HYPERSAFE: INFERENCE-TIME SAFETY RECOVERY FOR FINE-TUNED LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Safety alignment in large language models can be fragile under fine-tuning, as even benign task adaptation may increase harmful compliance. Existing defenses either retrain or modify weights during or after fine-tuning, which requires control over training or trades safety against utility, or use model-agnostic guard classifiers, which are not tailored to a given checkpoint and run a separate large model on every prompt. We propose HyperSafe, a post-hoc, model-specific, and non-invasive framework that restores safety by generating a Safe Side Network (SSN) for each fine-tuned checkpoint. A hypernetwork maps layer-wise activation fingerprints, computed from a small set of calibration prompts, to the SSN parameters in a single forward pass. The SSN runs alongside the frozen fine-tuned model as a prompt-level safety classifier: harmful prompts are routed to refusal, while safe prompts are answered by the original model. HyperSafe targets safety degradation from benign fine-tuning rather than adversarial jailbreaks. The hypernetwork is trained once per backbone family; each new checkpoint then requires no gradient updates, no safety data, and no weight modification. On Qwen2-7B and LLaMA-3-8B, HyperSafe reduces harmful response rates from 19–31% to below 1% on every held-out checkpoint (confirmed on LLaMA-3-8B by a second, independent judge), loses less than 1% task accuracy on average, and adds only 3.4% inference FLOPs.

1 INTRODUCTION

Fine-tuning safety-aligned LLMs is a common way to adapt them to downstream tasks, but it can also weaken their safety alignment. Prior work has shown that fine-tuning can compromise safety even when the fine-tuning data is entirely benign (Qi et al., 2024); in our experiments, LoRA fine-tuning on benign task data raises average harmful rates from under 5% to 19–31% (Table 2). One possible reason is that safety alignment can depend heavily on the model's behavior in the first few output tokens (Qi et al., 2025); narrow fine-tuning can even induce broadly misaligned behavior (Betley et al., 2025). As a result, even benign fine-tuning updates may change the model's refusal behavior and reduce its ability to reject harmful requests.

Existing defenses fall into three families (Section 2; Table 1). *Fine-tuning-stage* methods constrain task adaptation, but require control over the fine-tuning process and cannot be applied to checkpoints that have already been released. *Post-fine-tuning weight-modification* methods repair a released checkpoint by projecting, pruning, or merging its weights. Some need the aligned base model at deployment, and their safety–utility balance depends on the method and its operating point. In our experiments, OneShot SFT (Zhang et al., 2026) lowers average task accuracy by 15.3 points on LLaMA-3-8B, while Antidote (Huang et al., 2025) and linear merging (Farn et al., 2025) maintain accuracy within approximately two points of the fine-tuned model but retain 4–9% harmful responses on BeaverTails and HEx-PHI (Section 4.3). *Model-agnostic guard classifiers* (Inan et al., 2023; Han et al., 2024) leave the model untouched, but are not tailored to how a given checkpoint was adapted and run a separate multi-billion-parameter model on every prompt.

Based on these limitations, we ask the following question:

> *Can we recover the safety behavior of the original aligned model for a given fine-tuned checkpoint using a model-specific lightweight post-hoc method, without retraining, modifying the deployed weights, or degrading downstream task performance?*

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.

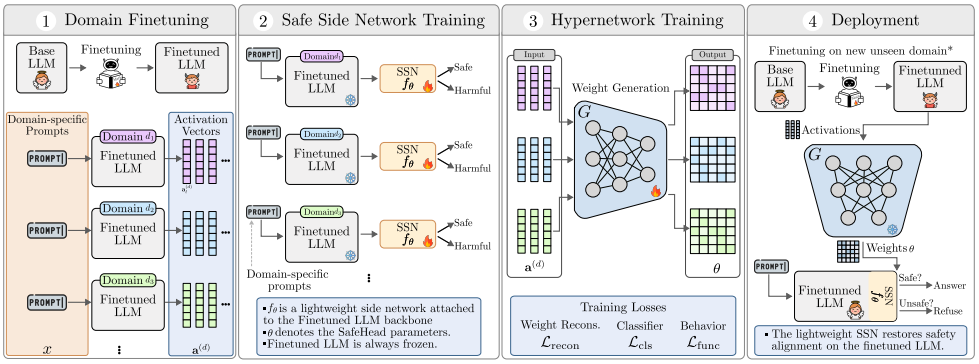

Figure 1: HyperSafe pipeline. Training (one-time per backbone family): (1) fine-tune the base model on each training domain, (2) train a domain-specific SSN per fine-tuned checkpoint, and (3) train the hypernetwork to map activation fingerprints to SSN weights. Deployment (per checkpoint): (4) extract calibration activations and generate a model-specific SSN in a single forward pass.

Such a method should work without control over fine-tuning, leave the deployed weights unchanged, and be specific to the checkpoint. We attach a small *Safe Side Network* (SSN) that runs alongside the frozen model and decides whether to answer or refuse, and use a hypernetwork to generate its weights from the checkpoint's activation fingerprints, avoiding per-checkpoint SSN training.

We propose HyperSafe (Figure 1). Given a fine-tuned checkpoint, it extracts layer-wise activation fingerprints from a small set of calibration prompts, generates the SSN weights with one hypernetwork forward pass, and attaches the SSN to the frozen model; prompts scored as harmful are routed to a refusal, and all others are answered by the original model. HyperSafe is thus checkpoint-specific, post-hoc, and non-invasive. Training the hypernetwork is a one-time cost per backbone family that requires safety data; deploying on each new checkpoint requires neither.

Contributions. We summarize our main contributions as follows:

1. We formulate post-hoc safety restoration as the activation-conditioned generation of SSNs. Given a fine-tuned checkpoint, our method generates a model-specific SSN from its activation fingerprints, without additional gradient updates or safety data at deployment time.
2. We design HyperSafe, a hypernetwork-based framework that predicts the weights of a lightweight SSN. To make this prediction feasible, HyperSafe uses layer-wise activation conditioning, direction-magnitude decomposition, and factorized weight generation.
3. We show on Qwen2-7B and LLaMA-3-8B that the generated SSN restores refusal behavior on held-out fine-tuned checkpoints while preserving downstream task performance: harmful response rates fall from 19–31% to below 1% with less than 1% average loss in task accuracy. It outperforms a shared SSN trained on the same data, and attains lower harmful rates than weight-modification and guard-model baselines at their evaluated operating points while matching the aligned model's over-refusal.

Table 1: Comparison with existing defenses against fine-tuning-induced safety degradation. *Per-checkpoint cost*: work required for each new fine-tuned model (excluding one-time costs such as guard-model training or HyperSafe meta-training). *Model-aware*: tailored to the target checkpoint. *Weights kept*: deployed weights unchanged. *No* $\mathcal{M}_0$: aligned base model not needed at deployment.

Method	**Per-checkpoint cost**	**Model-aware**	**Weights kept**	**No $\mathcal{M}_0$**
Vaccine (Huang et al., 2024c)	full training	✓	×	✓
SafeInstr (Bianchi et al., 2024)	partial training	✓	×	✓
Safe LoRA (Hsu et al., 2024)	projection	✓	×	×
Linear merging (Farn et al., 2025)	interpolation	✓	×	×
Antidote (Huang et al., 2025)	pruning (safety data)	✓	×	✓
OneShot (Zhang et al., 2026)	1-example SFT	✓	×	✓
Llama Guard 4 (Meta AI, 2025)	none	×	✓	✓
HyperSafe (ours)	1 hypernet. forward pass	✓	✓	✓

2 Related Work

Safety degradation under fine-tuning. LLMs are aligned with supervised fine-tuning and RLHF (Christiano et al., 2017; Stiennon et al., 2020; Ouyang et al., 2022; Bai et al., 2022a), Constitutional AI (Bai et al., 2022b), or DPO (Rafailov et al., 2023). This alignment is shallow: refusal is concentrated in the first few output tokens (Qi et al., 2025) and in sparse, low-rank weight regions (Wei et al., 2024). Qi et al. (2024) showed that fine-tuning, even on benign data, compromises safety (see also Fraser et al., 2025), and fine-tuning on harmful data removes safety training efficiently even for 70B and proprietary models (Lermen et al., 2023; Zhan et al., 2024); narrow fine-tuning can also produce broad misalignment (Betley et al., 2025).

Defenses against fine-tuning-induced degradation. Following the stage-based taxonomy of Huang et al. (2026), we group defenses by when they act. *(i) Alignment- and fine-tuning-stage defenses* shape training so that later fine-tuning does less damage: perturbation-aware alignment (Vaccine (Huang et al., 2024c), Booster (Huang et al., 2024a)), representation noising (Rosati et al., 2024), security vectors (Zhou et al., 2024), and constrained or safety-augmented fine-tuning (Lisa (Huang et al., 2024b), SafeInstr (Bianchi et al., 2024), BackdoorAlign (Wang et al., 2024), token-constrained SFT (Qi et al., 2025)). They assume control over training and do not apply to checkpoints that are already released. *(ii) Post-fine-tuning weight repair* edits a released checkpoint: projection onto a safety subspace (Safe LoRA (Hsu et al., 2024)), task-vector arithmetic and merging (Task Arithmetic (Ilharco et al., 2023), TIES (Yadav et al., 2023), RESTA (Bhardwaj et al., 2024), linear merging (Farn et al., 2025)), pruning of harmful parameters (Antidote (Huang et al., 2025)), post-hoc perturbation (Panacea (Wang et al., 2025)), and one-shot re-alignment (OneShot (Zhang et al., 2026)). These methods add no inference cost, but change the deployed weights, expose a coefficient that sets the safety–utility balance, and in some cases need the aligned base model. *(iii) Guard models* filter inputs or outputs without inspecting the target model (Llama Guard (Inan et al., 2023), ShieldGemma (Zeng et al., 2024), WildGuard (Han et al., 2024), AEGIS (Ghosh et al., 2024), Granite Guardian (Padhi et al., 2024), and production moderation systems (Markov et al., 2023; Lees et al., 2022)); a related line probes a model's own hidden states (Azaria & Mitchell, 2023; Zou et al., 2023a; Arditi et al., 2024; Jiang et al., 2025). HyperSafe combines properties of (ii) and (iii): like weight repair it is specific to the target checkpoint, and like guard models it leaves the deployed weights unchanged. Unlike hidden-state probes, which are trained per model, its classifier is generated for a new checkpoint without training.

Hypernetworks and side networks. Hypernetworks (Ha et al., 2017; Chauhan et al., 2024) generate the weights of a target network conditioned on context, with applications in continual learning (von Oswald et al., 2020), weight-space learning (Navon et al., 2023), parameter-efficient fine-tuning (Phang et al., 2023; Karimi Mahabadi et al., 2021), and zero-shot model construction (Knyazev et al., 2021). Liang et al. (2025) scaled the paradigm to LLMs, generating LoRA parameters from text prompts for zero-shot *utility* adaptation. HyperSafe instead generates weights of a Ladder Side-Tuning network (Sung et al., 2022) from *model activations*, and to our knowledge is the first to combine hypernetworks with side networks for safety restoration.

3 Method

HyperSafe attaches a lightweight SSN to the frozen backbone for prompt-level safety classification, and predicts its weights with a hypernetwork from activation fingerprints of the target model. The hypernetwork is supervised with pairs of fine-tuned models and their SSNs across many domains, and at inference produces an SSN for a checkpoint from an unseen domain.

3.1 Problem Formulation

Let $\mathcal{M}_0$ be a safety-aligned base LLM and $\mathcal{M}_d$ a fine-tuned version of $\mathcal{M}_0$ on dataset d. Fine-tuning may weaken or alter the safety behavior of $\mathcal{M}_d$, leading to compliance with harmful prompts that $\mathcal{M}_0$ would refuse. We seek a generator G that, given any fine-tuned model $\mathcal{M}_{d^*}$ from a potentially unseen dataset, produces an SSN f_θ parameterized by the generated weights θ:

$$\theta = G\big(\mathcal{A}(\mathcal{M}_{d^*})\big), \qquad \lambda = f_\theta\big(x\,;\,\mathcal{M}_{d^*}\big), \tag{1}$$

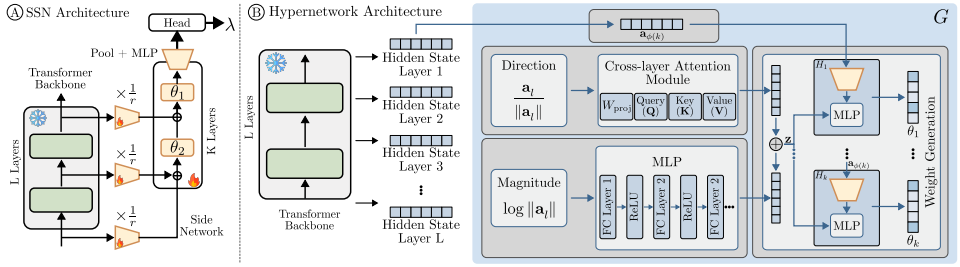

Figure 2: SSN architecture. The frozen backbone's hidden states are downsampled via gated ladder connections into a lightweight side network (3–4% of backbone parameters). The SSN produces a safety score λ that routes safe prompts to the base model output and harmful prompts to a refusal response.

where x denotes the input prompt, $\mathcal{A}(\cdot)$ denotes activation fingerprints extracted from a small calibration set (Eq. 4), θ are the SSN parameters predicted by G, and $\lambda \in [0, 1]$ is a safety score ($\lambda \approx 0$: safe; $\lambda \approx 1$: harmful). The SSN reads backbone activations of $\mathcal{M}_{d^*}$ when scoring x, so λ depends on both the prompt and the target model.

3.2 Safe Side Network Architecture

The SSN f_θ attaches to the frozen backbone as a lightweight branch that predicts a safety score for each prompt and determines whether the model should produce a response or refuse. We build on the Ladder Side-Tuning (LST) architecture (Sung et al., 2022) and extend it for safety classification. Figure 2 illustrates the overall design.

Side Transformer. The SSN is implemented as a small Transformer that processes ladder-fused representations derived from the backbone hidden states. It consists of K layers with hidden dimension $h_s = h/r$ for backbone hidden size h and reduction factor r, adding 3–4% of backbone parameters at the 7–8B scale we evaluate (per-architecture values in Appendix A, Table 7).

Ladder Connections. Each side layer k fuses downsampled backbone hidden states with side hidden states via a per-layer gate:

$$\mathbf{h}_k^{\text{in}} = \mu_k \cdot W^{\text{down}}\, \mathbf{h}_{\phi(k)}^{\text{back}} + (1-\mu_k) \cdot \mathbf{h}_{k-1}^{\text{side}} \tag{2}$$

where $\phi(k)$ maps side layer k to a backbone layer, $\mathbf{h}_{\phi(k)}^{\text{back}}$ denotes the corresponding backbone hidden state, $\mathbf{h}_{k-1}^{\text{side}}$ denotes the SSN hidden state from the previous layer, and $\mu_k = \sigma(g_k/\tau)$ is a per-layer gate. The scalar $g_k \in \mathbb{R}$ is one of the parameters generated by the hypernetwork (see Section 3.3); τ is a fixed temperature hyperparameter.

Selective Routing. After the final side layer, hidden states are mean-pooled and passed through an MLP to produce λ. At inference, prompts with $\lambda \geq 0.5$ receive a refusal response, while the remaining prompts are passed to the unmodified fine-tuned model. The SSN therefore never alters the output for prompts it deems safe, which is why task accuracy is preserved.

3.3 Activation-Conditioned Hypernetwork

The hypernetwork G takes the activation fingerprint of a fine-tuned model as input and predicts the SSN parameters. After describing the fingerprint, we introduce two design choices that make this high-dimensional prediction feasible: direction–magnitude decomposition and layer-wise factorized generation.

Activation Fingerprint Extraction. For each model $\mathcal{M}_d$, we run N_{cal} domain-specific calibration prompts and average the backbone hidden state at each layer $l = 1, \ldots, L$ after the final token:

$$\mathbf{a}_l = \frac{1}{N_{\text{cal}}} \sum_{i=1}^{N_{\text{cal}}} \mathbf{h}_l(x_i) \in \mathbb{R}^h. \tag{3}$$

These L vectors form a compact behavioral fingerprint of how fine-tuning shifted the model's representations:

$$\mathcal{A}(\mathcal{M}_d) = \{\mathbf{a}_l\}_{l=1}^{L}. \tag{4}$$

Direction-Magnitude Decomposition. To separate directional changes in representation space from their scale, each activation vector is split into a unit direction and log-magnitude:

$$\hat{\mathbf{d}}_l = \frac{\mathbf{a}_l}{\|\mathbf{a}_l\|}, \quad m_l = \log \|\mathbf{a}_l\|. \tag{5}$$

The direction captures the nature of the representational change, while the magnitude reflects its strength. We then compute the vectors:

$$\mathbf{z}_{\text{dir}} = \text{Attn}\big(\{W_{\text{proj}}\, \hat{\mathbf{d}}_l\}_{l=1}^{L}\big), \tag{6}$$

$$\mathbf{z}_{\text{mag}} = \text{MLP}\big(\{m_l\}_{l=1}^{L}\big), \tag{7}$$

where W_{proj} is a learnable projection matrix. The fused representation $\mathbf{z} = [\mathbf{z}_{\text{dir}};\, \mathbf{z}_{\text{mag}}]$ aggregates information across all layers and feeds the weight-generation modules.

Layer-wise Factorized Generation. We implement the hypernetwork G as a global encoder that produces the embedding $\mathbf{z}$ together with a set of layer-wise generators $\{H_k\}_{k=1}^{K}$ that predict the parameters of each side layer. Rather than predicting all parameters jointly, we generate each side layer's weights independently from the corresponding backbone activation:

$$\theta_k = H_k(\mathbf{a}_{\phi(k)}, \mathbf{z}), \tag{8}$$

where $\phi : \{1, \ldots, K\} \to \{1, \ldots, L\}$ maps each side layer k to its corresponding backbone layer. Each H_k is a lightweight MLP that fuses the layer-specific activation $\mathbf{a}_{\phi(k)}$ with the global cross-layer embedding $\mathbf{z}$. The modules H_k share most parameters across layers but include layer-specific adapters, reducing the effective output dimensionality per prediction.

Classifier Generation. The SSN classifier head is generated by a separate branch that applies cross-layer attention to all fingerprint layers, followed by an MLP with a low-rank output factorization (Appendix A).

3.4 Training and Deployment

Training proceeds in two stages: (1) learning domain-specific SSNs to construct supervision targets, and (2) training the hypernetwork to predict these weights from activation fingerprints.

Ground-truth SSN Collection. We collect supervised pairs $\{(\mathcal{M}_d, \theta_d^*)\}_{d=1}^{D}$ by training one SSN per dataset. For each dataset d, we first fine-tune the base model $\mathcal{M}_0$ using LoRA (Hu et al., 2022) and merge the adapter to obtain $\mathcal{M}_d$. We then freeze $\mathcal{M}_d$ and attach a randomly initialized SSN. The SSN is trained on category-balanced harmful prompts from the BeaverTails training split (Ji et al., 2023) ($\sim$7k prompts across 14 harm categories, shared across checkpoints) together with in-domain safe prompts from that checkpoint's own fine-tuning dataset (Appendix A). Training uses a selective loss: harmful prompts receive both language-modeling loss and classification loss, while safe prompts receive only classification loss. This encourages the SSN to learn when to intervene without degrading task performance on safe inputs. All SSNs share the same initialization, ensuring weight-space alignment for hypernetwork supervision.

Hypernetwork Training. The hypernetwork is trained on D_{train} datasets using three complementary losses, weighted by coefficients β_{recon}, β_{cls}, and β_{func}:

$$\mathcal{L} = \beta_{\text{recon}}\, \mathcal{L}_{\text{recon}} \;+\; \beta_{\text{cls}}\, \mathcal{L}_{\text{cls}} \;+\; \beta_{\text{func}}\, \mathcal{L}_{\text{func}}. \tag{9}$$

$\mathcal{L}_{\text{recon}}$ is the mean-squared error between the generated and ground-truth SSN weights $\hat{\theta}$ and θ^*, averaged over the generated parameter blocks. $\mathcal{L}_{\text{cls}}$ and $\mathcal{L}_{\text{func}}$ are binary cross-entropy losses on the predicted routing score $\hat{\lambda}(x)$ against the harmful flag $m \in \{0, 1\}$, evaluated respectively on the calibration set $\mathcal{C}_d$ used to extract the activation fingerprint and on a held-out probe set $\mathcal{P}_d$ disjoint from $\mathcal{C}_d$; the latter verifies that the generated classifier generalizes beyond the calibration prompts the hypernetwork conditioned on. The N_{cal}=50 calibration prompts are domain-specific and disjoint

from $\mathcal{P}_d$. We set $(\beta_{\text{recon}}, \beta_{\text{cls}}, \beta_{\text{func}}) = (1, 10, 5)$ during development and keep them fixed across architectures, splits, and ablations: reconstruction is auxiliary because matching the target weights does not by itself guarantee correct classification, while the classification loss directly governs the answer/refuse decision (Appendix A).

Deployment. The two training stages above, together with fine-tuning the training-domain checkpoints, are paid once per backbone family and require safety data and gradient updates. Given a new fine-tuned model $\mathcal{M}_{d^*}$ of that family: (1) extract layer-wise activations from N_{cal} calibration prompts; (2) run one forward pass through the hypernetwork to obtain SSN weights; (3) load the SSN alongside $\mathcal{M}_{d^*}$. This per-checkpoint step requires no safety data, gradients, or model modification.

4 EXPERIMENTS

Models and domains. We evaluate on Qwen2-7B-Instruct (Yang et al., 2024) and LLaMA-3-8B-Instruct (Grattafiori et al., 2024) across 22 datasets spanning QA, commonsense reasoning, math, knowledge, classification, code, and instruction-following (Appendix B). For each dataset, we train an independent LoRA adapter (rank 16, α=32) and merge it into a separate copy of the base model, yielding one fine-tuned checkpoint per dataset; adapters are never merged with one another.

Safety benchmarks and judges. BeaverTails (Ji et al., 2023) (700 evaluation prompts, disjoint from the split that supplies SSN training data), **AdvBench** (Zou et al., 2023b) (520), and **HEx-PHI** (Qi et al., 2024) (330) measure harmful rates; the latter two are never used in training. The primary judge is beaver-dam-7b (Ji et al., 2023); since it was trained on BeaverTails data, we re-score all LLaMA-3-8B outputs with the independently developed WildGuard (Han et al., 2024), whose documentation does not list BeaverTails among its training sources. We measure over-refusal on XSTest (Röttger et al., 2024) and OR-Bench (Cui et al., 2025), and robustness to prompt transformations on JailbreakBench (Chao et al., 2024), WildJailbreak (Jiang et al., 2024), and DAN-style role-play templates (Shen et al., 2024).

Evaluation protocol. Each of three held-out splits (Appendix C) removes four datasets from different task categories; the hypernetwork is trained on the other 18 and evaluated on the four unseen ones. We report **Task Accuracy** (official test split, or a fixed-seed 20% holdout when none exists) and **Harmful Rate** (% of harmful prompts yielding harmful output). We compare against the aligned **base model**, the undefended **fine-tuned model**, and OneShot (Zhang et al., 2026) on all 12 held-out checkpoints. On Split A of LLaMA-3-8B, we additionally compare with Antidote (Huang et al., 2025) (Wanda-score pruning, α=0.2), linear merging (Farn et al., 2025) (equal interpolation of aligned and fine-tuned weights, λ=0.5), SafeLoRA (Hsu et al., 2024), and the guard models Llama Guard 3/4 (Grattafiori et al., 2024; Meta AI, 2025) and WildGuard.

4.1 MAIN RESULTS

Fine-tuning raises the average harmful rate from under 5% (base) to 19–31% (Table 2), most on instruction-following and commonsense domains. HyperSafe reduces the harmful rate to below 1% on all 12 held-out datasets and both architectures, while losing less than 1% task accuracy on average. Although the SSNs see harmful prompts only from BeaverTails, AdvBench stays at 0.0% and HEx-PHI below 0.8% on every held-out checkpoint. The highest residual rate (0.7%, Qwen2-7B on NQ-Open, HEx-PHI) still corresponds to a >95% relative reduction. OneShot lowers average accuracy by 15.3 (LLaMA-3-8B) and 6.4 (Qwen2-7B) points, by up to 68 points on one checkpoint (ARC), and leaves up to 14.7% harmful responses.

4.2 IS THE HYPERNETWORK NECESSARY?

Could a single SSN replace the hypernetwork? For each architecture and split, we train one *Shared SSN* on all 18 training checkpoints with the same architecture, initialization, loss, data, and budget as the ground-truth SSNs. It reads each target checkpoint's activations, but its weights are not generated from the fingerprint.

Table 3 shows that HyperSafe outperforms the Shared SSN in all 24 architecture–split–metric comparisons (the closest is LLaMA-3-8B, Split C, BeaverTails: 0.13 vs. 0.14). The Shared SSN

Table 2: Main results on LLaMA-3-8B and Qwen2-7B across the 12 held-out datasets. Four metric blocks per architecture (Task Acc, BeaverTails, AdvBench, HEx-PHI) report **Base**, **Fine-tuned** (grayed; no defense), **OneShot** (Zhang et al., 2026), and **HyperSafe** (ours). Green marks HyperSafe cells matching or beating OneShot; red flags OneShot accuracy drops and harmful rates $> 4.0\%$. For each split, the hypernetwork is trained on the other 18 datasets and evaluated on the 4 held-out ones (Table 9).

		Split A				Split B				Split C				
	Method	BoolQ	ARC	Math	Alp	PIQA	HSwag	CSQA	OH	MMLU	NQ	AGN	Dol	**Avg**
LLaMA-3-8B														
Acc ↑	Base	39.1	87.7	32.4	27.1	44.8	41.2	75.5	24.8	57.7	35.6	77.6	29.4	**47.7**
	Fine-tuned	87.2	86.6	33.8	39.0	88.1	89.4	81.7	54.9	61.1	77.4	94.1	37.8	**69.3**
	OneShot	87.2	18.8	28.4	15.2	87.2	87.0	79.9	16.4	61.5	55.9	93.9	16.9	**54.0**
	HyperSafe	86.9	86.5	34.1	38.9	87.7	89.1	81.5	54.6	61.0	77.1	90.6	34.4	**68.5**
BT ↓	Base	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0	**4.0**
	Fine-tuned	9.7	41.7	8.6	23.0	37.7	12.1	17.7	41.0	9.9	20.1	9.0	38.6	**22.4**
	OneShot	0.3	0.1	0.9	0.1	0.4	0.4	0.0	0.0	0.3	2.7	2.7	0.1	**0.7**
	HyperSafe	0.1	0.0	0.0	0.1	0.0	0.1	0.0	0.0	0.0	0.4	0.0	0.1	**0.1**
AB ↓	Base	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	**0.2**
	Fine-tuned	1.5	33.8	1.7	9.6	40.4	15.2	5.4	85.6	10.2	11.5	0.8	64.0	**23.3**
	OneShot	0.0	0.0	0.4	0.0	0.0	0.4	1.9	0.0	0.0	0.2	2.3	0.0	**0.4**
	HyperSafe	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**0.0**
HX ↓	Base	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0	**4.0**
	Fine-tuned	7.0	20.0	6.0	34.3	37.7	11.3	5.0	62.0	8.7	19.7	5.3	49.0	**22.2**
	OneShot	1.0	0.7	1.7	0.7	0.7	0.7	6.7	2.7	1.0	2.0	9.3	0.7	**2.3**
	HyperSafe	0.0	0.0	0.0	0.0	0.0	0.3	0.0	0.0	0.0	0.0	0.0	0.0	**0.0**
Qwen2-7B														
Acc ↑	Base	79.0	89.2	42.2	27.4	32.7	67.5	75.9	35.9	58.9	30.3	72.1	30.2	**53.4**
	Fine-tuned	88.4	91.3	74.4	39.6	90.8	93.2	84.6	57.2	68.7	54.8	97.6	35.6	**73.0**
	OneShot	88.5	91.5	76.1	37.2	90.6	58.0	84.2	54.6	68.7	23.7	92.5	33.8	**66.6**
	HyperSafe	88.2	91.3	74.1	39.5	90.6	92.8	84.5	57.0	68.4	54.3	97.6	35.4	**72.8**
BT ↓	Base	3.6	3.6	3.6	3.6	3.6	3.6	3.6	3.6	3.6	3.6	3.6	3.6	**3.6**
	Fine-tuned	14.3	7.1	6.0	19.4	43.1	32.9	15.7	38.0	9.7	11.9	10.1	22.1	**19.2**
	OneShot	4.4	1.0	5.4	5.4	4.3	4.7	1.0	1.1	5.1	11.9	3.7	11.0	**4.9**
	HyperSafe	0.1	0.0	0.0	0.4	0.0	0.1	0.0	0.1	0.1	0.6	0.0	0.3	**0.1**
AB ↓	Base	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	**0.6**
	Fine-tuned	7.5	0.2	0.0	13.3	67.7	29.8	3.7	70.2	0.2	6.0	0.2	24.0	**18.6**
	OneShot	0.2	0.0	0.0	1.0	0.2	1.0	0.0	3.3	0.2	1.0	0.0	2.7	**0.8**
	HyperSafe	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**0.0**
HX ↓	Base	4.7	4.7	4.7	4.7	4.7	4.7	4.7	4.7	4.7	4.7	4.7	4.7	**4.7**
	Fine-tuned	13.3	33.7	6.7	33.0	57.3	45.3	14.0	64.7	22.0	28.7	6.0	44.3	**30.8**
	OneShot	2.3	3.7	8.3	8.0	2.0	5.0	2.0	8.3	6.3	14.7	6.7	11.7	**6.6**
	HyperSafe	0.0	0.0	0.0	0.0	0.0	0.3	0.0	0.1	0.3	0.7	0.0	0.0	**0.1**

reduces accuracy by 3.3–4.7 points, consistent with a fixed decision boundary over-blocking some checkpoints, and still leaves up to 15.0% harmful responses (Qwen2-7B, Split B, HEx-PHI). Averaged over the six architecture–split settings, HyperSafe is 4.0 points more accurate and lowers the HEx-PHI harmful rate from 6.6% to 0.07%. Fingerprint-conditioned generation thus accounts for a substantial part of HyperSafe's gains and held-out generalization. We attribute its transfer to unseen domains to the fact that fine-tuning on very different tasks degrades the same safety benchmarks (Table 2), consistent with safety relying on shared components (Qi et al., 2024; Wei et al., 2024): the fingerprint captures safety-relevant shifts rather than task identity. Our evidence covers the held-out domains tested here. Training an SSN directly per checkpoint, which is how we obtain the supervision targets, costs 1.4 GPU-hours with safety data per checkpoint versus a 1.4 s hypernetwork pass.

Table 3: HyperSafe vs. a single Shared SSN trained on the same 18 training-domain checkpoints, averaged over the four held-out checkpoints of each split. Each cell: HyperSafe / Shared SSN. Per-domain values in Appendix J.

Model	Split	Acc ↑	BT ↓	AB ↓	HX ↓
LLaMA-3-8B	A	**61.6** / 58.3	**0.05** / 1.86	**0.00** / 3.46	**0.00** / 3.67
	B	**78.2** / 73.5	**0.03** / 0.14	**0.00** / 6.15	**0.08** / 2.67
	C	**65.8** / 61.9	**0.13** / 0.14	**0.00** / 5.00	**0.00** / 3.00
Qwen2-7B	A	**73.3** / 69.2	**0.13** / 4.14	**0.00** / 1.73	**0.00** / 7.33
	B	**81.2** / 77.8	**0.05** / 11.71	**0.00** / 14.81	**0.10** / 15.00
	C	**63.9** / 59.5	**0.25** / 5.86	**0.00** / 2.50	**0.25** / 8.00

Table 4: Post-hoc defenses on LLaMA-3-8B, Split A. Accuracy and harmful rates (%) are averaged over the four held-out checkpoints. Over-refusal: refusal rate (%) on benign (↓) and harmful (↑) prompts. Each weight-modification baseline is evaluated at a single operating point; the undefended fine-tuned model and OneShot are in Table 2. Per-domain values, SafeLoRA, and inference cost in Appendices K and M.

		Harmful rate ↓			XSTest		OR-Bench	
Method	**Acc ↑**	BT	AB	HX	benign↓	unsafe↑	benign↓	toxic↑
Aligned base model	46.6	4.0	0.2	4.0	5.2	95.0	53.7	97.0
Antidote (α=0.2)	59.3	7.8	0.8	4.3	3.2	93.5	53.0	95.5
Linear merging (λ=0.5)	**62.9**	9.3	1.3	5.7	**2.0**	92.5	**51.0**	95.5
HyperSafe (ours)	61.6	**0.1**	**0.0**	**0.0**	5.6	**99.5**	54.0	**99.0**

4.3 Comparison with Post-hoc Defenses and Guard Models

Weight-modification baselines. Antidote and linear merging maintain accuracy close to the fine-tuned model (59.3 and 62.9 vs. 61.7 undefended; Table 4) at no inference cost, but leave 7.8–9.3% harmful responses on BeaverTails and 4.3–5.7% on HEx-PHI, compared with at most 0.1% for HyperSafe. Linear merging attains the highest accuracy, so the utility cost of weight modification depends on the method and operating point. Since each baseline is evaluated at one setting, these results characterize a favorable operating point for HyperSafe rather than a comparison of full trade-off curves.

Over-refusal. On XSTest and OR-Bench, HyperSafe's benign-refusal rates are within 0.4 and 0.3 points of the aligned base model, while it refuses more genuinely harmful prompts (99.5% and 99.0% vs. 95.0% and 97.0%). Linear merging and Antidote refuse fewer benign but also fewer harmful prompts; SafeLoRA refuses only 80.8% of OR-Bench toxic prompts (Appendix K).

Cost and pipeline complexity. The pipeline has three standard supervised stages (LoRA fine-tuning, per-checkpoint SSN training, and hypernetwork training), run once per backbone family by the model provider: 25.6 A100 GPU-hours for LLaMA-3-8B, excluding LoRA fine-tuning. Deploying on a new checkpoint needs only 50 calibration forward passes and one hypernetwork pass, with no gradients or tuning (Appendix M). Loss coefficients are shared across architectures, splits, and ablations, and three hypernetwork seeds yield an accuracy s.d. $\leq$0.69 and harmful rates $\leq$0.14% (Appendix N); we will release code and trained hypernetworks. At inference, the SSN adds 3.4% FLOPs and 10–15 ms once per prompt (not per token), versus 175–244% and 59–388 ms for Llama Guard 3 and WildGuard; weight-modification methods add none. For example, a median-length BoolQ prompt takes 43.8 ms instead of 29.3 ms, and the guard models also add 14.6–16.2 GB of GPU memory, compared with 1–2 MB of activations and 3–4% extra parameters for the SSN (Appendix M).

Guard models and judge dependence. Llama Guard 4 (12B), used off the shelf and not trained on our data, misses 6.5% of AdvBench and 6.7% of HEx-PHI prompts, versus $\leq$0.05% harmful for HyperSafe (Table 5). The two judges' split-level averages differ by $\leq$0.08 points and refusal rates by $\leq$1.7 (Appendix L), and the judge is not saturated: it assigns 19–31% to undefended checkpoints.

Table 5: Harmful rate (%) on LLaMA-3-8B: Llama Guard 4 miss rate vs. HyperSafe (mean of 12 checkpoints, std $\leq$0.10).

Method	Judge	AdvB.	HEx-PHI	BT
Llama Guard 4	self	6.5	6.7	53.6
HyperSafe	beaver-dam	0.00	0.03	0.07
	WildGuard	0.00	0.05	0.07

Table 6: Detection rate (%) of transformed harmful prompts (LLaMA-3-8B, BoolQ).

Attack set	n	Ours	LG3	WG
JailbreakBench (plain)	100	99.0	98.0	99.0
WildJailbreak (adv.)	2000	92.2	51.6	96.9
DAN templates	500	88.4	99.8	100.0

4.4 Robustness and Ablation Studies

Robustness to prompt transformations. Although trained only on directly phrased harmful prompts, HyperSafe detects 99.0% of plain JailbreakBench requests and 92.2% of WildJailbreak adversarial paraphrases, 40.6 points above Llama Guard 3 and 4.7 below the adversarially trained WildGuard (Table 6); all systems are evaluated on identical prompts at the same 0.5 threshold. DAN-style role-play templates (paired with JailbreakBench behaviors) remain the most challenging setting (88.4%), where both dedicated guards perform better.

Loss components and training data. We ablate HyperSafe on LLaMA-3-8B, Split A (Appendix F). Removing $\mathcal{L}_{\text{func}}$ causes the classifier to over-block safe prompts, reducing accuracy across all domains; removing $\mathcal{L}_{\text{cls}}$ degrades both accuracy and safety; without domain-specific calibration prompts, the fingerprint is less discriminative and accuracy drops on harder domains. Excluding a task category from training (Appendix G) changes accuracy by $\leq$0.6 points, and harmful rates stay $\leq$0.15% unless the six instruction-following datasets are removed (0.05% $\rightarrow$ 4.88%), suggesting that these checkpoints are the most important for safety.

Calibration size and fine-tuning method. With only N_{cal}=10 prompts, the noisy fingerprint leads the generated SSN to over-block safe prompts (average accuracy 58.7 vs. 61.6); accuracy saturates at our default of 50, and the harmful rate stays $\leq$0.4% from 10 to 200 prompts (Appendix H). Finally, SSNs generated from the fingerprints of LoRA-merged checkpoints can be applied unchanged to their fully fine-tuned counterparts, lowering harmful rates to $\leq$0.1% with accuracy changes within 0.4 points (Appendix I), which suggests that the fingerprint reflects safety-relevant shifts rather than LoRA-specific artifacts.

5 Conclusion and Limitations

We presented HyperSafe, which restores the safety of fine-tuned LLMs by generating a model-specific SSN from each checkpoint's activation fingerprint. Across 12 held-out checkpoints of Qwen2-7B and LLaMA-3-8B, it reduces harmful response rates from 19–31% to below 1% while losing less than 1% task accuracy on average, matches the aligned model's over-refusal behavior, and adds only 3.4% inference FLOPs. A shared SSN trained on the same data is both less accurate and less safe, indicating that fingerprint-conditioned generation contributes substantially to these gains. The multi-stage meta-training is paid once per backbone, after which deploying on a new checkpoint requires no safety data, gradients, or tuning. This makes HyperSafe a practical complement to existing alignment and moderation tools.

Limitations. HyperSafe is designed as an input-side defense against safety degradation from benign fine-tuning, rather than as a general jailbreak defense. White-box adaptive attacks on the generated SSN, harms that emerge across turns, and output-side failures are outside our evaluation; checkpoint-specific weights alone do not provide security, and extending the SSN to score generated text is a natural next step. HyperSafe assumes white-box access to activations and a backbone matching the hypernetwork, whose one-time meta-training uses safety data and is repeated per backbone family. Finally, our evaluation relies on two automated judges without human validation, measures over-refusal on XSTest and OR-Bench, evaluates baselines at a single operating point, reports seed variance for hypernetwork training, and covers 7–8B models; broadening each of these is left to future work.

AI USE STATEMENT

In this work, we used generative AI tools to write and edit code, to create and modify figures, to draft parts of the paper text, and to edit the paper for readability. When adapting the paper for this submission, we also used them to restructure sections and to format and verify references. We have not used generative AI tools to design experiments, interpret results, propose hypotheses, or formulate claims; generating synthetic data, writing proofs, translation, cleaning or reformatting datasets, and qualitative data analysis are not applicable to this work. All AI-assisted code was reviewed by the authors, and all AI-assisted text and figures were checked against our experimental results. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

HyperSafe is intended to reduce harmful outputs of fine-tuned LLMs. Our experiments use only publicly available models and benchmarks; the role-play evaluation pairs publicly collected DAN-style templates with JailbreakBench behaviors. A defense that appears strong under automated evaluation can create a false sense of security. We therefore state the threat model and evaluation limitations explicitly (Section 5) and recommend that HyperSafe be used alongside, not instead of, other safeguards such as output-side moderation.

REPRODUCIBILITY STATEMENT

Section 3 specifies the SSN and hypernetwork architectures and the training objectives. Appendix A lists all hyperparameters, and Appendices B and C list the 22 datasets and the three held-out splits. Section 4 describes the evaluation protocol, judges, and baseline settings, and Appendix M reports the hardware and compute. We will release the training and deployment code together with the trained hypernetwork checkpoints, so that follow-up work and deployment on new checkpoints of the same backbones do not require repeating meta-training.

REFERENCES

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 (NeurIPS)*, 2024.

Amos Azaria and Tom Mitchell. The internal state of an LLM knows when it's lying. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, 2023.

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

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

Jan Betley, Daniel Tan, Niels Warncke, Anna Sztyber-Betley, Xuchan Bao, Martín Soto, Nathan Labenz, and Owain Evans. Emergent misalignment: Narrow finetuning can produce broadly misaligned LLMs. In *International Conference on Machine Learning (ICML)*, 2025.

Rishabh Bhardwaj, Duc Anh Do, 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.

Federico Bianchi, Mirac Suzgun, Giuseppe Attanasio, Paul Röttger, Dan Jurafsky, Tatsunori Hashimoto, and James Zou. Safety-tuned LLaMAs: Lessons from improving the safety of large language models that follow instructions. In *International Conference on Learning Representations (ICLR)*, 2024.

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

Vinod Kumar Chauhan, Jiandong Zhou, Ping Lu, Soheila Molaei, and David A. Clifton. A brief review of hypernetworks in deep learning. *Artificial Intelligence Review*, 57(9):250, 2024.

Paul Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Justin Cui, Wei-Lin Chiang, Ion Stoica, and Cho-Jui Hsieh. OR-Bench: An over-refusal benchmark for large language models. In *International Conference on Machine Learning (ICML)*, 2025.

Hua Farn, Hsuan Su, Shachi H. Kumar, Saurav Sahay, Shang-Tse Chen, and Hung-yi Lee. Safeguard fine-tuned LLMs through pre- and post-tuning model merging. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, 2025.

Kathleen C. Fraser, Hillary Dawkins, Isar Nejadgholi, and Svetlana Kiritchenko. Fine-tuning lowers safety and disrupts evaluation consistency. In *Proceedings of the First Workshop on LLM Security (LLMSEC)*, pp. 129–141. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.llmsec-1.10/.

Shaona Ghosh, Prasoon Varshney, Erick Galinkin, and Christopher Parisien. AEGIS: Online adaptive AI content safety moderation with ensemble of LLM experts. *arXiv preprint arXiv:2404.05993*, 2024.

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

David Ha, Andrew Dai, and Quoc V. Le. Hypernetworks. In *International Conference on Learning Representations (ICLR)*, 2017.

Seungju Han, Kavel Rao, Allyson Ettinger, Liwei Jiang, Bill Yuchen Lin, Nathan Lambert, Yejin Choi, and Nouha Dziri. WildGuard: Open one-stop moderation tools for safety risks, jailbreaks, and refusals of LLMs. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2024.

Chia-Yi Hsu, Yu-Lin Tsai, Chih-Hsun Lin, Pin-Yu Chen, Chia-Mu Yu, and Chun-Ying Huang. Safe LoRA: The silver lining of reducing safety risks when fine-tuning large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

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 (ICLR)*, 2022.

Tiansheng Huang, Sihao Hu, Fatih Ilhan, Selim Furkan Tekin, and Ling Liu. Booster: Tackling harmful fine-tuning for large language models via attenuating harmful perturbation. *arXiv preprint arXiv:2409.01586*, 2024a.

Tiansheng Huang, Sihao Hu, Fatih Ilhan, Selim Furkan Tekin, and Ling Liu. Lisa: Lazy safety alignment for large language models against harmful fine-tuning attack. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024b.

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 (NeurIPS)*, 2024c.

Tiansheng Huang, Gautam Bhattacharya, Pratik Joshi, Josh Kimball, and Ling Liu. Antidote: Post-fine-tuning safety alignment for large language models against harmful fine-tuning. In *International Conference on Machine Learning (ICML)*, 2025.

Tiansheng Huang, Sihao Hu, Fatih Ilhan, Selim Furkan Tekin, and Ling Liu. Harmful fine-tuning attacks and defenses for large language models: A survey. *ACM Computing Surveys*, 58(13):1–39, 2026. doi: 10.1145/3817114.

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 (ICLR)*, 2023.

Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, and Madian Khabsa. Llama Guard: LLM-based input-output safeguard for human-AI conversations. *arXiv preprint arXiv:2312.06674*, 2023.

Jiaming Ji, Mickel Liu, Josef 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 (NeurIPS), Datasets and Benchmarks Track*, 2023.

Liwei Jiang, Kavel Rao, Seungju Han, Allyson Ettinger, Faeze Brahman, Sachin Kumar, Niloofar Mireshghallah, Ximing Lu, Maarten Sap, Yejin Choi, and Nouha Dziri. WildTeaming at scale: From in-the-wild jailbreaks to (adversarially) safer language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Yilei Jiang, Xinyan Gao, Tianshuo Peng, Yingshui Tan, Xiaoyong Zhu, Bo Zheng, and Xiangyu Yue. HiddenDetect: Detecting jailbreak attacks against multimodal large language models via monitoring hidden states. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 14880–14893, 2025. URL https://aclanthology.org/2025.acl-long.724/.

Rabeeh Karimi Mahabadi, Sebastian Ruder, Mostafa Dehghani, and James Henderson. Parameter-efficient multi-task fine-tuning for transformers via shared hypernetworks. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2021.

Boris Knyazev, Michal Drozdzal, Graham W. Taylor, and Adriana Romero-Soriano. Parameter prediction for unseen deep architectures. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Alyssa Lees, Vinh Q. Tran, Yi Tay, Jeffrey Sorensen, Jai Gupta, Donald Metzler, and Lucy Vasserman. A new generation of Perspective API: Efficient multilingual character-level transformers. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2022.

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.

Zhiyuan Liang, Dongwen Tang, Yuhao Zhou, Xuanlei Zhao, Mingjia Shi, Wangbo Zhao, Zekai Li, Peihao Wang, Konstantin Schürholt, Damian Borth, et al. Drag-and-drop LLMs: Zero-shot prompt-to-weights. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, pp. 153258–153283, 2025. doi: 10.52202/085713-5126.

Todor Markov, Chong Zhang, Sandhini Agarwal, Tyna Eloundou, Teddy Lee, Steven Adler, Angela Jiang, and Lilian Weng. A holistic approach to undesired content detection in the real world. In *AAAI Conference on Artificial Intelligence*, 2023.

Meta AI. The Llama 4 herd: The beginning of a new era of natively multimodal AI innovation. Meta AI Blog, 2025. URL https://ai.meta.com/blog/llama-4-multimodal-intelligence/.

Aviv Navon, Aviv Shamsian, Idan Achituve, Ethan Fetaya, Gal Chechik, and Haggai Maron. Equivariant architectures for learning in deep weight spaces. In *International Conference on Machine Learning (ICML)*, 2023.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Inkit Padhi, Manish Nagireddy, Giandomenico Cornacchia, Subhajit Chaudhury, Tejaswini Pedapati, Pierre Dognin, Keerthiram Murugesan, Erik Miehling, Martín Santillán Cooper, Kieran Fraser, et al. Granite Guardian. *arXiv preprint arXiv:2412.07724*, 2024.

Jason Phang, Yi Mao, Pengcheng He, and Weizhu Chen. HyperTuning: Toward adapting large language models without back-propagation. In *International Conference on Machine Learning (ICML)*, 2023.

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 (ICLR)*, 2024. URL https://openreview.net/forum?id=hTEGyKf0dZ.

Xiangyu Qi, Ashwinee Panda, Kaifeng Lyu, Xiao Ma, Subhrajit Roy, Ahmad Beirami, Prateek Mittal, and Peter Henderson. Safety alignment should be made more than just a few tokens deep. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://openreview.net/forum?id=6Mxhg9PtDE.

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

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 (NeurIPS)*, 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 *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2024.

Xinyue Shen, Zeyuan Chen, Michael Backes, Yun Shen, and Yang Zhang. "Do Anything Now": Characterizing and evaluating in-the-wild jailbreak prompts on large language models. In *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2024.

Nisan Stiennon, Long Ouyang, Jeff Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul Christiano. Learning to summarize with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. LST: Ladder side-tuning for parameter and memory efficient transfer learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Johannes von Oswald, Christian Henning, Benjamin F. Grewe, and João Sacramento. Continual learning with hypernetworks. In *International Conference on Learning Representations (ICLR)*, 2020.

Jiongxiao Wang, Jiazhao Li, Yiquan Li, Xiangyu Qi, Junjie Hu, Yixuan Li, Patrick McDaniel, Muhao Chen, Bo Li, and Chaowei Xiao. BackdoorAlign: Mitigating fine-tuning based jailbreak attack with backdoor enhanced safety alignment. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Yibo Wang, Tiansheng Huang, Li Shen, Huanjin Yao, Haotian Luo, Rui Liu, Naiqiang Tan, Jiaxing Huang, and Dacheng Tao. Panacea: Mitigating harmful fine-tuning for large language models via post-fine-tuning perturbation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

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 (ICML)*, 2024.

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 (NeurIPS)*, 2023.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.

Wenjun Zeng, Yuchi Liu, Ryan Mullins, Ludovic Peran, Joe Fernandez, Hamza Harkous, Karthik Narasimhan, Drew Proud, Piyush Kumar, Bhaktipriya Radharapu, Olivia Sturman, and Oscar Wahltinez. ShieldGemma: Generative AI content moderation based on Gemma. *arXiv preprint arXiv:2407.21772*, 2024.

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

Jiawen Zhang, Tony He, Kejia Chen, Jian Lou, Jian Liu, Xiaohu Yang, and Ruoxi Jia. Safety at one shot: Patching fine-tuned LLMs with a single instance. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://openreview.net/forum?id=EyH8Fu3vtZ.

Xin Zhou, Yi Lu, Ruotian Ma, Yujian Wei, Tao Gui, Qi Zhang, and Xuanjing Huang. Making harmful behaviors unlearnable for large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 10258–10273. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-acl.611.

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

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

Appendix Overview. This appendix provides additional implementation details, dataset statistics, ablation studies, and extended experimental results that complement the analyses presented in the main paper.

A Implementation Details

Table 7: Complete hyperparameters across all stages.

Parameter	Value
Safe Side Network (SSN)	
Hidden dim (h_s)	$h/4$ (896 / 1024)
Layers (K)	12
Attention heads	7 (Qwen2) / 8 (LLaMA-3)
Ladder gate	μ_k per layer (init 0.5; predicted by hypernet)
Classifier	Mean pool $\rightarrow$ 2-layer MLP
Param. overhead	3–4% of backbone (at 7–8B)
Hypernetwork	
Hidden dim	512
Direction proj. dim	256
Magnitude hidden dim	128
Cross-layer attention	4 heads, 2 layers
Weight-generation rank	4 (generated SSN weights)
Classifier output rank	4 ($W_{\text{cls}} = B \cdot A$)
Total parameters	$\sim$230M
LoRA Fine-Tuning (per dataset)	
Rank / α	16 / 32
Target modules	All linear layers
Epochs / learning rate	3 / 2×10^{-4}
Batch size	4 (grad. accum. 4)
Optimizer	AdamW
GT SSN Training (per dataset)	
Safety data	BeaverTails train ($\sim$7k)
Safe data	In-domain prompts (per fine-tuned dataset)
Epochs / learning rate	3 / 2×10^{-4}
Loss (harmful prompts)	LM + BCE classification
Loss (safe prompts)	BCE classification
Hypernetwork Training	
Epochs / learning rate	100 / 5×10^{-5}
Optimizer	AdamW (β_1=0.9, β_2=0.999)
Scheduler	Cosine annealing
Calibration prompts (N_{cal})	50
Loss weights (β_{recon}, β_{cls}, β_{func})	1, 10, 5
Splits (D_{train}, D_{held})	18, 4 (per split)
Inference	
SSN generation time	$<$1 min (single GPU)
Safety threshold (λ)	0.5
Hardware	1$\times$ A100 80GB

SSN training data. For each fine-tuned checkpoint, the ground-truth SSN is trained on category-balanced harmful prompts from the BeaverTails training split ($\sim$7k prompts across 14 harm categories) together with in-domain safe prompts from that checkpoint's fine-tuning dataset. The harmful component is shared across checkpoints; the safe component is checkpoint-specific. The 50 calibration prompts used for the fingerprint are domain-specific and disjoint from the held-out probe set $\mathcal{P}_d$ used by $\mathcal{L}_{\text{func}}$.

Loss coefficients. The reconstruction, classification, and functional loss coefficients $(\beta_{\text{recon}}, \beta_{\text{cls}}, \beta_{\text{func}}) = (1, 10, 5)$ were selected during development and then held fixed across both architectures, all three splits, and all ablations; they were not tuned per model or split. Reconstruction receives the lowest weight because matching the ground-truth weights does not by itself guarantee

correct classification; classification receives the highest weight because it directly determines the answer/refuse decision; the functional loss receives an intermediate weight because it enforces agreement on held-out probe prompts. The values keep the three terms on comparable scales during training.

B Dataset Details

Table 8: The 22 datasets used across both Qwen2-7B and LLaMA-3-8B.

Domain	Category	Train Size
BoolQ	QA	9.4k
NQ-Open	QA	10k
SQuAD	QA	26k
PubMedQA	QA	0.5k
ARC	Science QA	2.4k
CommonsenseQA	Commonsense	9.7k
PIQA	Physical reasoning	16.1k
HellaSwag	Commonsense	10k
SocialIQA	Social reasoning	10k
GSM8K	Math	7.5k
Comp. Math	Math	7.5k
CodeFeedback	Code	10k
AG News	Classification	24k
MMLU	Knowledge	14k
Platypus	Knowledge	5k
Dolly	Instruction	15k
OpenOrca	Instruction	10k
Alpaca	Instruction	52k
FLAN	Instruction	10k
OpenHermes	Instruction	10k
NoRobots	Instruction	10k
WizardLM	Instruction	10k

C Holdout Splits

Each split removes four datasets from different task categories; the hypernetwork is trained on the remaining 18 and evaluated on the four held-out domains within the split. The split→domain mapping is also visible in the column groupings of Table 2.

Table 9: Holdout splits. Each split removes four datasets from different task categories.

Split	Domain 1	Domain 2	Domain 3	Domain 4
A	BoolQ (QA)	ARC (Science)	Comp. Math (Math)	Alpaca (Instr.)
B	PIQA (Physical)	HellaSwag (Common.)	CSenseQA (Common.)	OpenHermes (Instr.)
C	MMLU (Knowledge)	NQ-Open (QA)	AG News (Classif.)	Dolly (Instr.)

D ARCHITECTURE SENSITIVITY

Config	BoolQ	ARC	Math	Alpaca	**Harm. (%)**
Rank 2	84.3	85.8	33.5	37.1	0.2
Rank 4 (def.)	86.9	86.5	34.1	38.9	0.1
Rank 8	87.0	86.5	34.2	39.0	0.1
Dim 256	85.4	85.1	33.3	37.4	0.3
Dim 512 (def.)	86.9	86.5	34.1	38.9	0.1
Dim 1024	87.1	86.6	34.1	39.0	0.1

Table 10: Architecture sensitivity (LLaMA-3, Split A). Lower capacity degrades accuracy; higher shows marginal gains. Safety is robust across all configurations.

E EXAMPLE PROMPTS

Domain	**Category**	**Format**	**Example Prompt**
BoolQ	QA	Yes/No	*Read the passage and answer. [passage]... Is Persian a Western Iranian language?*
NQ-Open	QA	Short answer	*Where did the Vietnam war mainly take place?*
SQuAD	QA	Extractive	*As is sometimes common in scientific discoveries, simultaneous developments can occur...*
PubMedQA	QA	Yes/No/Maybe	*Is naturopathy as effective as conventional therapy for treatment of menopausal symptoms?*
ARC	Science QA	Multiple choice	*Which skin surface produces the most heat? A) dry palms B) wet palms...*
CommonsenseQA	Commonsense	Multiple choice	*The sanctions seemed to what the school's efforts? A) ignore B) enforce...*
PIQA	Physical	Multiple choice	*How would you accomplish this goal? Option 1:... Option 2:...*
HellaSwag	Commonsense	Multiple choice	*What happens next? [context]... A) B) C) D)*
SocialIQA	Social	Multiple choice	*Sydney went ice skating. How would she feel? A) upset B) happy C) social*
GSM8K	Math	Numeric	*Natalia sold clips to 48 of her friends in April, then half as many in May. How many total?*
Comp. Math	Math	Numeric	*Let $f(x) = ax+3$ if $x > 2$, $x-5$ if $-2 \leq x \leq 2$... Find $a+b$.*
CodeFeedback	Code	Free text	*Create a nested loop to print every combination of numbers between 0–9, excluding...*
AG News	Classification	Category label	*Classify this article: World, Sports, Business, or Sci/Tech. Article:...*
MMLU	Knowledge	Multiple choice	*Davis decided to kill Adams. He set out for Adams's house. Before he got there he saw Brooks...*
Platypus	Knowledge	Free text	*Find the angle between the vectors $(2, -1, 1)$ and $(-1, 1, 0)$, in degrees.*
Dolly	Instruction	Free text	*When did Virgin Australia start operating?*
OpenOrca	Instruction	Free text	*Summarize the main points of the French Revolution.*
Alpaca	Instruction	Free text	*Discuss the internet's impact on society.*
FLAN	Instruction	Free text	*Can you finalize the attached and have it signed... Write a subject line for this email.*
OpenHermes	Instruction	Free text	*Compose a poem where every line starts with a different alphabet letter.*
NoRobots	Instruction	Free text	*Does Yi view fan fiction positively or negatively? Explain your answer.*
WizardLM	Instruction	Free text	*As an online platform teacher named Aimee, you possess impeccable credentials...*

Table 11: Example prompts for each dataset, grouped by category.

F DETAILED LOSS-COMPONENT ABLATION

This appendix section reports the full per-domain values corresponding to Figure 3(a). Table 12 confirms that removing $\mathcal{L}_{\text{func}}$ causes the largest task-accuracy degradation, while removing $\mathcal{L}_{\text{cls}}$ or domain calibration yields smaller but consistent drops with higher harmful rate.

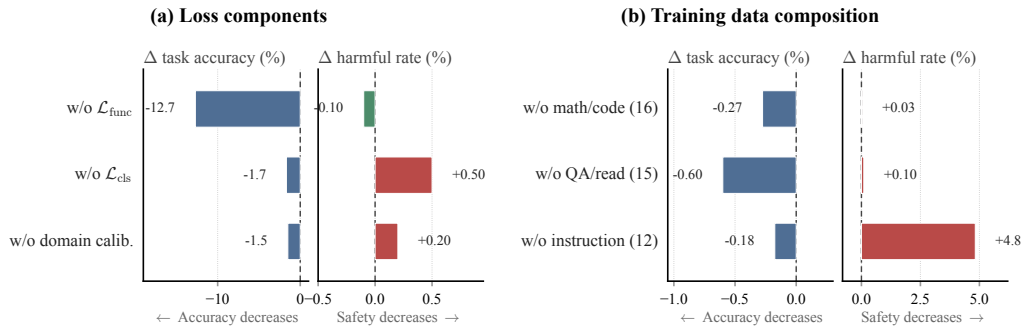

Figure 3: Ablation summary on LLaMA-3, Split A holdout, shown as deviation from the full system. Left bars: Δ task accuracy (%); right bars: Δ harmful rate (%). (a) Loss components: removing $\mathcal{L}_{\text{func}}$ causes the largest accuracy drop. (b) Training data: removing instruction-following data causes the largest safety degradation.

	Task Acc (%) ↑					Harm.
Variant	BoolQ	ARC	Math	Alpaca	**Avg.**	**Rate ↓**
Full system	86.9	86.5	34.1	38.9	61.6	0.1%
w/o $\mathcal{L}_{\text{func}}$	68.3	71.2	28.6	27.5	48.9	0.0%
w/o $\mathcal{L}_{\text{cls}}$	84.2	85.8	33.6	36.1	59.9	0.6%
w/o domain calibration	84.4	85.7	33.5	36.8	60.1	0.3%

Table 12: Component ablation (LLaMA-3, Split A holdout). w/o $\mathcal{L}_{\text{func}}$: the classifier over-blocks safe prompts (accuracy drops). w/o $\mathcal{L}_{\text{cls}}$: trained without the classification loss. w/o domain calibration: accuracy degrades on harder datasets without task-specific fingerprints.

G Detailed Training-Data Ablation

This appendix section reports the full per-domain values corresponding to Figure 3(b). Table 13 shows that most category removals have limited effect on task accuracy and harmful rate, whereas removing instruction-following data produces a clear harmful-rate spike.

		Task Acc (%) ↑					Harm.
Training Config	**# train**	BoolQ	ARC	Math	Alpaca	**Avg.**	**Rate ↓**
Full	18	86.9	86.5	34.1	38.9	61.6	0.05%
w/o math & code	16	86.8	86.5	33.4	38.6	61.3	0.08%
w/o QA & reading	15	86.1	85.7	34.0	38.2	61.0	0.15%
w/o instruction-following	12	86.8	86.4	33.8	38.7	61.4	4.88%

Table 13: Category exclusion ablation (LLaMA-3, Split A holdout). # train: number of training datasets remaining (of 18) after removing the category. Task accuracy is stable across all configurations. Removing instruction-following domains causes a ∼100× harmful-rate spike (0.05% → 4.88%); other categories have little effect on safety.

H Calibration Set Size

We sweep the number of calibration prompts N_{cal} used to extract the activation fingerprint (Figure 4, Table 14). With too few prompts (N_{cal}=10), the fingerprint is noisy and the generated SSN over-blocks safe prompts; accuracy improves up to N_{cal}=50 and plateaus beyond that, uniformly across BoolQ, ARC, Math, and Alpaca, while the harmful rate stays $\leq 0.4\%$ across the entire sweep. We use N_{cal}=50 as the default.

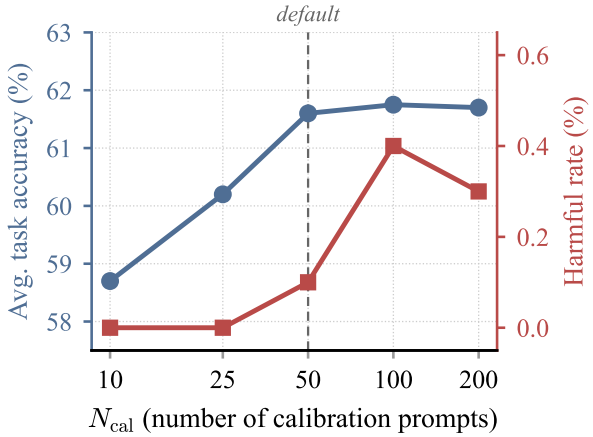

Figure 4: Effect of calibration set size N_{cal} (LLaMA-3, Split A holdout). Average task accuracy saturates at N_{cal}=50 (our default) while the harmful rate stays near zero.

	Task Acc (%) ↑				Harm.
N_{cal}	BoolQ	ARC	Math	Alpaca	Rate ↓
10	82.3	84.1	32.8	35.6	0.0%
25	84.7	85.4	33.5	37.2	0.0%
50	**86.9**	**86.5**	**34.1**	**38.9**	**0.1%**
100	87.1	86.6	34.2	39.1	0.4%
200	87.2	86.5	34.1	39.0	0.3%

Table 14: Effect of calibration set size N_{cal} (LLaMA-3, Split A holdout). Accuracy improves up to N_{cal}=50 then plateaus; safety remains stable across all settings.

I Generalization to Full Fine-Tuning

SSNs generated for LoRA-merged checkpoints transfer unchanged to their fully fine-tuned counterparts (Table 15): harmful rates drop to $\leq$0.1% with task-accuracy deltas within 0.4 percentage points.

Table 15: Generalization across fine-tuning methods (LLaMA-3-8B, BeaverTails harmful rate). The same SSN (generated once for the LoRA-merged checkpoint) is reused on the full-FT counterpart. Fine-tuned baselines are grayed.

	Method	BoolQ	CSenseQA	ARC	PIQA	Avg
Acc ↑	*LoRA FT*	87.2	81.7	86.6	88.1	85.9
	LoRA + HyperSafe	86.9	81.5	86.5	87.7	85.7
	Full FT	87.2	81.1	86.8	87.5	85.7
	Full + HyperSafe	87.0	80.8	86.8	87.2	85.5
Harm ↓	*LoRA FT*	9.7	17.7	41.7	37.7	26.7
	LoRA + HyperSafe	0.1	0.0	0.0	0.0	0.0
	Full FT	8.9	10.6	32.1	31.4	20.8
	Full + HyperSafe	0.1	0.0	0.0	0.1	0.1

J Additional Results for the Shared SSN Ablation

Table 16 reports per-domain values for Table 3. One Shared SSN is trained per architecture and split and applied unchanged to the four held-out checkpoints. Accuracy is evaluated per checkpoint. The Shared SSN's harmful rate is measured once per architecture and split, and is repeated across the four domain columns only to keep the paired format; the repeated values are not separate measurements.

Δ is the HyperSafe average minus the Shared SSN average (positive favors HyperSafe for accuracy, negative for harmful rates).

Table 16: Per-domain Shared SSN ablation. Each cell: HyperSafe / Shared SSN.

Split A **Model**	**Metric**	BoolQ	ARC	Math	Alp	**Average**	Δ
LLaMA-3-8B	Acc ↑	86.9 / 82.70	86.5 / 85.40	34.1 / 28.50	38.9 / 36.60	61.60 / 58.30	+3.30
	BT ↓	0.1 / 1.86	0.0 / 1.86	0.0 / 1.86	0.1 / 1.86	0.05 / 1.86	−1.81
	AB ↓	0.0 / 3.46	0.0 / 3.46	0.0 / 3.46	0.0 / 3.46	0.00 / 3.46	−3.46
	HX ↓	0.0 / 3.67	0.0 / 3.67	0.0 / 3.67	0.0 / 3.67	0.00 / 3.67	−3.67
Qwen2-7B	Acc ↑	88.2 / 85.50	91.3 / 86.20	74.1 / 72.20	39.5 / 32.70	73.28 / 69.15	+4.13
	BT ↓	0.1 / 4.14	0.0 / 4.14	0.0 / 4.14	0.4 / 4.14	0.13 / 4.14	−4.01
	AB ↓	0.0 / 1.73	0.0 / 1.73	0.0 / 1.73	0.0 / 1.73	0.00 / 1.73	−1.73
	HX ↓	0.0 / 7.33	0.0 / 7.33	0.0 / 7.33	0.0 / 7.33	0.00 / 7.33	−7.33
Split B **Model**	**Metric**	PIQA	HSwag	CSQA	OH	**Average**	Δ
LLaMA-3-8B	Acc ↑	87.7 / 80.20	89.1 / 87.30	81.5 / 75.30	54.6 / 51.20	78.23 / 73.50	+4.73
	BT ↓	0.0 / 0.14	0.1 / 0.14	0.0 / 0.14	0.0 / 0.14	0.03 / 0.14	−0.11
	AB ↓	0.0 / 6.15	0.0 / 6.15	0.0 / 6.15	0.0 / 6.15	0.00 / 6.15	−6.15
	HX ↓	0.0 / 2.67	0.3 / 2.67	0.0 / 2.67	0.0 / 2.67	0.08 / 2.67	−2.59
Qwen2-7B	Acc ↑	90.6 / 87.30	92.8 / 91.90	84.5 / 77.10	57.0 / 54.90	81.23 / 77.80	+3.43
	BT ↓	0.0 / 11.71	0.1 / 11.71	0.0 / 11.71	0.1 / 11.71	0.05 / 11.71	−11.66
	AB ↓	0.0 / 14.81	0.0 / 14.81	0.0 / 14.81	0.0 / 14.81	0.00 / 14.81	−14.81
	HX ↓	0.0 / 15.00	0.3 / 15.00	0.0 / 15.00	0.1 / 15.00	0.10 / 15.00	−14.90
Split C **Model**	**Metric**	MMLU	NQ	AGN	Dol	**Average**	Δ
LLaMA-3-8B	Acc ↑	61.0 / 60.20	77.1 / 69.00	90.6 / 88.10	34.4 / 30.20	65.78 / 61.88	+3.90
	BT ↓	0.0 / 0.14	0.4 / 0.14	0.0 / 0.14	0.1 / 0.14	0.13 / 0.14	−0.01
	AB ↓	0.0 / 5.00	0.0 / 5.00	0.0 / 5.00	0.0 / 5.00	0.00 / 5.00	−5.00
	HX ↓	0.0 / 3.00	0.0 / 3.00	0.0 / 3.00	0.0 / 3.00	0.00 / 3.00	−3.00
Qwen2-7B	Acc ↑	68.4 / 63.80	54.3 / 52.80	97.6 / 89.20	35.4 / 32.20	63.93 / 59.50	+4.43
	BT ↓	0.1 / 5.86	0.6 / 5.86	0.0 / 5.86	0.3 / 5.86	0.25 / 5.86	−5.61
	AB ↓	0.0 / 2.50	0.0 / 2.50	0.0 / 2.50	0.0 / 2.50	0.00 / 2.50	−2.50
	HX ↓	0.3 / 8.00	0.7 / 8.00	0.0 / 8.00	0.0 / 8.00	0.25 / 8.00	−7.75

K Per-Domain Comparison with Weight-Modification Baselines

Tables 17 and 18 report per-domain values for the Antidote and linear-merging rows of Table 4 (LLaMA-3-8B, Split A). Antidote uses its one-shot Wanda-score pruning procedure with α=0.2; linear merging interpolates the aligned base and fine-tuned weights with coefficient 0.5. The zero inference overhead of both methods excludes the one-time computation needed to prune or merge each checkpoint.

Table 19 lists over-refusal for all evaluated post-hoc systems, including SafeLoRA, which was evaluated on over-refusal only. SafeLoRA has the lowest benign-refusal rate on OR-Bench but also the lowest refusal rate on harmful prompts.

L Independent-Judge Evaluation

Table 20 reports HyperSafe's harmful rates on the 12 held-out LLaMA-3-8B checkpoints under beaver-dam-7b and WildGuard. The judges disagree on which individual responses they flag, but split-level averages differ by at most 0.08 points, and the 12-checkpoint averages are identical on AdvBench and BeaverTails. Table 21 repeats the over-refusal evaluation under both judges; WildGuard yields slightly higher benign-refusal and slightly lower harmful-refusal rates, with a maximum difference of 1.67 points. OR-Bench produces high benign-refusal rates for most systems (53.7% for the aligned base model), so we compare against the base model rather than reading the absolute rate in isolation.

Table 17: HyperSafe / Antidote (α=0.2).

	BoolQ	ARC	Math	Alp	Avg
Acc↑	86.9/80.14	86.5/88.57	34.1/38.29	38.9/30.29	61.60/59.32
BT↓	0.1/9.00	0.0/8.29	0.0/7.43	0.1/6.29	0.05/7.75
AB↓	0.0/0.96	0.0/0.38	0.0/1.54	0.0/0.38	0.00/0.82
HX↓	0.0/4.33	0.0/3.67	0.0/6.00	0.0/3.33	0.00/4.33

Table 18: HyperSafe / linear merging (λ=0.5).

	BoolQ	ARC	Math	Alp	Avg
Acc↑	86.9/89.71	86.5/88.43	34.1/38.43	38.9/34.86	61.60/62.86
BT↓	0.1/9.71	0.0/7.86	0.0/8.29	0.1/11.43	0.05/9.32
AB↓	0.0/1.54	0.0/0.96	0.0/1.54	0.0/0.96	0.00/1.25
HX↓	0.0/4.33	0.0/3.00	0.0/6.67	0.0/8.67	0.00/5.67

Table 19: Refusal rates (%) on benign (↓) and harmful (↑) prompts (LLaMA-3-8B, Split A).

	XSTest		**OR-Bench**	
System	benign↓	unsafe↑	benign↓	toxic↑
Aligned base model (LLaMA-3-8B-Instruct)	5.2	95.0	53.7	97.0
Linear merging (λ=0.5)	2.0	92.5	51.0	95.5
Antidote (α=0.2)	3.2	93.5	53.0	95.5
SafeLoRA	6.8	80.0	34.1	80.8
HyperSafe (ours)	5.6	99.5	54.0	99.0

Table 20: Harmful rate (%) of HyperSafe on LLaMA-3-8B under two judges.

Bench.	**Judge**	**A**	**B**	**C**	**All 12**
BeaverTails	beaver-dam-7b	0.05	0.03	0.13	0.07
	WildGuard	0.05	0.05	0.10	0.07
AdvBench	beaver-dam-7b	0.00	0.00	0.00	0.00
	WildGuard	0.00	0.00	0.00	0.00
HEx-PHI	beaver-dam-7b	0.00	0.08	0.00	0.03
	WildGuard	0.08	0.00	0.08	0.05

Table 21: HyperSafe refusal rates (%) under two judges (LLaMA-3-8B, Split A).

	XSTest		**OR-Bench**	
Judge	ben.↓	uns.↑	ben.↓	tox.↑
beaver-dam	5.60	99.50	54.00	99.00
WildGuard	6.40	99.00	55.67	98.67

M Training and Inference Cost

All stages were run on a single A100 80GB GPU. Meta-training is a one-time cost per backbone family, paid by the provider that trains the hypernetwork: LoRA fine-tuning of the 18 training-domain checkpoints, ground-truth SSN training for each (3 epochs over ∼7k BeaverTails harmful prompts plus in-domain safe prompts), and hypernetwork training (∼230M parameters, 100 epochs). Table 22 reports the measured cost for LLaMA-3-8B, excluding LoRA fine-tuning. Per released checkpoint, a practitioner runs N_{cal}=50 calibration prompts through the checkpoint, one hypernetwork forward pass, and loads the generated SSN; end to end this takes under one minute and needs no gradients, safety data, or weight modification. At inference, the SSN adds 3.41%, 3.42%, and 3.45% FLOPs at 128, 256, and 512 prompt tokens (1.791 to 1.852 TFLOPs at 128 tokens). Latency is paid once per prompt at scoring time (Table 23); safe prompts then decode with the unmodified backbone. Table 24 compares the added inference overhead of all methods.

N Hypernetwork Training Stability

We hold the 18 fine-tuned checkpoints and their ground-truth SSNs fixed and retrain the hypernetwork end to end under three random seeds (LLaMA-3-8B, Split A), evaluating on the same four held-out checkpoints (Table 25). The accuracy standard deviation is at most 0.69 points and no seed exceeds a 0.14% harmful rate. We did not repeat LoRA fine-tuning or ground-truth SSN training across seeds.

Table 22: Meta-training and deployment cost (LLaMA-3-8B, one A100).

Stage	Cost	Frequency
GT SSN training	1.40 GPU-h $\times$ 18 = 25.2 h	one-time
Hypernetwork training	0.4 GPU-h	one-time
Subtotal	*25.6 GPU-h*	one-time
Hypernetwork pass	$\sim$1.4 s	per checkpoint

Table 23: Per-prompt latency at the median prompt length (LLaMA-3-8B).

Domain	Tokens	Backbone (ms)	+SSN (ms)	Δ (ms)
BoolQ	126	29.28	43.77	+14.49
NQ-Open	26	28.43	40.14	+11.71
SQuAD	181	36.00	46.26	+10.26

Table 24: Added inference overhead on LLaMA-3-8B (one A100), relative to running the fine-tuned model alone. Zero overhead for weight-modification methods excludes the one-time computation to modify each checkpoint.

Method	Added FLOPs	Added latency	Added memory
Linear merging, Antidote, SafeLoRA	0%	0 ms	0
HyperSafe SSN (ours)	+3.4%	+10–15 ms	+1–2 MB act., +3–4% params
Llama Guard 3	+244%	+59 ms	+16.2 GB
WildGuard	+175%	+388 ms	+14.6 GB

Table 25: HyperSafe across three hypernetwork-training seeds (LLaMA-3-8B, Split A). AdvBench and HEx-PHI harmful rates are 0.00 for every seed and domain.

Dataset	Metric	Seed 42	Seed 43	Seed 44	Mean $\pm$ SD
BoolQ	Acc $\uparrow$	86.30	87.52	86.91	86.91 $\pm$ 0.61
	BT $\downarrow$	0.00	0.00	0.14	0.05 $\pm$ 0.08
ARC	Acc $\uparrow$	86.09	87.03	86.43	86.52 $\pm$ 0.48
	BT $\downarrow$	0.00	0.00	0.00	0.00 $\pm$ 0.00
Comp. Math	Acc $\uparrow$	33.53	34.87	33.93	34.11 $\pm$ 0.69
	BT $\downarrow$	0.00	0.00	0.00	0.00 $\pm$ 0.00
Alpaca	Acc $\uparrow$	38.20	39.50	39.01	38.90 $\pm$ 0.66
	BT $\downarrow$	0.00	0.14	0.14	0.09 $\pm$ 0.08