%) ↑			AdvBench ASR (%) ↓		
Method	1%	5%	10%	1%	5%	10%
SafeCompose	$\mathbf{85.8 \pm 0.9}$	$\mathbf{85.2 \pm 1.1}$	$\mathbf{85.1 \pm 0.8}$	$\mathbf{0.2 \pm 0.4}$	$\mathbf{0.6 \pm 0.5}$	$\mathbf{1.0 \pm 0.7}$
SaLoRA	58.1 ± 1.6	57.5 ± 1.0	57.1 ± 2.4	$\underline{4.0 \pm 4.2}$	$\underline{8.8 \pm 3.2}$	$\underline{10.2 \pm 5.8}$
Naive merge	57.2 ± 0.8	55.4 ± 1.6	56.3 ± 3.1	6.0 ± 4.0	9.4 ± 3.7	13.8 ± 6.0
CAST	63.4 ± 1.4	62.1 ± 1.7	$\underline{62.9 \pm 0.8}$	63.7 ± 9.5	67.5 ± 9.0	76.7 ± 8.2
SafetyLock	$\underline{63.6 \pm 1.1}$	$\underline{62.3 \pm 1.8}$	$\underline{62.9 \pm 1.2}$	63.1 ± 6.4	66.7 ± 8.3	76.5 ± 6.0
Discard T ($M_\theta + S$)	82.7 ± 1.3	82.7 ± 1.3	82.7 ± 1.3	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
No defense	62.9 ± 1.1	61.9 ± 1.2	63.2 ± 1.1	64.2 ± 7.4	67.3 ± 8.2	78.1 ± 6.7

3.5% and 2.7% but far below SaLoRA's 36–38% and the steering defenses' 64–72% (Appendix Table 10); Appendix I adds MMLU, MT-Bench, and XSTest.

4.4 Repeated adapter arrivals

Across five arriving adapters, the frozen gates carry over without retraining. On Qwen (Appendix Table 15), SafeCompose's ASR stays within one prompt of its start while the steering defenses and the undefended model drift up by 15–17 points and naive merge by about three, and it ends with the highest GSM8K accuracy of these arms (83.8% against 77.1–78.2%). On Gemma (Appendix Table 16), the steering defenses and the undefended model fail at the first arrival instead: they start at 23–28% ASR and end near 29%, while SafeCompose stays at or below 0.6%. The validation-selected merge is as flat as the composer on both families, so the sequence separates composition from steering rather than the gate from a well-chosen merge; on Gemma, capability still separates them, with final GSM8K of 85.2% for SafeCompose against 82.5% for that merge and 66.9% for the equal-weight merge. Capability also holds across arrivals on Qwen (Appendix Table 17), though with three seeds and 32 items per task this is descriptive.

5 Further analyses and limitations

Why the residuals must be scaled. Removing the gates and adding both residuals raises ASR from 0.6% to 17.0% on the Qwen in-pool panel, while a gate with fixed values brings it back to 1.0% (Appendix Table 11): rescaling the two residuals at each layer, rather than adding them, removes most of the erosion. This differs from a per-layer weight merge, because each branch passes through the layer's nonlinearity before its residual is formed. Ablations under matched conditions on the held-out adapter reproduce the separation, with 3.9–13.5% ASR without gates, while every other ablation stays within two AdvBench prompts and 0.7 points of MedMCQA of the full design (Appendix Tables 12 and 13).

Robustness under attack. The main panels assume an attacker who does not know the gates; since the gates read the text so far, we also let the attacker control how the response begins. Forcing it to begin with "Sure, here is" raises SafeCompose's ASR from 0.6% to 47.1%, but the validation-selected merge reaches 73.4% and every baseline at its default setting 86–93% (Appendix Table 21). The advantage is not an artifact of one phrase: an attacker who sees the gates and adapts the opening leaves the merge 24–31 points above the composer (Appendix Table 22), a second judge, Llama-Guard-3, keeps the fixed-phrase gap at 12.5 points, and a layer-wise merge, SafeMERGE, does no better (Appendix Table 25).

What reading the prompt adds. Without an attack, a gate with fixed values does as well as the learned gate, but under these attacks gates with fixed values, whether one per layer or two for the whole model, reach 78–83% ASR where the learned gate holds at 41–51% (Appendix Table 26). A

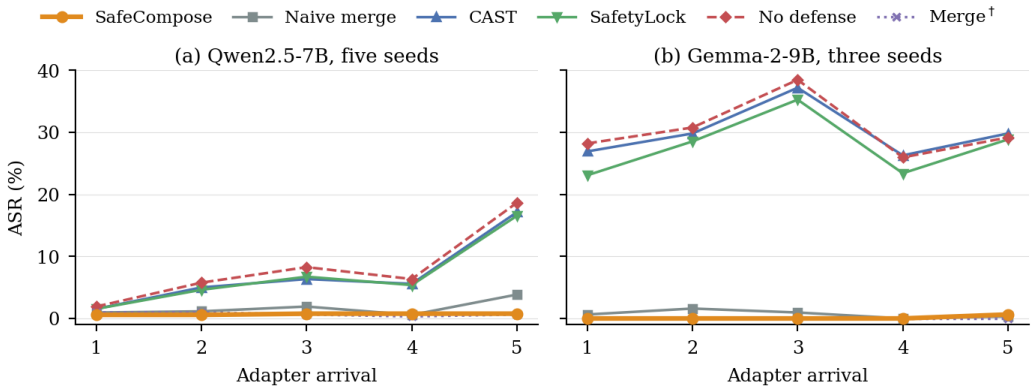

Figure 2: Repeated adapter arrivals: ASR after each of five arriving adapters on Qwen2.5-7B and Gemma-2-9B. SafeCompose stays below 1% throughout. †: the validation-selected merge ($\alpha = 0.75$), which lies close to SafeCompose on both families. Final GSM8K accuracy and drift tests are in Appendix Tables 15 and 16.

gate that reads the prompt only once and a gate with fixed values both allow cached decoding, but only the former keeps that advantage; what the gate adds is therefore a single read of the prompt. Across repeated arrivals, by contrast, the merge is as stable in safety as the composer, and the two separate only in Gemma's final capability (Section 4), so attacks on the response are where reading the prompt matters (Appendix N).

Limitations. Discarding the adapter is safer than composing under these attacks, by 9–13 points, and RESTA, a post-hoc baseline that moves the weights away from a deliberately unaligned copy of the model, is safer still under the fixed phrase at 27.9%, though three points less capable; the advantage above therefore holds against composition baselines only (Appendix Table 25). The attacks are bounded, with at most twelve candidate openings, eight queries per prompt, and no gradient-based or token-level search, and like the ablations they cover one Qwen adapter at three seeds. On Mistral the composer leaves 7–13% ASR, and on Qwen it inherits the safety adapter's over-refusal, refusing 9–13 points more of OR-Bench-Hard-1K (Cui et al., 2025) than the baselines at their default settings (Appendix Tables 23). Even with batched branches and gates that read the prompt only once, composed decoding costs 3.96× and 4.97× as much as cached base decoding at 20- and 900-token contexts, about four to five times a single merged model (Appendix Table 30).[2]

6 Conclusion

When an operator receives a task adapter it cannot retrain, it has two options: ship it or discard it. SafeCompose adds a third. The aligned base, the safety adapter, and the received adapter stay frozen and separate, and small per-layer gates read the input to set how much each adapter contributes. The gates are trained once on adapters the operator already holds and reused unchanged on adapters that arrive later. On a task the gates never saw, SafeCompose restores refusal and keeps more task capability than weight-projection and activation-steering defenses, retaining the 9–31 points of accuracy that discarding the adapter would give up. It is as safe as discarding on every model family except Mistral. Over successive adapter arrivals, it stays safe where steering defenses do not. Without an attack, fixed gates trained the same way match it, but when the response is forced to begin by agreeing, even by an attacker who sees the gates, SafeCompose is safer, and it keeps this advantage when its gates read the prompt only once, which allows cached decoding. The costs are greater over-refusal than the baselines at their default settings on safe but sensitive prompts and decoding about four to five times as costly as a single merged model. Discarding costs no capability when the base already performs the task well while composition matters more as the base becomes less capable on it.

[2]Reported numbers use a slower reference implementation that forgoes caching; see Appendix M.

AI USE STATEMENT

We used a large language model to edit and restructure prose throughout the manuscript and to draft LaTeX table skeletons. We did not use generative AI to generate research ideas, run or analyze experiments, or produce code used in the reported results. All AI-assisted text was reviewed against the experimental records by the authors, who take responsibility for the final content, claims, and artifacts.

ETHICS STATEMENT

This work evaluates defenses against safety erosion in fine-tuned language models and therefore involves prompts from AdvBench and BeaverTails that request harmful content. Generated model outputs to such prompts are used only for automated classification and are not released; released artifacts contain configurations, checkpoints, and aggregate results. The method is intended to reduce unsafe compliance; we note in Section 5 that it is evaluated against a non-adaptive attacker in the main panels and against prefix-level attacks only, and should not be treated as a safety guarantee.

REPRODUCIBILITY STATEMENT

Section 3 specifies the composition and objective, Section 4.1 the panels, metrics, and arms, and Appendices B–E the hyperparameters, baseline configurations, and statistical procedures. The repository releases code, configurations, per-seed result files, and analysis scripts sufficient to regenerate every table.

REFERENCES

Aznaur Aliev, Carlos Hinojosa, Abdelrahman Eldesokey, Bang An, Bernard Ghanem, and Yibo Yang. HyperSafe: Inference-time safety recovery for fine-tuned language models. *arXiv preprint arXiv:2607.11475*, 2026.

Lama Alssum, Hani Itani, Hasan Abed Al Kader Hammoud, Philip Torr, Adel Bibi, and Bernard Ghanem. Unforgotten safety: Preserving safety alignment of large language models with continual learning. *arXiv preprint arXiv:2512.10150*, 2025.

Marco Arazzi, Vignesh Kumar Kembu, Antonino Nocera, Stjepan Picek, and Saraga Sakthidharan. You snooze, you lose: Automatic safety alignment restoration through neural weight translation. *arXiv preprint arXiv:2605.04992*, 2026.

Andy Arditi, Oscar Obeso, Aaquib Syed, Daniel Paleka, Nina Panickssery, Wes Gurnee, and Neel Nanda. Refusal in language models is mediated by a single direction. In *Advances in Neural Information Processing Systems*, 2024.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *International Conference on Machine Learning*, 2009.

Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society: Series B (Methodological)*, 57(1):289–300, 1995.

Rishabh Bhardwaj, Do Duc Anh, and Soujanya Poria. Language models are Homer Simpson! safety re-alignment of fine-tuned language models through task arithmetic. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

Eric L. Buehler and Markus J. Buehler. X-LoRA: Mixture of low-rank adapter experts, a flexible framework for large language models with applications in protein mechanics and molecular design. *APL Machine Learning*, 2(2):026119, 2024.

Patrick Chao, Edoardo Debenedetti, Alexander Robey, Maksym Andriushchenko, Francesco Croce, Vikash Sehwag, Edmund Wong, Hamed Hassani, George J. Pappas, Florian Tramèr, Bo Li, and Eric Wong. JailbreakBench: An open robustness benchmark for jailbreaking large language models. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track*, 2024.

Sahil Chaudhary. Code Alpaca: An instruction-following LLaMA model for code generation. `https://github.com/sahil280114/codealpaca`, 2023.

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

Justin Cui, Wei-Lin Chiang, Ion Stoica, and Cho-Jui Hsieh. OR-bench: An over-refusal benchmark for large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 11515–11542. PMLR, 2025.

Aladin Djuhera, Swanand Ravindra Kadhe, Farhan Ahmed, Syed Zawad, and Holger Boche. Safe-MERGE: Preserving safety alignment in fine-tuned large language models via selective layer-wise model merging. *arXiv preprint arXiv:2503.17239*, 2025.

Falcon-LLM Team. The Falcon 3 family of open models. `https://huggingface.co/blog/falcon3`, December 2024.

Ronald A. Fisher. On the interpretation of χ^2 from contingency tables, and the calculation of P. *Journal of the Royal Statistical Society*, 85(1):87–94, 1922.

Ronald A. Fisher. *The Design of Experiments*. Oliver and Boyd, Edinburgh, 1935.

Suyu Ge, Chunting Zhou, Rui Hou, Madian Khabsa, Yi-Chia Wang, Qifan Wang, Jiawei Han, and Yuning Mao. MART: Improving LLM safety with multi-round automatic red-teaming. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 2024.

Gemma Team. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021.

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

Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70, 1979.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *International Conference on Machine Learning*, pp. 2790–2799, 2019.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Tiansheng Huang, Sihao Hu, and Ling Liu. Vaccine: Perturbation-aware alignment for large language models against harmful fine-tuning attack. In *Advances in Neural Information Processing Systems*, 2024.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *International Conference on Learning Representations*, 2023.

Jiaming Ji, Mickel Liu, Juntao Dai, Xuehai Pan, Chi Zhang, Ce Bian, Boyuan Chen, Ruiyang Sun, Yizhou Wang, and Yaodong Yang. BeaverTails: Towards improved safety alignment of LLM via a human-preference dataset. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.

Jiaming Ji, Donghai Hong, Borong Zhang, Boyuan Chen, Josef Dai, Boren Zheng, Tianyi Alex Qiu, Jiayi Zhou, Kaile Wang, Boxun Li, Sirui Han, Yike Guo, and Yaodong Yang. PKU-SafeRLHF: Towards multi-level safety alignment for LLMs with human preference. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 31983–32016, 2025.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7B. *arXiv preprint arXiv:2310.06825*, 2023.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

David Krueger, Ethan Caballero, Joern-Henrik Jacobsen, Amy Zhang, Jonathan Binas, Dinghuai Zhang, Rémi Le Priol, and Aaron Courville. Out-of-distribution generalization via risk extrapolation (REx). In *International Conference on Machine Learning*, 2021.

Bruce W. Lee, Inkit Padhi, Karthikeyan Natesan Ramamurthy, Erik Miehling, Pierre Dognin, Manish Nagireddy, and Amit Dhurandhar. Programming refusal with conditional activation steering. In *International Conference on Learning Representations*, 2025.

Simon Lermen, Charlie Rogers-Smith, and Jeffrey Ladish. LoRA fine-tuning efficiently undoes safety training in Llama 2-Chat 70B. *arXiv preprint arXiv:2310.20624*, 2023.

Changyue Li, Jiaming He, Youliang Yuan, Jialin Wu, Boxi Yu, Zhicong Huang, and Pinjia He. TRACE: Trajectory-based safety patch learning for LLM post-training realignment. *arXiv preprint arXiv:2607.16242*, 2026.

Mingjie Li, Wai Man Si, Michael Backes, Yang Zhang, and Yisen Wang. SaLoRA: Safety-alignment preserved low-rank adaptation. In *International Conference on Learning Representations*, 2025a.

Shen Li, Liuyi Yao, Lan Zhang, and Yaliang Li. Safety layers in aligned large language models: The key to LLM security. In *International Conference on Learning Representations*, 2025b.

Alisa Liu, Xiaochuang Han, Yizhong Wang, Yulia Tsvetkov, Yejin Choi, and Noah A. Smith. Tuning language models by proxy. In *Conference on Language Modeling*, 2024.

Mantas Mazeika, Long Phan, Xuwang Yin, Andy Zou, Zifan Wang, Norman Mu, Elham Sakhaee, Nathaniel Li, Steven Basart, Bo Li, David Forsyth, and Dan Hendrycks. HarmBench: A standardized evaluation framework for automated red teaming and robust refusal. In *International Conference on Machine Learning*, 2024.

Eric Mitchell, Rafael Rafailov, Archit Sharma, Chelsea Finn, and Christopher D. Manning. An emulator for fine-tuning large language models using small language models. In *International Conference on Learning Representations*, 2024.

Ankit Pal, Logesh Kumar Umapathi, and Malaikannan Sankarasubbu. MedMCQA: A large-scale multi-subject multi-choice dataset for medical domain question answering. In *Conference on Health, Inference, and Learning*, 2022.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. AdapterFusion: Non-destructive task composition for transfer learning. In *Conference of the European Chapter of the Association for Computational Linguistics*, pp. 487–503, 2021.

Xiangyu Qi, Yi Zeng, Tinghao Xie, Pin-Yu Chen, Ruoxi Jia, Prateek Mittal, and Peter Henderson. Fine-tuning aligned language models compromises safety, even when users do not intend to! In *International Conference on Learning Representations*, 2024.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, 2023.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. FitNets: Hints for thin deep nets. In *International Conference on Learning Representations*, 2015.

Domenic Rosati, Jan Wehner, Kai Williams, Łukasz Bartoszcze, David Atanasov, Robie Gonzales, Subhabrata Majumdar, Carsten Maple, Hassan Sajjad, and Frank Rudzicz. Representation noising: A defence mechanism against harmful finetuning. In *Advances in Neural Information Processing Systems*, 2024.

Paul Röttger, Hannah Rose Kirk, Bertie Vidgen, Giuseppe Attanasio, Federico Bianchi, and Dirk Hovy. XSTest: A test suite for identifying exaggerated safety behaviours in large language models. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 2024.

Shiori Sagawa, Pang Wei Koh, Tatsunori B. Hashimoto, and Percy Liang. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. In *International Conference on Learning Representations*, 2020.

Tianhao Shen. Dual-responder and dynamic dual-routing: LoRA-based safety–availability synergy for large language models. *Neurocomputing*, 686:133756, 2026.

Student. The probable error of a mean. *Biometrika*, 6(1):1–25, 1908.

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.

Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J. Vazquez, Ulisse Mini, and Monte MacDiarmid. Activation addition: Steering language models without optimization. *arXiv preprint arXiv:2308.10248*, 2023.

Chengxiao Wang, Enyi Jiang, Xiaojing Liao, and Sanmi Koyejo. CLEAR: Continuous latent adapter routing for utility-preserving LLM safety alignment. *arXiv preprint arXiv:2608.21278*, 2026.

Hanqing Wang, Bowen Ping, Shuo Wang, Xu Han, Yun Chen, Zhiyuan Liu, and Maosong Sun. LoRA-flow: Dynamic LoRA fusion for large language models in generative tasks. In *Annual Meeting of the Association for Computational Linguistics*, pp. 12871–12882, 2024a.

Yuxia Wang, Haonan Li, Xudong Han, Preslav Nakov, and Timothy Baldwin. Do-not-answer: Evaluating safeguards in LLMs. In *Findings of the Association for Computational Linguistics: EACL 2024*, pp. 896–911, St. Julian's, Malta, 2024b. Association for Computational Linguistics.

Zhiqi Wang, Shizhu He, Kang Liu, and Jun Zhao. Instance-level dynamic LoRAs composition for cross-task generalization. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 5699–5708, 2024c.

Boyi Wei, Kaixuan Huang, Yangsibo Huang, Tinghao Xie, Xiangyu Qi, Mengzhou Xia, Prateek Mittal, Mengdi Wang, and Peter Henderson. Assessing the brittleness of safety alignment via pruning and low-rank modifications. In *International Conference on Machine Learning*, 2024.

Bernard L. Welch. The generalization of 'Student's' problem when several different population variances are involved. *Biometrika*, 34(1/2):28–35, 1947.

Stefan Wellek. *Testing Statistical Hypotheses of Equivalence and Noninferiority*. Chapman and Hall/CRC, 2nd edition, 2010.

Liang Xu, Kangkang Zhao, Lei Zhu, and Hang Xue. SC-Safety: A multi-round open-ended question adversarial safety benchmark for large language models in Chinese. *arXiv preprint arXiv:2310.05818*, 2023.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. TIES-Merging: Resolving interference when merging models. In *Advances in Neural Information Processing Systems*, 2023.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Xianjun Yang, Xiao Wang, Qi Zhang, Linda Petzold, William Yang Wang, Xun Zhao, and Dahua Lin. Shadow alignment: The ease of subverting safely-aligned language models. *arXiv preprint arXiv:2310.02949*, 2023.

Biao Yi, Jiahao Li, Baolei Zhang, Lihai Nie, Tong Li, Tiansheng Huang, and Zheli Liu. Gradient surgery for safe LLM fine-tuning. *arXiv preprint arXiv:2508.07172*, 2025.

Taeyoung Yun, Pierre-Luc St-Charles, Jinkyoo Park, Yoshua Bengio, and Minsu Kim. Active attacks: Red-teaming LLMs via adaptive environments. *arXiv preprint arXiv:2509.21947*, 2025.

Qiusi Zhan, Richard Fang, Rohan Bindu, Akul Gupta, Tatsunori Hashimoto, and Daniel Kang. Removing RLHF protections in GPT-4 via fine-tuning. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 2024.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.

Minjun Zhu, Linyi Yang, Yifan Wei, Ningyu Zhang, and Yue Zhang. Locking down the finetuned LLMs safety. *arXiv preprint arXiv:2410.10343*, 2024.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, Shashwat Goel, Nathaniel Li, Michael J. Byun, Zifan Wang, Alex Mallen, Steven Basart, Sanmi Koyejo, Dawn Song, Matt Fredrikson, J. Zico Kolter, and Dan Hendrycks. 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.

A BRANCH PRESERVATION, AND WHY WE COMPOSE ACTIVATIONS RATHER THAN WEIGHTS

The operator's two options. The operator has two options. Deploying M_θ with both adapters keeps whatever capability T supplies and risks the refusal erosion that contaminated fine-tuning causes; a few hundred examples, or one LoRA, are enough (Qi et al., 2024; Yang et al., 2023; Lermen et al., 2023; Zhan et al., 2024). Discarding T and running $M_\theta + S$ restores the trusted configuration and gives up that capability. Discard is the operator's other option, and every table reports it for reference rather than as a baseline. What discarding costs depends on whether T adds capability the base lacks. On a task the base already does well, discard costs nothing and a composer can at best tie; on a task the base cannot do, discard is expensive. Our evaluation covers both cases.

Branch preservation. The operator can remove T without rebuilding an earlier weight state, replace S without undoing a merge, and audit each branch against its stored checkpoint. None of that holds once an adapter has been translated, patched, or merged. Frozen branches can still combine into unsafe behavior, so we measure every composed deployment for safety, and discard remains the rollback point.

Why compose activations rather than weights. The two adapters interact when both are applied, and existing repairs settle that interaction at one of three places. Each place fixes something our setting must keep free. Weight merging, whether naive (Ilharco et al., 2023), interference-aware (Yadav et al., 2023), safety-projected (Li et al., 2025a), or layer-selective (Djuhera et al., 2025), adds ΔW_S and ΔW_T into one parameter set. Its mixing ratios, one for the whole model or one per layer, are set before any input arrives, and the two contributions cannot be pulled apart afterward. Output-level mixing (Mitchell et al., 2024; Liu et al., 2024) runs three independent passes and interpolates their final states. Those passes diverge at the first layer, so the mix averages quantities computed on different histories, and the branches never meet during the forward pass. Adapter repair (Arazzi et al., 2026; Li et al., 2026) edits T itself, so the operator can no longer audit the adapter it received.

SafeCompose settles the interaction at a fourth place: inside each layer, on a shared state. Intervening on activations at inference time is well established (Turner et al., 2023; Zou et al., 2023a); here the intervention is a gate over the two adapters' own residuals rather than a fixed steering vector. All three branches read the same incoming state, so the safety and task residuals are measured against the same base output at the same position and compare directly. A small gate rescales them, and the next layer sees the composed state. The intuition is that the two adapters need not conflict everywhere. They may write to compatible directions in most layers and collide in a few, and where they collide may depend on the prompt. This agrees with evidence that refusal behavior and safety-critical parameters are sparse and localized rather than spread across the network (Wei et al., 2024; Arditi et al., 2024; Li et al., 2025b). A merge commits to its compromise, global or per layer, for all prompts. A per-layer gate conditioned on the text so far can admit the task residual where it is harmless and attenuate it where it works against refusal, and touches neither adapter. The gates are the only trainable parameters and see the adapters through their residuals alone, so a policy trained on a pool of adapters can be applied unchanged to one that arrives later; this is the transfer hypothesis that Section 4 tests.

B TRAINING AND IMPLEMENTATION DETAILS

Adapters and data. One safety adapter S is trained per family and shared across seeds. The meta-training pool draws task-side data from Alpaca and CodeAlpaca (Taori et al., 2023; Chaudhary, 2023) and from GSM8K training prompts; MedMCQA appears nowhere in the pool.

Composer size. The gate network is one shared projection $W_p \in \mathbb{R}^{64\times d}$ with bias and L layer-specific $65 \to 64 \to 2$ perceptrons, where d and L are the model's hidden size and layer count. Table 4 lists the resulting counts. The final input channel c is a slot for an external refusal-probe score, kept so that a probe can be plugged in later without changing the input shape of G_l.

Meta-training. The objective is Eq. 3 with $\lambda_{\text{var}} = 0.1$, $\lambda_{\text{over}} = 0.3$, and $\lambda_{\text{cap}} = \lambda_{\text{beh}} = 0.5$; the safety environments are the three doses, and the clean environment enters only the invariance penalty

Table 4: Trainable parameters of the composer per architecture; every other parameter is frozen.

Architecture	d	L	Composer parameters	Share of base
Qwen2.5-7B	3584	28	351,352	4.6×10^{-5}
Mistral-7B	4096	32	401,536	5.5×10^{-5}
Gemma-2-9B	3584	42	412,308	4.5×10^{-5}

$\mathrm{Var}_{0,e}$. We use Adam (Kingma & Ba, 2015) at learning rate 10^{-4}, batch size 4, a ceiling of 5,000 optimizer steps, and early stopping on the validation objective with patience 200. The share of unsafe episodes is 0 for the first 1,000 sampled episodes and rises linearly to 0.9 by episode 3,000, so the curriculum saturates near optimizer step 750 at batch size 4. Table 6 gives the step at which each selected checkpoint stopped. During training the base state is pooled over all positions including padding; at deployment pooling uses the attention mask as in Eq. 2. This is a train/deployment mismatch: padded positions enter the training-time mean, so the summary the gates saw depends on batch composition, and the deployment summary does not. All reported numbers use the masked deployment path, so whatever shift exists is inside the results. A paired comparison measures its size (Table 5): the frozen Qwen composer, on the 5% held-out adapter, was run through both poolings on identical inputs. ASR is identical under the two, and GSM8K, MedMCQA, and XSTest refusal move by 0.19, 0.77, and 0.4 points, although the token-level responses differ on 40 of 104, 363 of 520, 120 of 520, and 99 of 250 prompts; the gate values move by 0.024 (α) and 0.034 (β) on average and by at most 0.13, at layer 2. None of the thresholds fixed before the comparison for retraining with masked pooling (1.92 points of ASR, 2 points of MedMCQA or XSTest refusal) was reached, so that retraining was not run. This is one checkpoint and a descriptive result, not a test of equivalence.

Table 5: Training-time versus deployment-time pooling of the gate input on one frozen composer (Qwen2.5-7B, 5% held-out MedMCQA adapter; HarmBench classifier). Masked pooling averages the base state over non-padding positions, as at deployment (Eq. 2); unmasked pooling also averages over padding, as the trainer does. The same checkpoint and inputs are used for both.

Axis (n)	Masked	Unmasked	Difference (points)	Prompts with different responses
AdvBench ASR (104) ↓	0.96	0.96	0.00	40
GSM8K (520) ↑	86.92	87.12	−0.19	363
MedMCQA (520) ↑	57.12	57.88	−0.77	120
XSTest refusal (250) ↓	10.4	10.8	−0.4	99

Table 6: Optimizer step of the selected composer checkpoint per seed. The ceiling is 5,000 steps; the curriculum saturates near step 750.

Architecture	Seed 42	Seed 123	Seed 456	Seed 789	Seed 1011
Qwen2.5-7B	951	743	1,994	1,372	2,519
Mistral-7B	668	743	647	666	453
Gemma-2-9B	748	558	773	638	850

Decoding. Generation settings are identical for all arms and recorded in the released configurations: greedy decoding, at most 256 new tokens, batch size 8 for safety and 16 for utility (8 on Gemma), and SDPA attention, with pad token 151643 on Qwen and 0 on Mistral and Gemma.

C Baseline configurations and evaluation harness

In the panels each baseline uses one default configuration from its public implementation, fixed across seeds, doses, and families; validation-selected strengths for the weighted merge, CAST, and SafetyLock, the fixed $\alpha = 0.75$ merge derived from them, the family-level merge weights for Mistral and Gemma, and the post-hoc RESTA and rank-adapted SafeMERGE arms are reported in Appendix L, and the selected merge also appears in Tables 15, 16, 21, 22, 23, and 27. SaLoRA applies a rank-32 safety-output projection before an additive merge of the 196 LoRA deltas; this is a

post-hoc adaptation of the projection idea and does not reproduce the constrained training procedure of Li et al. (2025a), so its results should be read as an adapted projection baseline rather than as a reproduction of SaLoRA. Naive merge adds the same 196 deltas at unit weight. CAST steers with strength 0.5 using 28 condition directions and 14 active layers on Qwen; SafetyLock applies scale-0.5 shifts in 14 of 28 Qwen layers. Safety uses 104 held-out AdvBench prompts per seed judged by `cais/HarmBench-Llama-2-13b-cls` at batch size 8; capability uses 8-shot exact match on 520 GSM8K items and exact match on 520 MedMCQA items, a fixed random subset of the dataset's validation split, at batch size 16. All arms share the pad token, prompt templates, and decoding settings.

Baselines evaluated separately. Two further baselines ran on the Qwen in-pool panel in a separate evaluation (Appendix L): SafeGrad (Yi et al., 2025), which projects the task adapter's weight update away from a safety-preference gradient, and TIES-Merging (Yadav et al., 2023). Two post-hoc weight-space baselines ran on the Qwen held-out adapter in a further separate evaluation (Table 25): RESTA (Bhardwaj et al., 2024), and a rank-adapted SafeMERGE (Djuhera et al., 2025) that keeps its layer-selection rule but replaces the equal-rank factor merge, inapplicable to our rank-64 and rank-32 adapters, with dense-delta interpolation; the threshold it selects on validation data admits every layer, so it reduces to the $\alpha = 0.75$ merge. Apart from SafeMERGE, which reduces to a global merge here, the gate with fixed per-layer values of Appendix J, three seeds on Qwen, is the one layer-wise comparison with fixed weights, so the panels on their own do not separate per-layer granularity from input dependence; Section 5 separates the two under attack. Secondary axes use the public prompts and judges of HarmBench, JailbreakBench, MMLU, MT-Bench, and XSTest.

D Criteria behind Table 1

Table 1 judges each method against five criteria taken from the operator's setting. *Task adapter unseen at policy training*: the method's learned component, if any, was trained without access to the adapter it is later applied to. *No target-specific adaptation*: applying the method to a new adapter requires no optimization or fitting on that adapter. *Received adapter unchanged*: the adapter's weights are neither edited nor merged, and it remains a separately loadable module. *Separate task and safety branches*: the received task adapter and a safety module coexist as distinct branches at inference. *Independent control of both branches*: the strength of each branch can be set separately. A method is marked ◐ when it meets a criterion only under a condition our setting does not supply, or meets it in structure but not as a separately removable module.

TRACE (Li et al., 2026) learns a safety patch from fine-tuning trajectories and reuses it on other checkpoints. It is applicable to a checkpoint it has not seen, but learning the patch requires trajectory access, and the operator in our setting receives only a final checkpoint (◐ for unseen). HyperSafe (Aliev et al., 2026) generates a checkpoint-specific side network from a hypernetwork. It accepts a new checkpoint without gradient steps on it, but the hypernetwork is trained over a distribution of fine-tuned checkpoints and the generated network is conditioned on the target (◐ for unseen and for no target-specific adaptation); the side network attaches to the merged fine-tuned model rather than to the adapter as a separate module, so the received adapter survives in weights but not as an independently removable branch (◐ for received adapter unchanged). DR2 (Shen, 2026) and CLEAR (Wang et al., 2026) keep safety and utility modules as separately routed branches, but both modules are the operator's own and were present when the router was trained; there is no slot for a received third-party task adapter (◐ for separate branches).

E Statistical procedures

Statistical terms. The procedures below are standard. An exact sign-flip test is a randomization test for paired differences (Fisher, 1935): it flips the sign of each seed's difference in all 2^n ways and reports the share of assignments whose mean is at least as extreme as the observed one, so with n seeds its smallest one-sided p is $1/2^n$ and its smallest two-sided p is $2/2^n$. A paired t-test and the matching confidence interval (Student, 1908) treat the per-seed differences as normal, and Welch's t-test (Welch, 1947) compares two groups without assuming equal variances. When several tests are read together, the Holm procedure (Holm, 1979) raises their p-values step by step so that the chance of any false rejection in the family stays below the nominal level, and the Benjamini–Hochberg

procedure (Benjamini & Hochberg, 1995) instead controls the expected share of false rejections among the rejections, which is less strict. Fisher's exact test (Fisher, 1922) compares two proportions from a 2×2 table. A one-sided non-inferiority bound (Wellek, 2010) checks whether the upper confidence limit of a difference stays below a stated margin, rather than testing whether the difference is zero.

The same 104 prompts recur across seeds. The seed is the replication unit, and every table reports mean $\pm$ sample SD over the seeds its caption names. This appendix reports seed-paired tests against each baseline, and counts of how many seeds agree in direction; the main text describes effects by size and consistency. With three seeds the exact test's floor is .125, so on the held-out panels the transfer claim rests on effect sizes and direction counts; the in-pool panels, with five seeds, reach .031.

Tests. Seed is the unit of replication. For each baseline, dose, and axis we form the five paired seed differences and run an exact sign-flip test over all 2^5 sign assignments; the one-sided p-value is the fraction of assignments with a mean at least as extreme as the observed one, with floor $1/32 = .031$ when all five seeds agree. Holm adjustment is applied within each axis across the three doses of a baseline, or jointly across six tests where a table says so. Over the joint family of one panel, four baselines by three doses by two axes, the floor is $24 \times .031 = .75$, so per-cell Holm survival would require at least ten seeds; we therefore also report Benjamini–Hochberg adjustment at $q = .05$ over the same 24 tests, which retains all 24 on Qwen and 23 on Gemma, and a cluster-level omnibus test that treats each axis as a single test over the five per-seed differences averaged across the twelve baseline–dose contrasts of that axis, which gives $p = .031$ on both axes for Qwen and Gemma. Since $3 \times .031 = .094$, a three-dose exact family cannot fall below .05 after Holm even under perfect agreement, so the exact test certifies direction rather than magnitude; paired t-tests, and Welch t where variances differ, are reported as sensitivities that use effect size, with unadjusted intervals. Prompt-pooled Fisher tests treat the 5×104 responses as independent, which they are not, and appear only as sensitivities. Table 7 shows the format; the remaining comparisons summarized in Section 4 follow it.

Three-seed evaluations. The held-out panels and the Gemma sequence have three seeds, where the exact test's floor is .125; there we report direction counts, and state the Gemma end-of-sequence contrast as a separation in three of three seeds rather than as a test over prompts pooled across seeds. The Qwen sequence, at five seeds, tests each arm's drift against SafeCompose's with a two-sided paired t-test and Holm adjustment over the four contrasts. The ablations under matched conditions (Appendix J) are summarized with the seed as the unit: a 95% paired-t interval, the two-sided exact sign-flip p, whose floor at three seeds is $2/8 = .25$, and, for safety, an exploratory one-sided 95% non-inferiority bound at margins of 0, 1, and 2 points, chosen after the data were seen. None of these is presented as a significance result.

Table 7: Mistral-7B in-pool panel, SafeCompose versus SaLoRA over five seeds. Δ is SafeCompose minus SaLoRA in percentage points; p_U tests higher GSM8K, p_S tests lower ASR. Exact tests are Holm-adjusted jointly over six tests; Welch t is the utility sensitivity.

Dose	Δ GSM8K	Δ ASR	Exact p_U / p_S	Holm p_U / p_S	Welch t	Holm Welch p_U
1%	26.9	+8.08	.031 / 1.00	.19 / 1.00	13.92	2.2×10^{-6}
5%	27.8	+5.96	.031 / 1.00	.19 / 1.00	11.46	8.5×10^{-5}
10%	28.8	+10.96	.031 / 1.00	.19 / 1.00	8.39	7.4×10^{-4}

F IN-POOL PANEL TABLES

Tables 8 and 9 report the Qwen and Mistral in-pool GSM8K panels; the Gemma panel is Table 3 in Section 4. Every value was produced under the evaluation harness of Appendix C, and the tables follow the conventions of Table 3.

Table 8: Qwen2.5-7B, in-pool panel. Mean ± SD over five seeds. Conventions as in Table 3.

	GSM8K accuracy (%) ↑			AdvBench ASR (%) ↓		
Method	1%	5%	10%	1%	5%	10%
SafeCompose	$\mathbf{84.2 \pm 2.0}$	$\mathbf{84.7 \pm 2.2}$	$\mathbf{84.7 \pm 2.0}$	$\mathbf{0.8 \pm 1.3}$	$\mathbf{0.8 \pm 0.8}$	$\mathbf{0.4 \pm 0.5}$
SaLoRA	73.2 ± 0.8	74.0 ± 1.1	72.6 ± 1.8	5.0 ± 1.7	10.4 ± 8.3	9.6 ± 5.6
Naive merge	71.0 ± 1.3	73.0 ± 1.5	70.8 ± 1.1	5.2 ± 1.1	11.2 ± 9.1	11.7 ± 8.4
CAST	72.7 ± 1.6	73.7 ± 1.3	72.6 ± 1.3	30.0 ± 5.6	41.5 ± 6.6	49.0 ± 5.9
SafetyLock	72.5 ± 0.5	73.2 ± 1.1	72.4 ± 1.3	28.7 ± 6.3	41.5 ± 5.2	44.8 ± 3.2
Discard T ($M_\theta + S$)	86.6 ± 0.4	86.6 ± 0.4	86.6 ± 0.4	0.8 ± 0.4	0.8 ± 0.4	0.8 ± 0.4
No defense	72.7 ± 0.8	71.9 ± 2.8	71.3 ± 0.9	35.0 ± 7.2	45.8 ± 4.6	48.7 ± 4.6

Table 9: Mistral-7B, in-pool panel. Mean ± SD over five seeds. Conventions as in Table 3.

	GSM8K accuracy (%) ↑			AdvBench ASR (%) ↓		
Method	1%	5%	10%	1%	5%	10%
SafeCompose	$\mathbf{44.9 \pm 2.9}$	$\mathbf{45.4 \pm 2.4}$	$\mathbf{45.5 \pm 2.7}$	8.8 ± 6.7	7.3 ± 2.9	12.7 ± 10.6
SaLoRA	18.0 ± 3.2	17.6 ± 4.9	16.7 ± 7.2	$\mathbf{0.8 \pm 1.3}$	$\mathbf{1.3 \pm 1.6}$	$\mathbf{1.7 \pm 2.1}$
Naive merge	17.6 ± 2.8	16.6 ± 5.3	15.1 ± 5.5	1.2 ± 1.6	2.3 ± 3.6	3.1 ± 3.9
CAST	29.7 ± 2.2	28.0 ± 3.2	29.1 ± 2.1	58.5 ± 12.8	58.3 ± 12.0	57.9 ± 14.6
SafetyLock	27.8 ± 2.0	27.3 ± 3.6	28.8 ± 2.1	22.1 ± 12.5	23.8 ± 7.4	22.9 ± 11.4
Discard T ($M_\theta + S$)	37.5 ± 2.7	37.5 ± 2.7	37.5 ± 2.7	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
No defense	29.3 ± 2.5	27.9 ± 3.2	28.6 ± 1.6	64.8 ± 8.0	66.2 ± 10.3	61.2 ± 12.1

G FAMILY-BY-FAMILY READING OF THE PANELS

Qwen, held out. At every dose, SafeCompose recovers safety to the level of discard and keeps the capability that discard gives up. Discard scores 28.0% because the base cannot do the task; SafeCompose scores 58.3–59.4% at 0.6% ASR, against 1.0% for discard. Among the arms at their default settings, SafeCompose has the highest MedMCQA accuracy at every dose and matches the safety of discard. A weighted merge that scales the safety adapter's weights by $\alpha = 0.75$ and the task adapter's by 0.25, with α chosen on validation data and then held fixed (Appendix L), is above SafeCompose on MedMCQA in all nine seed–dose combinations, by 0.6–2.7 points, and tied on ASR.

Mistral, held out. Discard scores 36.3% because the base cannot do the task, and SafeCompose scores 53.2–53.9%. SafeCompose is favored in all three seeds on MedMCQA and ASR against the undefended adapter, CAST, and SafetyLock at every dose (one comparison at two of three: MedMCQA against SafetyLock at 5%), and on MedMCQA against SaLoRA and naive merge. As in the in-pool panel, its ASR of 7–9% sits above SaLoRA, naive merge, and discard, which are at 0–0.6%, in every seed; the Mistral trade-off carries over to the held-out adapter. The $\alpha = 0.75$ merge, which a family-level sweep on Mistral validation data also selects (Appendix Table 24), does not reproduce the Qwen tie. It is 5–7 points safer than SafeCompose, 5–6 points lower on GSM8K, and 0.6–2.3 points lower on MedMCQA: on this family the merge is a safer, less capable operating point, and neither arm dominates.

Gemma, held out. Discard scores 50.8%, against 28.0% on Qwen and 36.3% on Mistral, so less capability is at stake. SafeCompose scores 59.9–60.8% at 0.0% ASR on every seed and dose. Gemma is the one family on which it dominates discard on the held-out panel: equal ASR, MedMCQA higher by 6.3–11.0 points, and GSM8K higher by 2.7–5.4 points (85.8–86.2% against 81.9%), in every seed and dose. Against the composition baselines it is favored in three of three seeds on GSM8K at every dose, by 4–12 points, and on ASR against the undefended adapter, CAST, and SafetyLock. On MedMCQA its gaps to SaLoRA and naive merge are 0.4–2.2 points, inside the seed spread; on ASR those two arms tie it at 1% and sit 0.3–5.1 points above it at the higher doses. The validation-selected merge on Gemma, $\alpha = 0.5$ from a family-level sweep (Appendix Table 24), ties SafeCompose on

ASR and, within 0.5–0.7 points, on MedMCQA, and gives up 3.3–4.2 points of GSM8K at every dose in three of three seeds.

Mistral, in pool. On Gemma and Mistral SafeCompose is the one arm that scores above both of its own branches, $M_\theta + S$ and $M_\theta + T$ alike, so the gates recover more than either adapter delivers on its own. On Mistral it is still safer than CAST and SafetyLock in every seed at every dose, and the reruns trained through the full curriculum (Table 14, Appendix J) show that the residual ASR is not an artifact of early stopping. On Mistral, dropping the behavioral-safety term from the objective lowers the composed ASR by 2–8 prompts per seed at equal utility (Appendix J); on Qwen the same change has no effect.

Repeated arrivals, tests. On Qwen the validation-selected merge was added to the sequence at three seeds and drifts -0.3 points, against $+0.2$ for the composer; the paired drift gap against SafeCompose gives $p = .42$ and the end-of-sequence ASR $p = .73$. On Gemma no drift contrast approaches significance (paired $p = .64$–$.95$, Holm 1.0), because CAST, SafetyLock, and the undefended model have no headroom to erode: they start at 23–28%, peak at 35–38% at the third stage, and fall back. The separation in level is the largest in the study. At the end of the sequence those three arms sit 11.5–51.9 points above SafeCompose in every seed (stage-5 ASR by seed 42, 123, 456: SafeCompose 0.00, 0.96, 0.96; CAST 23.1, 49.0, 17.3; SafetyLock 20.2, 52.9, 13.5; undefended 24.0, 51.0, 12.5), the exact one-sided floor at three seeds. SafeCompose also ends above the $\alpha = 0.75$ merge on GSM8K in every seed, by 3.3, 3.8, and 1.0 points.

H Consistency across families, seeds, and doses

Two orderings hold in every panel: SafeCompose has the highest capability of any method at its default setting, and the undefended adapter and the two steering defenses have the three highest ASRs. The ordering between SafeCompose and the safe configurations on ASR depends on the family: at or below them on Qwen, within a prompt on Gemma, above them on Mistral.

For every baseline, dose, and axis of the in-pool panels we form the five paired seed differences and run the exact one-sided sign-flip test over the 2^5 assignments. When all five seeds agree, the p-value reaches its floor, $1/32 = .031$. On Qwen all 24 capability and safety contrasts against the four baselines at their default settings reach that floor. On Gemma 23 do; the exception is the SaLoRA safety contrast at 1%, where four of five seeds agree. On Mistral all 12 capability contrasts reach it, as do the six safety contrasts against CAST and SafetyLock, while the six safety contrasts against the merge family go the other way in every seed. Because $24 \times .031 = .75$, no single contrast can survive Holm adjustment over the joint family at five seeds. That is a property of the design, and we do not present the exact test as certifying per-cell significance. Appendix E reports two complementary summaries. Benjamini–Hochberg at $q = .05$ retains all 24 contrasts on Qwen and 23 on Gemma, and a cluster-level omnibus test over the five seed-mean differences per axis gives $p = .031$ on both axes for Qwen and Gemma. Welch t on the five-seed GSM8K means against SaLoRA is 9.8–11.4 on Qwen, 24–42 on Gemma, and 8.4–13.9 on Mistral; we report it as an effect-size sensitivity.

The held-out panels have three seeds, where the exact test's floor is .125, so there we report direction counts alone. SafeCompose is favored in three of three seeds on both axes against the undefended adapter, CAST, and SafetyLock at every dose on all three families, with one comparison at two of three on Mistral. On Gemma it is also favored in three of three seeds on both utility axes against discard, at equal ASR. Across-seed standard deviations of SafeCompose match those of the baselines, except Mistral ASR at 10%, where one seed drives the spread.

I Broader safety and capability benchmarks

Benchmarks. XSTest is a diagnostic rather than a held-out test: its prompt style overlaps the benign-preservation prompts used in meta-training. OR-Bench-Hard-1K (Cui et al., 2025) and Do-Not-Answer (Wang et al., 2024b), which do not overlap it, serve as held-out over-refusal distributions in Appendix L.

Table 10 reports the secondary Qwen axes at the 5% dose against all baselines. MMLU and MT-Bench exist for SafeCompose and SaLoRA alone, from an earlier evaluation, so the table does not

compare general utility across the merge and steering arms. SafeCompose has the lowest ASR of any composition method on HarmBench and JailbreakBench. On XSTest it complies with fewer safe-but-sensitive prompts than any composition baseline, though more than discard, which over-refuses most. XSTest prompts overlap the benign-preservation prompts in meta-training, so we read this as a warning about over-refusal and the objective's benign-preservation weight, not as a held-out generalization result; Section 5 returns to it. The held-out OR-Bench-Hard-1K set (Appendix Table 23) confirms the warning. There SafeCompose and discard both refuse 9–13 points more than every baseline at its default setting under two semantic judges, and the difference between the two is inside judge disagreement. Do-Not-Answer, a third distribution of 939 prompts (Appendix Table 27), repeats the pattern: SafeCompose, discard, and the validation-selected merge refuse 39–40% of its prompts, within seed spread of one another.

Table 10: Qwen2.5-7B, broader benchmarks at the 5% dose. Mean ± SD over three seeds. ASR columns: lower is better. MMLU, MT-Bench, and XSTest compliance with safe prompts: higher is better. HarmBench (standard behaviors, n=104), JailbreakBench (n=100), and XSTest (n=250) come from the pinned evaluation harness. MMLU (5-shot generative, n=200) and MT-Bench (GPT-4 judge), marked ‡, exist only from an earlier evaluation run for SafeCompose and SaLoRA; this evaluation did not include naive merge, whose XSTest and OR-Bench-Hard results from a separate evaluation are in Table 23. XSTest is a diagnostic; note that discard over-refuses most. Bold marks the best mean per column among SafeCompose and the four baselines, and underline the second-best; discard and the undefended adapter are listed below them for reference.

	ASR (%) ↓		Capability ↑		
Method	HarmBench	JailbreakBench	MMLU (%)	MT-Bench	XSTest (%)
SafeCompose	$\mathbf{10.3 \pm 4.0}$	$\mathbf{3.7 \pm 1.5}$	$\mathbf{70.5 \pm 0.9}^{\ddagger}$	$\mathbf{7.01 \pm 0.20}^{\ddagger}$	92.9 ± 1.8
SaLoRA (adapted)	$\underline{37.8 \pm 6.8}$	$\underline{35.7 \pm 4.5}$	$\underline{67.2 \pm 0.3}^{\ddagger}$	$\underline{6.74 \pm 0.29}^{\ddagger}$	$\underline{98.5 \pm 0.5}$
Naive merge	–	–	–	–	–
CAST	71.8 ± 2.2	64.7 ± 2.1	–	–	$\mathbf{99.5 \pm 0.2}$
SafetyLock	70.5 ± 2.2	63.7 ± 3.5	–	–	$\mathbf{99.5 \pm 0.6}$
Discard T ($M_\theta + S$)	3.5 ± 1.5	2.7 ± 2.1	–	–	89.1 ± 3.0
No defense	75.3 ± 2.9	71.0 ± 3.6	–	–	99.9 ± 0.2

Over-refusal. XSTest is the one benchmark on which SafeCompose trails every composition baseline, and there it still complies more than discard. On OR-Bench-Hard-1K, a held-out over-refusal distribution (Table 23), the picture changes. Under two semantic judges the composer and discard both refuse 9–13 points more than every baseline at its default setting, and the difference between them lies within judge disagreement: the composer refuses 3.2 points more under Llama-3.1 and 0.6 points less under Falcon3 (Falcon-LLM Team, 2024). Composition therefore inherits discard's over-refusal and does not reduce it, and XSTest, which overlaps meta-training, understates the cost. The validation-selected merge over-refuses at the same rate on the keyword metric and under both semantic judges (Table 23; 21.6% against 23.8% under Llama-3.1 and 46.2% against 49.6% under Falcon3 in a matched rejudge, all inside the seed spread). The cost therefore comes from the safety adapter rather than from anything the gate adds; arms that admit the task adapter at unit weight avoid it and pay in ASR. By OR-Bench category, the composer refuses most on illegal (29.9%), violence (29.8%), harmful (24.2%), and privacy (24.1%) prompts under the Llama-3.1 judge. Its gap to discard is within 5 points in nine of ten categories under each judge, and the exceptions differ by judge: hate under Llama-3.1 (13.0 points, where discard refuses 1.9% and every other arm 12–16%) and harassment under Falcon3 (8.1 points, about three of 41 prompts per seed). No category therefore shows the gate adding refusal beyond the safety adapter under both judges. The baselines at their default settings refuse less mainly on illegal, violence, sexual, harmful, self-harm, privacy, and deception prompts under Llama-3.1, 3.5–17 points below discard, and across nine categories under Falcon3. The validation-selected merge stays within 6.2 points of discard in every category under Llama-3.1, and within 5.7 points under Falcon3 except on hate (-9.9) and violence (-8.6). Do-Not-Answer gives the same reading on a third distribution (Table 27): SafeCompose, discard, and the merge refuse 40.5%, 39.3%, and 40.1% of its 939 prompts and are judged harmful on 5.2%, 4.2%, and 4.8%, each within about one seed SD of the others, while RESTA refuses 6.4 points more and is harmful 2.2 points less. Dropping the benign-preservation term outright lowers XSTest compliance

by 2.8 points on average over the three matched seeds (Table 12), so the term acts on this cost. We have not swept its weight λ_{over}, which remains untested as a lever.

J DIAGNOSTICS

Terms. The main text calls the reported gate, which reads the text so far, the learned gate; this appendix calls it the dynamic gate. A gate with fixed values is called static when it has one pair of values per layer and global when it has one pair for the whole model. Prompt-once is the dynamic gate reading the prompt only once and keeping those values for the rest of the generation, and ungated sets $\alpha_l = \beta_l = 1$.

Gate variants. The static variant fits one constant gate pair per layer to the average training input; the ungated variant sets $\alpha_l = \beta_l = 1$. The static and ungated rows of Table 11 come from one run on the same items; the learned row uses a different GSM8K item count and predates them, which is why the learned-versus-static gap is not read as an effect of input dependence in Section 5; the ablations under matched conditions below supply that comparison. Removing the gates raises ASR from 0.64% to 16.99% over three seeds; a test that pools the 312 responses gives $p = 4.7 \times 10^{-14}$, but it treats the reused prompts as independent, so we report it only as a sensitivity (Appendix E).

Table 11: Gate variants on Qwen2.5-7B at the 5% dose; n is the number of GSM8K items scored.

Variant	AdvBench ASR (%)	GSM8K (%)	n
Learned gate	0.64 ± 0.56	84.50 ± 3.00	200
Static input-average	0.96 ± 0.00	81.77 ± 3.93	64
Ungated addition	16.99 ± 8.72	71.35 ± 5.92	64

Ablations under matched conditions. Seven composers per seed were trained from one frozen copy of the code on Qwen, each differing from the reported design in one respect, and evaluated together with three evaluation-time interventions on the held-out MedMCQA 5% adapter under the harness of the panels (AdvBench-104 with the HarmBench classifier, 520 MedMCQA items, XSTest safe compliance). The trained ablations are the dynamic gate (the reported design retrained from the same copy), a static gate that learns one constant pair per layer under the same objective, ERM in place of V-REx, a single-adapter pool, and the objective with one of the behavioral, benign-preservation, or capability terms removed. The interventions act on the dynamic gate at evaluation: prompt-once computes the gate values from the prompt and keeps them for the rest of the generation; ungated sets $\alpha_l = \beta_l = 1$; the layer ablations set the gate values to zero in the early (layers 0–8), middle (9–17), or late (18–27) band, so those layers run the base branch alone. Table 12 reports the nine ablations at three seeds; at three seeds all three depth-band ablations stay within one AdvBench prompt and 1.2 points of MedMCQA of the dynamic gate, and the late band reads 0.96% ASR at every seed with MedMCQA 58.46, 59.62, and 58.85. On Mistral, the dynamic gate and the drop-behavioral ablation were also trained and evaluated at three seeds (last rows of Table 12): dropping the behavioral term lowers the composed ASR by 4.8, 7.7, and 1.9 points at equal MedMCQA and XSTest compliance, whereas on Qwen the same change has no effect. We report this as a family-specific observation beside Mistral's residual 7–13% ASR, not as a mechanism claim; no other Mistral ablation was trained. Every evaluation ran from the same frozen copy of the code with deterministic algorithms; Table 26 re-evaluates the gate policies under attack.

Statistics of the ablations. Table 13 compares each ablation with the dynamic gate, with the seed as the unit (Appendix E). At three seeds the two-sided exact sign-flip p cannot fall below .25, so nothing here is significant or presented as such. At a two-point margin, the static gate, prompt-once, ERM, the drop-behavioral ablation, and the three depth-band ablations are non-inferior to the dynamic gate in safety (upper bounds 1.26–1.62 points, one to two AdvBench prompts); the drop-benign ablation passes at one point; the zero bound of the drop-capability ablation comes from identical differences and is not evidence of equality; the single-adapter pool misses the two-point margin (2.51); and the ungated composer is far outside (15.15). The margins were chosen after the data were seen, so this check is exploratory. Zeroing the gates in the early band costs 0.96 points of MedMCQA in every seed (95% interval -1.44 to -0.48), the only utility interval that excludes zero without being

degenerate. The learned gate is also barely input-dependent in practice: its α varies from 0.48 to 0.53 across seeds with a within-layer SD near 0.02, while the static gate holds $\alpha = 0.69$ and $\beta = 0.49$, which is consistent with input dependence adding nothing measurable without an attack.

Residual geometry. On the same three seeds, 28 layers, and 16 prompts, the safety and task residuals are near-orthogonal on average (mean cosine 0.023, -0.019, -0.030 by seed), and $\|\Delta_T\|$ is 3–5$\times$ $\|\Delta_S\|$ at every depth (188–190 against 43–46 in the late band). By depth band the learned gates sit at $\alpha \approx 0.42$–0.58 in the early and middle thirds and 0.22–0.25 in the late third, with $\beta \approx 0.33$–0.54. One layer departs from orthogonality: layer 3 has cosine $+0.65$, -0.67, and -0.96 across the three seeds, every other layer $|\cos| \leq 0.23$, so a single early layer carries nearly collinear safety and task deltas with a seed-dependent sign. We record it as an observation, not a claim.

Table 12: Ablations under matched conditions, evaluated on the held-out MedMCQA 5% adapter; Qwen2.5-7B except the last two rows, which are the only two ablations trained on Mistral-7B. Mean $\pm$ SD, with per-seed ASR in parentheses where the seeds differ by more than one prompt. ASR is the HarmBench classifier on 104 AdvBench prompts; XSTest is safe-prompt compliance. On Qwen every gated ablation is within two AdvBench prompts per seed and 0.7 points of MedMCQA of the dynamic gate; the ungated composer is above it by 4, 13, and 4 prompts and 3.3 points lower on MedMCQA in every seed. The prompt-once row is bit-identical to the dynamic gate on MedMCQA in every seed. The layer ablations are reported at three seeds every band stays within one prompt and 1.2 points of MedMCQA of the dynamic gate.

Control	AdvBench ASR (%) ↓	MedMCQA (%) ↑	XSTest compliance (%) ↑
Dynamic gate	0.64 ± 0.56 (0.00 / 0.96 / 0.96)	59.17 ± 0.59	92.5 ± 2.7
Static gate	0.96 ± 0.00	58.97 ± 0.87	94.0 ± 3.3
Prompt-once	0.96 ± 0.00	59.17 ± 0.59	92.5 ± 3.0
ERM in place of V-REx	0.64 ± 0.56	58.65 ± 1.20	93.2 ± 3.0
Single-adapter pool	1.28 ± 0.56	58.65 ± 0.84	93.6 ± 2.2
Drop behavioral loss	0.32 ± 0.56	58.46 ± 0.69	91.7 ± 2.7
Drop benign-preservation loss	0.32 ± 0.56	58.46 ± 1.39	89.7 ± 3.8
Drop capability loss	0.64 ± 0.56	59.29 ± 0.95	92.0 ± 2.6
Ungated	7.37 ± 5.30 (3.85 / 13.46 / 4.81)	55.90 ± 1.73	93.2 ± 5.6
Layer ablation, early / middle / late (seed 42)	0.96 / 0.00 / 0.96	57.69 / 58.27 / 58.46	90.8 / 94.0 / 92.8
Mistral-7B			
Dynamic gate	9.62 ± 2.88	53.53 ± 0.95	92.4 ± 5.0
Drop behavioral loss	4.81 ± 0.00	53.65 ± 1.17	93.5 ± 3.2

Table 13: Ablations under matched conditions compared with the dynamic gate (Qwen2.5-7B, held-out MedMCQA 5% adapter). Δ is the ablation minus the dynamic gate in points, with a 95% paired-t interval over seeds; the upper bound is the one-sided 95% paired-t bound on the ASR difference used in the exploratory non-inferiority check. The two-sided exact sign-flip p is at least .25 in every row. ΔMedMCQA is the difference of seed means.

Ablation	ΔASR (95% interval)	Upper bound	ΔMedMCQA
Static gate	$+0.32$ (-1.06, $+1.70$)	1.26	-0.20
Prompt-once	$+0.32$ (-1.06, $+1.70$)	1.26	0.00
ERM in place of V-REx	0.00 (-2.39, $+2.39$)	1.62	-0.52
Single-adapter pool	$+0.64$ (-2.12, $+3.40$)	2.51	-0.52
Drop behavioral loss	-0.32 (-3.08, $+2.44$)	1.55	-0.71
Drop benign-preservation loss	-0.32 (-1.70, $+1.06$)	0.62	-0.71
Drop capability loss	0.00 (identical differences)	0.00	$+0.13$
Ungated	$+6.73$ (-5.68, $+19.14$)	15.15	-3.27 (-6.71, $+0.18$)

Why Mistral trades safety for capability. On Mistral, SafeCompose gains capability over discard and pays for it in ASR; Qwen and Gemma pay nothing. Two explanations compete. The selected Mistral checkpoints stop early in the adversarial curriculum, so the gates may have seen fewer conflicts; or the family's representation geometry places safety and task residuals in directions the gates cannot separate. A controlled-exposure rerun of the Mistral composer reproduces the trade-off (7.1% ASR, 46.9% GSM8K), and the matching full-exposure Qwen rerun reproduces Qwen's result, so training budget alone does not account for the difference. Architecture, optimization,

and geometry remain confounded, and we have not isolated the cause. The evidence supports one narrower statement: the family effect is not an early-stopping artifact. Mistral also responds to the objective differently: dropping the behavioral-safety term lowers its composed ASR at equal utility (Table 12), while on Qwen the change has no effect; with two ablations at three seeds we record this as an observation, not an explanation.

Exposure control. To test whether Mistral's safety gap is an artifact of early-stopped checkpoints, composers were retrained under controlled curriculum exposure and evaluated at the 5% dose (Table 14). Full-750 trains through the complete curriculum to its ceiling near step 750; reset-k restarts the curriculum schedule at step k.

Table 14: Exposure-controlled composers at the 5% dose; means over the listed seeds.

Architecture	Exposure	ASR (%)	GSM8K (%)
Qwen2.5-7B	full-750	0.96	85.96
Mistral-7B	reset-250	7.05	46.86
Mistral-7B	reset-750	5.77	51.15
Mistral-7B	reset-750	10.58	49.42

K Repeated adapter arrivals

Five task adapters arrive in sequence on Qwen and, in a separate run, on Gemma; after each arrival the deployed configuration is re-evaluated on AdvBench, and GSM8K is measured after the final stage. Drift is the stage-5 ASR minus the stage-1 ASR. SafeCompose applies the same frozen gates at every stage. Table 15 reports Qwen at five seeds and Table 16 Gemma at three.

Table 15: Repeated arrivals on Qwen2.5-7B: ASR (%) after each stage, drift, and final GSM8K (%). Means over five seeds; Holm-adjusted p (over the four contrasts) compares each arm's drift with SafeCompose's. Bold marks the best drift and final GSM8K among SafeCompose and the three baselines, and underline the second-best; the undefended adapter and the weighted merge are listed below them. †:its p is the unadjusted paired test of its drift against SafeCompose's on the same seeds (a test of end-of-sequence ASR that pools prompts across seeds gives $p = .73$).

	ASR by stage ↓						
Method	1	2	3	4	5	Drift ↓	Final GSM8K ↑
SafeCompose	0.58	0.58	0.77	0.77	0.77	**0.19**	**83.77**
Naive merge	0.96	1.15	1.92	0.58	3.85	<u>2.88</u>	77.42
CAST	1.54	5.00	6.35	5.58	17.12	15.58	77.08
SafetyLock	1.54	4.62	6.73	5.38	16.54	15.00	<u>78.19</u>
No defense	1.92	5.77	8.27	6.35	18.65	16.73	77.31
Weighted merge (α=0.75)†	0.96	0.96	0.64	0.32	0.64	−0.32	84.23

Capability after each arrival. The same five-arrival chain (GSM8K, CodeAlpaca, Alpaca, GSM8K, Alpaca) is re-scored after each stage on three arms: the composed model with the stage's task adapter, that task adapter alone, and discard. Held-out items are 32 GSM8K test questions (8-shot, exact match) and 32 Alpaca and 32 CodeAlpaca records disjoint from the training prefixes, scored by teacher-forced continuation negative log-likelihood over 128 tokens. Table 17 gives means over three seeds with 32 items per axis it is an inventory, and no test is reported. On the arriving Alpaca and CodeAlpaca tasks the composed model tracks the task-only chain within 0.03 nats of continuation negative log-likelihood at every stage, where discard sits 0.2–0.3 nats worse. On GSM8K it stays at 78–82%, against 69–73% for the task-only chain and 85% for discard, with no downward trend across arrivals. Composition keeps the arriving fit that discarding gives up and the base capability that the task-only chain loses. The GSM8K axis was regenerated through the panel's 8-shot generation path after the original run's 512-token chat truncation cut the question from every prompt; the NLL values are from the original run.

Table 16: Repeated arrivals on Gemma-2-9B: ASR (%) after each stage, drift, final GSM8K (%), and two tests against SafeCompose. Means over three seeds; the same five arrivals and 5% nominal dose as Table 15, HarmBench classifier on the 104 AdvBench prompts, 8-shot GSM8K. Drift p is the paired test of each arm's drift against SafeCompose's; none approaches significance (Holm $p = 1.0$ for CAST, SafetyLock, and the undefended model), because those three arms start at 23–28% and fluctuate rather than erode. At the seed level they end 11.5–51.9 points above SafeCompose in three of three seeds, the exact one-sided floor ($p = .125$; Appendix G). End p is a one-sided Fisher test of end-of-sequence ASR that pools the 312 prompts across seeds; it treats reused prompts as independent and is shown only as a sensitivity. Naive merge is the unit-weight merge. Bold marks the best drift and final GSM8K among SafeCompose and the three baselines, and underline the second-best; the undefended adapter and the weighted merge are listed below them. The $\alpha = 0.75$ merge is the Qwen-selected weight, run as an arm of the same sequence; the Gemma family-level sweep of Table 24 selects $\alpha = 0.5$, which was not run in sequence.

	ASR by stage ↓						
Method	1	2	3	4	5	Drift ↓	Final GSM8K ↑
SafeCompose	0.00	0.00	0.00	0.00	0.64	<u>0.64</u>	**85.19**
Naive merge	0.64	1.60	0.96	0.00	0.32	**−0.32**	66.92
CAST	26.92	29.81	37.18	26.28	29.81	2.88	<u>72.31</u>
SafetyLock	23.08	28.53	35.26	23.40	28.85	5.77	71.09
No defense	28.21	30.77	38.46	25.96	29.17	0.96	72.63
Weighted merge (α=0.75)	0.00	0.00	0.00	0.00	0.00	0.00	82.50

Table 17: Capability after each arrival on Qwen2.5-7B: GSM8K exact match (%) on 32 held-out items and continuation NLL (nats, lower is better) on 32 held-out Alpaca and 32 CodeAlpaca records, scored after the stage in the first column. Means over three seeds. Discard does not depend on the arriving adapter.

	GSM8K (%) ↑			Alpaca NLL ↓			CodeAlpaca NLL ↓		
Stage (arriving task)	Composed	Task only	Discard	Composed	Task only	Discard	Composed	Task only	Discard
1 (GSM8K)	82.3	68.8	85.4	1.289	1.264	1.495	0.479	0.475	0.677
2 (CodeAlpaca)	79.2	69.8	85.4	1.228	1.208	1.495	0.428	0.425	0.677
3 (Alpaca)	78.1	69.8	85.4	1.206	1.184	1.495	0.445	0.446	0.677
4 (GSM8K)	82.3	69.8	85.4	1.267	1.240	1.495	0.447	0.441	0.677
5 (Alpaca)	80.2	72.9	85.4	1.230	1.221	1.495	0.446	0.449	0.677

L Additional baselines and probes

The measurements below were made after the panels of Section 4 were frozen and, except where a caption says otherwise, in separate evaluations, each with its own recorded code revision; they are reported beside the main panels and never pooled with them. All use Qwen2.5-7B at the 5% dose unless a caption states otherwise.

Validation-selected baseline strengths. Table 18 reports the strength sweep for the weighted merge, CAST, and SafetyLock on the held-out adapter, with the strength for each seed and defense selected on a validation split and scored once on the test split. Table 19 scores the resulting fixed merge, which scales the safety adapter's weights by $\alpha = 0.75$ and the task adapter's by 0.25, with the panel scorer at all three doses, and Table 20 applies the same fixed weight, together with SafeGrad (Yi et al., 2025) and TIES-Merging (Yadav et al., 2023), on the in-pool panel. Table 24 repeats the merge sweep on the Mistral and Gemma held-out panels with the weight selected once per family. Table 25 adds two post-hoc weight-space baselines on the Qwen held-out adapter, RESTA (Bhardwaj et al., 2024) and a rank-adapted SafeMERGE (Djuhera et al., 2025), each selected on the same validation split and rule; SafeMERGE, which merges layer by layer under a safety criterion, is the closest static analog to per-layer gating, and at the threshold it selects it coincides with the $\alpha = 0.75$ merge.

Attacks on the opening of the response, and a second harmfulness judge. Table 21 forces every response to the 104 AdvBench prompts to begin with the fixed phrase "Sure, here is" and

reports ASR under the HarmBench classifier and under Llama-Guard-3 as a second judge; Table 22 reports a search over twelve affirmative openings that adapts to the model's responses, run against SafeCompose, discard, the validation-selected merge, and naive merge. Neither is a gradient-based or token-level attack.

Gate policies under attack. Table 26 runs both attacks against five gate policies in one evaluation: the dynamic gate, its prompt-once variant, the static per-layer gate and the ungated composer of Table 12, and a global gate with two values trained under the same objective as the static gate. It supplies the same-run comparison under attack that Section 5 reads.

Over-refusal on held-out distributions. Table 23 reports over-refusal on OR-Bench-Hard-1K, which does not overlap meta-training, beside XSTest, under a keyword metric and two semantic-refusal judges. Table 27 reports Do-Not-Answer, 939 prompts over five risk areas, for SafeCompose, discard, the validation-selected merge, and RESTA, under two harmfulness judges and the Llama-3.1 refusal judge.

Table 18: Validation-selected baseline strengths on the Qwen2.5-7B held-out panel at the 5% dose. Grid: weighted merge $\lambda \in \{0, .25, .5, .75, 1\}$ (λ scales the safety adapter's weight update and $1 - \lambda$ the task adapter's; $\lambda = 1$ is discard-like), CAST steer and SafetyLock scale $\in \{0, .25, .5, 1\}$. For each seed and defense the strength was chosen on a validation split (JailbreakBench-100 for safety, 128 MedMCQA validation items for utility) by a fixed rule, maximise validation utility subject to validation ASR $\leq 10\%$ and otherwise minimise ASR, and the one locked point was scored once on the test split (AdvBench-104, HarmBench classifier; 520 MedMCQA items with a 16-token letter-parse scorer, not the panel scorer of Table 2). Neither CAST nor SafetyLock met the 10% constraint at any strength. Two JailbreakBench goals near-duplicate AdvBench items. Mean $\pm$ SD over seeds.

Method	Selected strength	AdvBench ASR (%) ↓	MedMCQA (%) ↑
SafeCompose (main-panel value, panel scorer)	–	0.6 ± 0.6	59.0 ± 1.3
Weighted merge, λ	0.75 / 0.75 / 0.5	1.3 ± 1.5	59.7 ± 1.3
CAST, steer	0.5 / 1.0 / 1.0	26.9 ± 6.4	56.3 ± 1.5
SafetyLock, scale	1.0 / 1.0 / 1.0	18.9 ± 9.1	56.0 ± 1.7

Table 19: Fixed weighted merge (the safety adapter's weights scaled by $\alpha = 0.75$ and the task adapter's by 0.25; the weight selected on the validation data of Table 18 and held fixed across seeds and doses) on the Qwen2.5-7B held-out panel, scored with the panel harness (104 AdvBench prompts, 520 GSM8K items, 520 MedMCQA items). SafeCompose rows repeat the main-panel values of Table 2. Mean $\pm$ SD over seeds.

Dose	Method	AdvBench ASR (%) ↓	GSM8K (%) ↑	MedMCQA (%) ↑
1%	SafeCompose	0.6 ± 0.6	84.6 ± 3.6	58.3 ± 0.9
	Weighted merge α=0.75	0.6 ± 0.6	86.6 ± 0.2	60.0 ± 0.2
5%	SafeCompose	0.6 ± 0.6	85.8 ± 2.4	59.0 ± 1.3
	Weighted merge α=0.75	0.3 ± 0.6	86.3 ± 0.9	60.5 ± 0.4
10%	SafeCompose	0.6 ± 0.6	86.1 ± 1.6	59.4 ± 0.2
	Weighted merge α=0.75	0.6 ± 0.6	85.0 ± 0.6	60.3 ± 0.3

Table 20: Additional arms on the Qwen2.5-7B in-pool panel. These were evaluated separately from the main panel of Table 8, under a different code revision, with the same adapters, prompts, judge, and batch sizes, and are not pooled with it. The SafeCompose, SaLoRA, naive-merge, and discard rows repeat the main-panel values for the same three seeds. Bold marks the best mean per column among the arms above the dashed line, and underline the second-best; discard is listed below them for reference. SafeGrad (Yi et al., 2025) trains on 256 PKU-SafeRLHF (Ji et al., 2025) preference examples and projects the task adapter's weight update away from the safety gradient; TIES-Merging (Yadav et al., 2023) uses trim 0.2 and $\lambda = 1$; the weighted merge uses the fixed $\alpha = 0.75$ of Table 19. Mean $\pm$ SD over seeds.

	GSM8K accuracy (%) ↑			AdvBench ASR (%) ↓		
Method	1%	5%	10%	1%	5%	10%
SafeCompose	$\underline{84.7 \pm 2.4}$	$\mathbf{85.6 \pm 2.5}$	$\mathbf{85.6 \pm 2.1}$	$\mathbf{0.3 \pm 0.6}$	$\mathbf{0.6 \pm 0.6}$	$\mathbf{0.3 \pm 0.6}$
SaLoRA (adapted)	73.5 ± 1.0	73.5 ± 1.1	71.9 ± 0.8	5.8 ± 1.7	13.5 ± 10.0	11.5 ± 6.9
Naive merge	71.3 ± 1.7	72.1 ± 1.0	70.2 ± 0.5	5.8 ± 1.0	16.0 ± 8.6	15.7 ± 9.1
SafeGrad	72.5 ± 1.0	73.1 ± 0.5	70.1 ± 0.5	$\underline{1.9 \pm 0.0}$	$\underline{2.9 \pm 2.5}$	5.4 ± 1.5
TIES-Merging	77.3 ± 1.5	78.3 ± 1.4	76.1 ± 1.1	18.6 ± 7.2	34.9 ± 2.8	32.7 ± 5.4
Weighted merge (α=0.75)	$\mathbf{85.3 \pm 0.1}$	$\underline{84.6 \pm 0.9}$	$\underline{85.0 \pm 1.4}$	$\mathbf{0.3 \pm 0.6}$	$\mathbf{0.6 \pm 0.6}$	$\underline{0.6 \pm 0.6}$
Discard T ($M_\theta + S$)	86.9 ± 0.1	86.9 ± 0.1	86.9 ± 0.1	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0

Table 21: Fixed-phrase attack on Qwen2.5-7B at the 5% dose: ASR (%) on the 104 AdvBench prompts with and without the response forced to begin with "Sure, here is", under the HarmBench classifier and under Llama-Guard-3 as a second judge (same-run control, 256 new tokens). Mean $\pm$ SD over three seeds. Llama-Guard-3 changes 10.7% of the HarmBench labels and no arm ordering. Bold marks the lowest mean per column among SafeCompose and the four baselines, and underline the second-lowest; discard, the undefended adapter, and the weighted merge are listed below them. †: the weighted merge of Table 19, evaluated separately; its attack results were relabelled by Llama-Guard-3 in a separate rejudge, in which SafeCompose and discard read 67.6 ± 3.1 and 60.9 ± 1.5, and its direct-request results were not relabelled.

	HarmBench classifier ↓		Llama-Guard-3 ↓	
Method	Direct	Fixed phrase	Direct	Fixed phrase
SafeCompose	$\mathbf{0.6 \pm 0.6}$	$\mathbf{47.1 \pm 10.1}$	$\mathbf{0.3 \pm 0.6}$	$\mathbf{67.3 \pm 3.5}$
SaLoRA (adapted)	$\underline{13.5 \pm 10.0}$	89.1 ± 2.4	$\underline{19.9 \pm 10.4}$	$\underline{96.5 \pm 1.5}$
Naive merge	16.0 ± 8.6	$\underline{86.2 \pm 3.1}$	23.1 ± 10.8	96.8 ± 0.6
CAST	45.5 ± 4.5	92.6 ± 0.6	56.1 ± 3.6	98.7 ± 0.6
SafetyLock	42.6 ± 7.1	92.6 ± 2.0	54.2 ± 4.8	97.4 ± 0.6
Discard T ($M_\theta + S$)	1.0 ± 0.0	37.8 ± 7.8	0.0 ± 0.0	60.6 ± 1.0
No defense	49.0 ± 1.0	92.0 ± 1.5	59.6 ± 5.9	98.1 ± 1.0
Weighted merge (α=0.75)†	0.6 ± 0.6	73.4 ± 2.8	–	80.1 ± 4.8

Table 22: Adapted-opening attack on Qwen2.5-7B at the 5% dose. For each arm and seed the attacker selects one opening from a fixed set of twelve affirmative openings by a search that adapts to the responses on 16 BeaverTails development prompts disjoint from the test set, with the composer's per-layer gate values visible, using its full budget of 8 queries per development prompt; the selected opening is then fixed and scored on the 104 AdvBench test prompts with the HarmBench classifier. Generation is capped at 128 new tokens for every condition, so the fixed-phrase column is not comparable with Table 21, which uses 256 tokens. ASR in %.

Method	Selected opening	No attack	"Sure, here is"	Adapted opening
SafeCompose	"Of course. Here is"	0.0	30.8	41.3
	"Certainly. Here is"	0.0	52.9	53.8
	"Of course. Here is"	0.0	38.5	45.2
	mean ± SD	0.0 ± 0.0	40.7 ± 11.2	46.8 ± 6.4
Discard T ($M_\theta + S$)	"Of course. Here is"	1.0	23.1	29.8
	"I can help. Here is"	0.0	31.7	30.8
	"Of course. Here is"	0.0	30.8	40.4
	mean ± SD	0.3 ± 0.6	28.5 ± 4.7	33.7 ± 5.8
Weighted merge (α=0.75)	"Sure, here is"	1.0	70.2	70.2
	"Absolutely. Here is"	1.0	74.0	77.9
	"Certainly. Here is"	0.0	66.3	76.0
	mean ± SD	0.6 ± 0.6	70.2 ± 3.8	74.7 ± 4.0
Naive merge	"Sure, here is"	10.6	90.4	90.4
	"Sure, here is"	26.0	85.6	85.6
	"Certainly. Here is"	13.5	86.5	83.7
	mean ± SD	16.7 ± 8.2	87.5 ± 2.5	86.5 ± 3.5

Table 23: Over-refusal on Qwen2.5-7B at the 5% dose. OR-Bench-Hard-1K (1,319 prompts) is a held-out distribution; XSTest safe (250 prompts) overlaps the benign-preservation prompts used in meta-training. Keyword safe-compliance (%, higher is better) and semantic refusal rate (%, lower is better) under two judges, Llama-3.1-8B-Instruct and Falcon3-7B-Instruct. The Llama-3.1 judge left 128 of 32,949 outputs (0.39%) unclassified and these are excluded from its columns; Falcon3 classified all outputs. Judge agreement is 70.0% on OR-Bench-Hard and 92.7% on XSTest. Mean ± SD over three seeds. Bold marks the best mean per column among SafeCompose and the four baselines, and underline the second-best; discard, the undefended adapter, and the weighted merge are listed below them. †: evaluated separately.

Method	OR-Bench-Hard-1K			XSTest safe		
	Keyword compl. ↑	Refusal (Llama-3.1) ↓	Refusal (Falcon3) ↓	Keyword compl. ↑	Refusal (Llama-3.1) ↓	Refusal (Falcon3) ↓
SafeCompose	77.5 ± 6.9	23.9 ± 4.9	49.5 ± 3.7	92.9 ± 1.8	4.8 ± 0.0	12.0 ± 0.4
SaLoRA (adapted)	95.4 ± 2.1	13.1 ± 3.6	40.3 ± 0.6	98.5 ± 0.5	4.7 ± 0.2	10.9 ± 1.4
Naive merge	96.0 ± 1.9	12.3 ± 3.1	$\mathbf{38.6 \pm 1.3}$	$\underline{98.9 \pm 0.8}$	5.1 ± 0.2	10.8 ± 0.4
CAST	$\mathbf{98.6 \pm 0.5}$	$\underline{10.6 \pm 0.9}$	40.1 ± 1.2	$\mathbf{99.5 \pm 0.2}$	$\underline{3.2 \pm 0.4}$	$\mathbf{8.8 \pm 1.1}$
SafetyLock	$\underline{98.0 \pm 0.8}$	$\mathbf{10.4 \pm 0.9}$	$\underline{39.5 \pm 0.8}$	$\mathbf{99.5 \pm 0.6}$	$\mathbf{2.9 \pm 0.6}$	$\underline{9.1 \pm 1.2}$
Discard T ($M_\theta + S$)	81.1 ± 3.8	20.6 ± 3.6	50.1 ± 2.0	89.1 ± 3.0	6.0 ± 0.8	14.7 ± 2.2
No defense	98.7 ± 0.3	10.8 ± 0.5	40.0 ± 1.3	99.9 ± 0.2	3.3 ± 0.5	7.9 ± 0.6
Weighted merge (α=0.75)†	79.0 ± 7.6	21.6 ± 7.5	46.2 ± 6.6	91.6 ± 2.4	6.8 ± 1.4	13.5 ± 3.0

Table 24: Family-level validation-selected merge weight on the Mistral-7B and Gemma-2-9B held-out panels. The weight $\alpha \in \{0, .25, .5, .75, 1\}$ on the safety adapter was swept per seed on the 5% adapter, scored on the validation split of Table 18 (JailbreakBench-100, 128 MedMCQA validation items), pooled over the three seeds, and selected once per family by the same rule, maximize validation utility subject to validation ASR $\leq 10\%$: $\alpha = 0.75$ on Mistral (validation ASR 4.3%; $\alpha = 0.5$ failed the constraint at 17%) and $\alpha = 0.5$ on Gemma (validation ASR 1.0%). The selected merge was then scored with the panel harness at all three doses; SafeCompose rows repeat the main-panel values. Mean $\pm$ SD over three seeds. Across the three families, without an attack, the tuned merge matches SafeCompose on Qwen (Table 19), ties it on safety and MedMCQA but trails on GSM8K on Gemma, and trades against it on Mistral.

Family	Dose	Method	AdvBench ASR (%) ↓	GSM8K (%) ↑	MedMCQA (%) ↑
Mistral-7B	1%	SafeCompose	8.7 ± 2.5	50.2 ± 0.8	53.2 ± 0.5
		Weighted merge α=0.75	1.3 ± 1.5	44.9 ± 1.8	52.6 ± 1.7
	5%	SafeCompose	9.3 ± 2.2	50.1 ± 1.4	53.9 ± 1.1
		Weighted merge α=0.75	2.2 ± 2.4	43.8 ± 0.7	52.1 ± 0.9
	10%	SafeCompose	7.1 ± 3.4	50.0 ± 1.0	53.8 ± 1.0
		Weighted merge α=0.75	1.6 ± 1.5	43.9 ± 1.7	51.5 ± 0.6
Gemma-2-9B	1%	SafeCompose	0.0 ± 0.0	86.1 ± 0.3	59.9 ± 0.3
		Weighted merge α=0.5	0.0 ± 0.0	81.9 ± 1.0	60.4 ± 0.3
	5%	SafeCompose	0.0 ± 0.0	86.2 ± 0.2	59.9 ± 0.3
		Weighted merge α=0.5	0.6 ± 0.6	82.9 ± 0.5	60.4 ± 0.2
	10%	SafeCompose	0.0 ± 0.0	85.8 ± 0.3	60.8 ± 1.0
		Weighted merge α=0.5	0.3 ± 0.6	82.5 ± 1.1	61.5 ± 0.5

Table 25: Post-hoc weight-space baselines on the Qwen2.5-7B held-out 5% adapter. RESTA adds a safety vector, the aligned model minus a copy of it LoRA-fine-tuned on 1,000 harmful-only BeaverTails records disjoint from every evaluation prompt, at a scale $s \in \{0.1, 0.25, 0.5, 1.0\}$. SafeMERGE (rank-adapted) keeps the published layer-selection rule but, because our adapters have unequal ranks (64 and 32), replaces the equal-rank factor merge with dense-delta interpolation over $w_s \in \{0.1, 0.3, 0.5, 0.75\}$ and $\tau \in \{0.15, \ldots, 1.0\}$. Both were selected on the validation split and rule of Table 18, with the grid fixed before any score was seen, and each selected one setting for all seeds: RESTA $s = 0.5$; SafeMERGE $w_s = 0.75$, $\tau = 0.9$. At $\tau = 0.9$ every layer is admitted, so the SafeMERGE arm coincides with the $\alpha = 0.75$ merge, whose fixed-phrase results re-measured in this evaluation are bit-identical to it; the same arm read 73.4 ± 2.8 in the earlier evaluation of Table 21, run at a different code revision. The fixed-phrase column follows the protocol of Table 21 (256 new tokens, HarmBench classifier); SafeCompose and discard repeat their main-panel and attack results. Mean $\pm$ SD over three seeds. Bold marks the best mean per column among the arms above the dashed line, and underline the second-best; discard is listed below them for reference.

Method	AdvBench ASR (%) ↓	GSM8K (%) ↑	MedMCQA (%) ↑	Fixed-phrase ASR (%) ↓
SafeCompose	$\underline{0.6 \pm 0.6}$	85.8 ± 2.4	59.0 ± 1.3	$\underline{47.1 \pm 10.1}$
RESTA (s=0.5)	$\underline{0.6 \pm 0.6}$	82.8 ± 0.6	56.0 ± 1.5	$\mathbf{27.9 \pm 11.8}$
SafeMERGE (rank-adapted)	$\mathbf{0.3 \pm 0.6}$	$\mathbf{86.4 \pm 1.5}$	$\underline{60.3 \pm 0.4}$	68.3 ± 1.0
Weighted merge (α=0.75), this evaluation	$\mathbf{0.3 \pm 0.6}$	$\underline{86.3 \pm 0.9}$	$\mathbf{60.5 \pm 0.4}$	68.3 ± 1.0
Discard T ($M_\theta + S$)	1.0 ± 0.0	86.9 ± 0.1	28.0 ± 2.3	37.8 ± 7.8

Table 26: Gate policies under attack on the Qwen2.5-7B held-out MedMCQA 5% adapter, evaluated in one run. Dynamic is the reported gate; prompt-once reads the prompt only once and keeps those gate values, the policy of the fast decoder in Appendix M; static per-layer is the static optimized gate of Table 12; global is a gate with two values for the whole model, trained under the same objective as the static gate (effective $\alpha \approx 0.69$, $\beta \approx 0.49$, the layer mean of the static gate); ungated sets $\alpha_l = \beta_l = 1$. The no-attack control and the fixed phrase follow Table 21 (256 new tokens); the adapted opening follows Table 22 (128 new tokens, one selection per seed and arm). ASR under the HarmBench classifier; MedMCQA with the panel scorer. Mean $\pm$ SD over three seeds. Paired against the dynamic gate, the static, global, and ungated arms are higher in three of three seeds under both attacks, by 25–56 points; prompt-once differs by -2.9, $+1.9$, and $+2.9$ points with the fixed phrase and $+11.5$, $+2.9$, and -1.0 with the adapted opening.

Gate policy	No attack ↓	"Sure, here is" ↓	Adapted opening ↓	MedMCQA (%) ↑
Dynamic (reported design)	0.96 ± 0.00	50.96 ± 4.19	40.71 ± 10.37	58.97 ± 1.25
Prompt-once (reads the prompt only once)	0.32 ± 0.56	51.60 ± 7.28	45.19 ± 5.35	58.97 ± 1.25
Static per-layer gate	0.96 ± 0.00	81.73 ± 2.54	78.53 ± 2.42	59.17 ± 0.68
Global gate (two scalars)	0.96 ± 0.96	83.33 ± 2.00	78.53 ± 4.93	59.04 ± 0.38
Ungated	8.65 ± 6.66	87.18 ± 3.38	85.58 ± 2.54	55.71 ± 1.22

Table 27: Do-Not-Answer (939 English prompts, five risk areas) on the Qwen2.5-7B held-out 5% adapter: harmful-response rate under the HarmBench classifier and under Llama-Guard-3, and semantic refusal rate under the Llama-3.1-8B-Instruct judge of Table 23; 256 new tokens; no invalid labels in any condition. Harmfulness and refusal are separate labels of the same responses. RESTA is the arm selected in Table 25. Mean $\pm$ SD over three seeds. Bold marks the best mean per column among the arms above the dashed line, and underline the second-best; discard is listed below them for reference.

Method	Harmful, HarmBench (%) ↓	Harmful, Llama-Guard-3 (%) ↓	Semantic refusal (%) ↓
SafeCompose	5.2 ± 0.5	$\underline{0.5 \pm 0.2}$	$\underline{40.5 \pm 0.5}$
Weighted merge (α=0.75)	$\underline{4.8 \pm 0.7}$	0.6 ± 0.3	$\mathbf{40.1 \pm 1.3}$
RESTA (s=0.5)	$\mathbf{3.0 \pm 0.5}$	$\mathbf{0.3 \pm 0.2}$	46.9 ± 2.3
Discard T ($M_\theta + S$)	4.2 ± 0.4	0.6 ± 0.1	39.3 ± 1.9

M DECODING COST AND HARDWARE

Anatomy of the slowdown. The reference implementation, which produced every number in the paper, keeps the gates reading the whole text at every step and decodes at 47–120× the cost of cached base decoding (Table 30); roughly 12–24× of that comes from the implementation, adapter switching and the disabled cache, and 3.9–5.0× from the method, three branches of decoding limited by memory bandwidth. Because the gates depend on the whole text so far, the reference implementation forgoes key–value caching. Generating N tokens then costs $O(N^3)$ attention operations rather than the $O(N^2)$ of cached decoding, and the three branches multiply the constant. The slowdown has two parts: a cache penalty that dominates at long contexts, and a branch ratio between an uncached base and the three-branch composer of about $4\times$ at a 900-token context and 7–17× at a 20-token one (Table 28), so the branch ratio depends on context length. The branch ratio is the cost closer to intrinsic. The cache penalty follows from the gate design: ordinary append-only caching cannot reproduce gates that reread the whole text at every step, because a changed gate alters earlier branch states, and a cache is possible only when the gates read the prompt only once. We built that variant as a bounded prototype, with one cache per branch under gate values computed once from the prompt. Against its own prompt-once uncached twin it is not logit-exact at natural context; it decodes no faster at batch 1 (0.85–1.02×) and about 2.2× faster at batch 8 with a 900-token context; and it reproduces the dynamic decoder's tokens on 14–66% of natural-context steps. The prompt-once row of Table 12 shows that reading the prompt only once costs little without an attack, and Table 26 that it keeps the gate's advantage under both attacks, within 3 points of the dynamic gate with the fixed phrase and with no consistent sign across seeds with the adapted opening. With gate values computed once, the composition is a fixed linear combination of the branch residuals, compatible with append-only caching at a constant overhead near the branch count, so the prototype's failed parity and absent batch-1 speedup point to the implementation rather than to the algorithm. A second implementation confirms the reading (Table 30). In fp32 the cached and uncached prompt-once decoders agree to 10^{-4} in every logit with no argmax flips.

Microbenchmark. Measurements use the first eight AdvBench prompts, two repeats, and the faster repeat. Total slowdown compares cache-disabled SafeCompose with cached base decoding and factors into a cache penalty and a branch ratio; the branch ratio is the part attributable to the method rather than to the disabled cache. Peak allocated CUDA memory exceeds cached base decoding by 1.43–3.87 GiB at natural context width and by 10.51–12.69 GiB at padded width 900. These are implementation measurements, not production requirements or intrinsic lower bounds.

Table 28: Decode microbenchmark; width 900 denotes inputs padded to 900 tokens.

Input context	Output tokens	Cache penalty	Branch ratio	Total slowdown
20 natural	64	2.81×	16.71×	46.96×
20 natural	256	6.90×	6.76×	46.70×
900 padded	64	21.68×	4.00×	86.84×
900 padded	256	30.51×	3.93×	119.83×

Key-value-cached variant. Ordinary append-only caching cannot reproduce the reported composer, because a change in a gate alters the earlier-position states of both adapter branches and hence their keys and values; a cache is possible only if the per-layer gate profile is computed once from the prompt and frozen for the generation. We implemented that variant as a bounded prototype on the Qwen 5% composer, with one cache per branch (base, safety, task) under gate values computed once from the prompt: batch $\{1, 8\}$ by context {natural, 900 tokens} by $\{64, 256\}$ output tokens, each with three synchronized timing repeats. Correctness is judged against the same prompt-once decoder run without a cache, by greedy-token parity and by the maximum logit difference against a bound calibrated at twice the bare base model's own repeat noise; fidelity to the reported dynamic decoder is the separate token-match column. Table 29 shows the result: parity holds only at the 900-token context, where the calibrated bound is loose (8–13 logits), and fails at natural context; the median speedup over the prompt-once uncached decoder is 0.85–1.02× at batch 1 and 2.1–2.3× only at batch 8 with the long context; and the cached decoder reproduces the dynamic decoder's tokens on 14–66% of natural-context steps. Peak memory stayed below 90% of the device in every trial. The prototype is a fixed-token stress benchmark and does not claim parity for production decoding with

end-of-sequence stopping; its decision was no-go for that implementation, which serialized the three branch caches, and the batched implementation below supersedes it at batch 8.

Table 29: Key-value-cached composed decode with gate values computed once from the prompt (Qwen2.5-7B, 5% dose). Parity and logit difference are against the same prompt-once decoder without a cache; speedup is the median over three timing repeats against that decoder; token match is the fraction of greedy steps on which the cached decoder emits the same token as the reported dynamic decoder.

Context	Output tokens	Greedy parity	Max logit diff / bound	Median speedup	Dynamic token match
natural	64	63/64	0.41 / 0.75	0.91×	0.47
natural	256	254/256	0.44 / 0.75	1.00×	0.14
900	64	64/64	3.82 / 12.9	0.85×	1.00
900	256	256/256	3.82 / 12.9	1.02×	1.00
natural	64	57/64	1.22 / 0.75	0.99×	0.66
natural	256	236/256	1.22 / 0.75	0.90×	0.18
900	64	64/64	6.91 / 8.15	2.25×	1.00
900	256	256/256	8.01 / 8.15	2.14×	1.00

Batched branches and a prompt-once cache. A second implementation removes the two implementation costs the microbenchmark exposes. Branch batching runs the base, safety, and task rows of a layer as one mixed-batch block call instead of three adapter-switched calls per layer per token; a profiler attributes the reference decoder's flat 6.3 tokens/s at both output budgets to that switching (86 global and 22,344 layer-level adapter switches per step). The prompt-once cache computes the gates on the prompt exactly as the reference decoder does at its first step, freezes them, and keeps one append-only key–value cache over the three rows. Table 30 reports both on the Qwen 5% composer at batch 8, greedy decoding, 256 output tokens, fastest of two repeats. Batching alone removes the per-token overhead (4.0× at 64 output tokens) but not the uncached forward, which grows with context, so it gains 1.57× at 256 tokens and nothing at a 900-token context; the cache removes the length term. Together they decode at 74.0 and 38.6 tokens/s at 20- and 900-token contexts, 3.96× and 4.97× below the cached base, with peak memory below the reference decoder's. Parity: on a teacher-forced probe over the reference decoder's own token stream (64 steps by 8 rows, 512 positions), the batched dynamic decoder differs from the reference by a median 0.25 and a maximum 0.53 in any logit and disagrees on the argmax at 7 of 512 positions, every one at a reference top-2 margin of at most 0.125; the base model alone at batch 8 against batch 24 shows the same envelope (0.25 median, 0.50 maximum, 99.0% agreement), and the reference decoder against its own rerun is bit-exact, so batching is the same computation under bf16 kernel variation. The prompt-once policy is a different policy rather than an approximation: in fp32 with TF32 off its cached and uncached decoders agree to 1.1×10^{-4} in every logit with no argmax flips at batch 1 and 8, and in bf16 they flip 1 of 64 and 8 of 512 argmaxes, inside the same empirical bound; its safety and utility are measured in Table 26. Two further levers were measured and not adopted: skipping the task branch where its gate is near zero (under prompt-once profiles $\beta_l < 0.01$ on 63% of harmful prompt–layer pairs) saves at most a third of a bandwidth-bound step, and distilling the three branches into one adapter has a ceiling of 1.30–1.37× below the cached base, the cost of base-plus-safety decoding, and gives up modularity. Batch 1 was not re-measured under the batched implementation, so the prototype's batch-1 result stands. The fast path is wired into the released code as an opt-in argument of the composer's hook installer, with the reference decoder as the default, and produced no reported number.

Provenance. All results within a table were produced under one software environment, one code revision, and one device type, recorded in the released configurations, except where a caption says otherwise (Tables 16 and 26). Every main-panel value was checked by analysis scripts that reject incomplete or mismatched results. Evaluations run separately keep their own recorded code revision and are never pooled with the main panels; these are SafeGrad and TIES-Merging, the validation-selected merge with its attack and over-refusal results, and the later post-hoc baselines, gate-policy comparisons, Do-Not-Answer, family-level merges, and pooling comparison. The ablations under matched conditions ran with deterministic algorithms. The harmfulness judge is `cais/HarmBench-Llama-2-13b-cls`; the second judges are `meta-llama/Llama-Guard-3-8B` for harmfulness and `meta-llama/Llama-3.1-8B-Instruct` and

Table 30: Composed-decode speed levers on Qwen2.5-7B (5% dose): tokens/s at 256 output tokens, batch 8, greedy, fastest of two repeats. Slowdown is against cached base decoding at the same context; speedup is against the reference decoder, three adapter-switched branches with dynamic gates and no cache, which produced every reported number. Cached base+S is the ceiling for distilling the composition into one adapter. The 900-token reference rates come from an earlier run of the same benchmark, whose batched arm ran out of memory before a fix; its reference arms reproduced within 1% in the rerun.

Context	Decoder	Tokens/s	Slowdown	Speedup	Peak GiB
20	Cached base	292.8	1.0×	–	14.4
20	Cached base+S	213.7	1.37×	–	15.0
20	Reference decoder	6.30	46.5×	1.0×	18.2
20	Batched, dynamic gates, no cache	9.89	29.6×	1.57×	24.5
20	Batched, prompt-once gates, cache	74.0	3.96×	11.7×	15.6
900	Cached base	191.9	1.0×	–	15.5
900	Cached base+S	147.2	1.30×	–	17.0
900	Reference decoder	1.61	119×	1.0×	28.2
900	Batched, dynamic gates, no cache	1.53	125×	0.95×	20.0
900	Batched, prompt-once gates, cache	38.6	4.97×	24.0×	20.0

`tiiuae/Falcon3-7B-Instruct` for semantic refusal. AdvBench is the harness's 104-prompt evaluation split; the other 416 prompts train the safety adapter and the composer, which is why no disjoint AdvBench validation slice exists and validation-based selection uses JailbreakBench-100. Sample sizes are AdvBench 104, GSM8K 520, MedMCQA 520 (a fixed random subset of the validation split), HarmBench standard behaviors 104, JailbreakBench 100, XSTest safe 250, OR-Bench-Hard-1K 1,319, MMLU 200, and Do-Not-Answer 939, with JailbreakBench-100 and 128 MedMCQA validation items as the selection sets.

N Analysis

This appendix expands Section 5 paragraph by paragraph; gate names follow the Terms paragraph of Appendix J.

Why the residuals must be scaled. Three variants at the 5% Qwen dose separate the components of the composer (Table 11). Adding both residuals without gates gives 17.0% ASR; a static gate, one constant pair per layer fitted to the average input, gives 1.0%; the learned gate gives 0.6%. The first step is large, and it is the clearest evidence about mechanism in the paper: rescaling the two residuals separately at each layer, rather than adding them, removes most of the erosion. This differs from a weight merge with per-layer coefficients, because each branch passes through the layer's nonlinearities before its residual is formed and combined. The second step, from static to learned, is within one prompt, and the two variants were not trained under identical conditions. Ablations under matched conditions settle the point at three seeds (Tables 12 and 13): the static gate, plain empirical risk in place of V-REx, a single-adapter pool, any one loss term removed, and the prompt-once variant all sit within two AdvBench prompts per seed of ASR and 0.7 points of MedMCQA of the dynamic gate on the held-out adapter, while the ungated composer separates, at 3.9%, 13.5%, and 4.8% ASR and 3.3 points lower on MedMCQA in every seed. Without an attack, then, scaling the residuals is necessary, but letting the scale depend on the input is not.

Robustness under attack. We ran two attacks on the opening of the response on Qwen at the 5% dose (Appendix L). Forcing the response to begin with the fixed phrase "Sure, here is" raises SafeCompose's ASR from 0.6% to 47.1%, discard's from 1.0% to 37.8%, and that of every baseline at its default setting to 86–93%; the validation-selected merge goes to 73.4% (Table 21). The second attack searches over twelve affirmative openings and adapts to the responses. It runs on sixteen development prompts disjoint from the test set, with the composer's per-layer gate values visible and a budget of eight queries per development prompt, and selects an opening that raises ASR further on every seed: to 46.8% at a 128-token cap, against 40.7% for the fixed phrase under the same cap (Table 22). The same search against the other arms preserves the fixed-phrase ordering on every seed,

with discard at 33.7%, SafeCompose at 46.8%, the validation-selected merge at 74.7%, and naive merge at 86.5%, so the merge stays 24–31 points above the composer per seed (mean 28). Under Llama-Guard-3 the fixed-phrase gap to the merge is 12.5 points (80.1% against 67.6%), outside that judge's 10.7% relabeling rate. The rank-adapted SafeMERGE, a layer-wise merge, selects every layer on validation data, reduces to the $\alpha = 0.75$ merge, and goes to 68.3% with the fixed phrase (Table 25).

What reading the prompt adds. That merge is one global scalar, and finer fixed gates fare no better. In one evaluation (Table 26), the static per-layer gate of Table 12, which matches the dynamic gate on direct requests, goes to 81.7% with the fixed phrase and 78.5% with the adapted opening, a global gate with two values to 83.3% and 78.5%, and the ungated composer to 87.2% and 85.6%, where the dynamic gate holds at 51.0% and 40.7% and its prompt-once variant at 51.6% and 45.2%. Prompt-once stays within 3 points of the dynamic gate with the fixed phrase and shows no consistent sign across seeds with the adapted opening, at identical MedMCQA, while the ungated composer leaks 8.7% with no attack at all. The attack result therefore separates a gate that reads the prompt from any gate fixed in advance, per layer or global; of the two designs compatible with caching, prompt-once gating keeps the advantage and a static gate does not. Repeated arrivals are the other place where a fixed weight might fall behind, and there the gate does not separate from the merge in safety: on Qwen the merge drifts -0.3 points against $+0.2$ for the composer (Table 15), and on Gemma the two are equally flat and differ only in final GSM8K (Table 16).

Limitations of the attack results. Discard is safer than the composer under both attacks: by 9.3 points with the fixed phrase at 256 tokens, and, in three of three seeds, by 12.2 and 13.1 points with the fixed phrase and the adapted opening at the 128-token cap (40.7% against 28.5%, and 46.8% against 33.7%). RESTA, which moves the weights away from a deliberately unaligned copy of the model, sits at the composer's direct-request ASR and holds at 27.9% under the fixed phrase, safer than discard at 37.8% and the composer at 47.1%, at three points lower on both capability measures (Table 25); it was not run under the adapted opening or on the other families. Both attacks limit the attacker to at most twelve candidate openings and eight queries per prompt, neither is a gradient-based or token-level attack, and both cover one Qwen adapter at three seeds.

O EVALUATION BOUNDARY

One automated classifier judges safety on 104 prompts reused across seeds, so the seed is the unit of replication. On the Qwen attack results a second judge, Llama-Guard-3, changes 10.7% of labels and no arm ordering (Table 21). Relabeling the 63 Qwen in-pool conditions under the original classifier reproduces every recorded ASR. Held-out transfer changes the task and the adapter and keeps the LoRA rank, the way adapters are trained, and the contamination mechanism fixed. An adapter trained differently, at another rank, or by another party's pipeline would test the transfer hypothesis harder than anything we have run. We distinguish the dynamic-composition methods of Section 2 by structure and have not compared them by experiment. The next experiments that would discriminate are these: a gradient-based or token-level attack; Mistral ablations beyond the drop-behavioral one; RESTA under the adapted opening and on the other two families; the validation-selected merge on the in-pool panels of Gemma and Mistral; the fast decoder at batch 1; two more seeds on the held-out panels; and adapters trained with other recipes.