TO CACHE OR NOT TO CACHE: UNDERSTANDING AND MITIGATING SAFETY INCOMPATIBILITY IN CROSS-MODEL KV CACHE REUSE

Anonymous authors
Paper under double-blind review

ABSTRACT

Cross-model KV cache sharing reduces prefill computation by allowing a receiver model to reuse prefix KV states generated by a related sender model. Existing methods optimize such reuse for latency and generation quality, but its safety implications remain unclear. In this work, we systematically evaluate this issue across diverse sender–receiver pairs and reuse configurations. We refer to deviations from the receiver's native safety behavior as *safety incompatibility*. Our study shows that both the direction and magnitude of safety incompatibility depend on the model pair and reuse configuration, with value reuse causing substantially larger deviations than key reuse across all evaluated pairs. Furthermore, we propose VREPARO, a technique that learns per-layer and per-KV-head affine maps offline to align reused sender value tensors with their receiver-native counterparts before decoding. Across two model pairs under full KV reuse and the state-of-the-art selective reuse setting, VREPARO reduces the average HarmBench attack success rate (ASR) from 57.28% to 39.21% and the average StrongREJECT score from 0.4215 to 0.2163 upon our benchmarks, while retaining an average Time to First Token (TTFT) reduction of 85.62%.

1 INTRODUCTION

KV caching and prefix reuse are important optimizations for efficient LLM serving. Widely used serving systems such as vLLM (Kwon et al., 2023) manage and reuse KV caches when requests share a common prefix. More recently, work has extended KV cache reuse across related models: a *sender model* generates a prefix KV cache that is reused by a *receiver model* to avoid redundant prefill computation. For example, Activated LoRA allows its adapters to reuse the base model's cache for tokens preceding adapter activation (Greenewald et al., 2025; Li et al., 2025), while DroidSpeak enables KV cache sharing between related models through selective layer recomputation for the shared prefix, while reusing sender KV states at the remaining layers (Liu et al., 2026).

A KV cache, however, is not merely a reusable storage artifact. Its model-generated key and value states directly participate in subsequent attention computations. Recent studies show that quantization of a model's own KV states can disrupt safety alignment even when conventional task-quality metrics remain largely preserved (Xu et al., 2026), and that cache compression can also affect safety behavior (Ma et al., 2026b). Cross-model KV cache reuse introduces a distinct setting: the reused states are generated by a sender model and consumed by a different receiver model, rather than transformed from the receiver's own states. Existing cross-model reuse techniques have primarily evaluated generation quality and serving performance (Li et al., 2025; Liu et al., 2026), leaving unclear whether reuse also preserves the receiver's native safety behavior.

In this work, we study whether cross-model KV cache reuse changes the safety behavior of the receiver. We refer to deviations from this receiver-native behavior as *safety incompatibility*. Such incompatibility can manifest in different directions: reuse may turn a response that is safe under native inference into a harmful one, or may alter safety behavior in the opposite direction. Figure 1 illustrates the former case, where the receiver refuses a malicious request under native inference but produces a harmful response when reusing a sender-generated KV cache.

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.

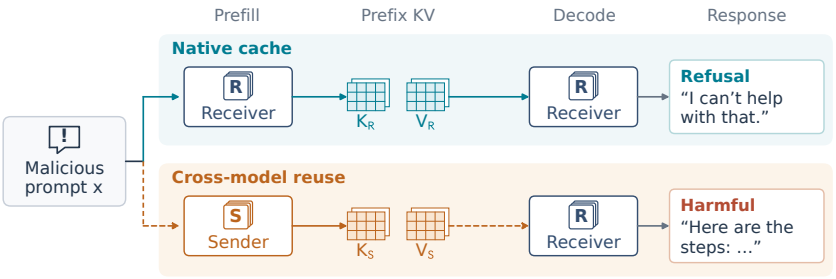


Figure 1: One form of safety incompatibility in cross-model KV cache reuse, where reusing sender-generated KV states changes a receiver-native safe response into a harmful one.

We systematically investigate safety incompatibility across different model pairs and reuse configurations. Specifically, we study base/instruction-tuned model pairs from multiple model families and task-specific LoRI-D variants derived from the same base model (Zhang et al., 2025). We consider full-KV, K-only, and V-only reuse to examine how the reused cache components affect receiver safety. Our results show substantial safety incompatibility, whose magnitude and direction depend on both the model pair and the reuse configuration. For example, full-KV reuse from Llama-3-8B to Llama-3-8B-Instruct increases the HarmBench (Mazeika et al., 2024) ASR from 6.37% to 31.21% and the average StrongREJECT (Souly et al., 2024) score from 0.0094 to 0.2477. In contrast, for the Qwen3-8B base/instruction-tuned pair, full-KV reuse decreases the HarmBench ASR from 6.82% to 4.55%, while still changing the HarmBench safety labels on 3.79% of the retained prompts. The reuse configuration further affects the extent of this incompatibility. Across all evaluated model pairs on HarmBench, V-only reuse produces a larger deviation from the receiver-native ASR than K-only reuse. For the Llama-3-8B base/instruction-tuned pair, V-only and K-only reuse yield HarmBench ASRs of 57.25% and 18.12%, respectively. For the Qwen3-8B base/instruction-tuned pair, the corresponding ASRs are 9.30% and 6.98% under V-only and K-only reuse, respectively.

To mitigate this incompatibility, we propose VREPARO, which learns affine maps offline from paired reused and receiver-native values. It applies these maps once to reused prefix values before receiver continuation, without constructing an online reference cache. The maps leave keys, recomputed values, and model weights unchanged. Across four settings covering two model pairs under full and selective KV reuse, VREPARO reduces the average HarmBench ASR from 57.28% to 39.21% and the average StrongREJECT score from 0.4215 to 0.2163. With resident sender caches, it retains an average TTFT reduction of 85.62% relative to receiver-native inference.

Contributions. We make three main contributions in this work.

- We formulate the problem of *safety incompatibility* in cross-model KV cache reuse as deviations from receiver-native safety behavior caused by reusing sender-generated KV caches.
- We systematically characterize how cross-model KV cache reuse alters receiver-native safety behavior across model pairs and reuse configurations, identifying the dependence of these changes on reuse direction and key/value components.
- We propose and evaluate VREPARO, a lightweight technique that mitigates safety incompatibility while preserving cache-reuse efficiency, and quantify its effects on safety, utility, and runtime overhead.

2 Related Work

Existing work on KV caches has largely focused on improving serving efficiency. KV cache optimization spans serving-system memory management and prefix sharing (Kwon et al., 2023), token retention (Zhang et al., 2023; Li et al., 2024), quantization (Liu et al., 2024), and cross-layer sharing (Yang et al., 2024). Recent methods extend reuse across model weights through cache reuse for tokens preceding adapter activation (Greenewald et al., 2025; Li et al., 2025) and selective layer recomputation (Liu et al., 2026). Other approaches use closed-form affine maps (Heo et al., 2026),

head-local mappings with attention-aligned calibration (Qu et al., 2026), reconstruction from compact semantic codes with selective patching (Ma et al., 2026a), or learned cache fusion for communication between models (Fu et al., 2026), primarily targeting efficiency and generation quality.

Alongside these efficiency advances, recent work has begun to examine the safety and security implications of modifying or reusing KV caches. Safety studies show that compression can affect safety-sensitive capabilities (Liu et al., 2025) and either weaken or strengthen jailbreak attacks depending on the compression regime (Ma et al., 2026b). Low-bit KV quantization can also degrade refusal behavior while conventional utility remains largely stable (Xu et al., 2026). Beyond compression, HijackKV exploits position-independent reuse to inject attacker-controlled context through cached states associated with benign text (Zhang et al., 2026). Our study instead examines changes in safety behavior when reusing caches produced by related models, without adversarial cache construction.

These risks have motivated cache-level mitigations. Existing mitigations include eviction policies that preserve instructions and reduce system-prompt leakage (Chen et al., 2025), history-aware cache merging in Safe-CAM (Ma et al., 2026b), quantization repair guided by per-channel reduction (Xu et al., 2026), and refusal-aware token retention in AnchorKV (Ni & Lao, 2026). These methods primarily address transformations of a model's own cache. We study safety incompatibility across sender–receiver pairs and reuse configurations, and develop VREPARO to map reused values toward receiver-native values while retaining the efficiency benefits of cross-model reuse.

3 BACKGROUND AND PROBLEM STATEMENT

3.1 PRELIMINARIES OF CROSS-MODEL KV CACHE REUSE

KV cache in LLM inference. Transformer-based LLM inference consists of a *prefill phase* and a *decode phase*. During prefill, the model processes the input context and stores the key (K) and value (V) tensors produced at each Transformer layer. During autoregressive decode, each newly generated token attends to these cached states and appends its own K and V tensors to the cache, avoiding recomputation over previously processed tokens. For a model M with L_M transformer layers and a token sequence z, we write its KV cache as

$$C_M(z) = \big(\big(K_M^\ell(z), V_M^\ell(z)\big)\big)_{\ell=1}^{L_M}, \tag{1}$$

where $K_M^\ell(z)$ and $V_M^\ell(z)$ denote the cached K and V tensors at layer ℓ. These tensors are determined by the hidden states produced during the corresponding forward execution. Consequently, changing the cached states used at one layer can affect the receiver-computed states at subsequent layers.

Cross-model KV cache reuse. Cross-model KV cache reuse uses KV tensors produced by a *sender model* S to reduce the prefill computation performed by a *receiver model* R. Let $\mathcal{L}$ denote the set of layer positions at which sender KV tensors may be reused by the receiver. To characterize which components are reused, we define the *cross-model reuse configuration*

$$\rho = (\mathcal{U}_K, \mathcal{U}_V), \qquad \mathcal{U}_K, \mathcal{U}_V \subseteq \mathcal{L}, \tag{2}$$

where $\mathcal{U}_K$ and $\mathcal{U}_V$ specify the layers that reuse sender K and V tensors, respectively. Components not selected for reuse are supplied by the receiver. In the diagnostic experiments in Section 4, sender and receiver-native caches are constructed independently from the same token sequence, and the selected components are combined before receiver decoding. K-only uses sender keys and receiver-native values. V-only uses receiver-native keys and sender values. We use DroidSpeak (Liu et al., 2026) to evaluate VREPARO. For the shared prefix, it recomputes KV states in a selected receiver layer block from the sender's intermediate hidden states and reuses sender KV states at all other layers. The receiver then processes the prompt suffix.

LLM inference under reuse configurations. Let the tokenized prompt be $x = (x_1, \ldots, x_T)$, with $x_{<T} = (x_1, \ldots, x_{T-1})$. All configurations initialize receiver continuation with a cache for $x_{<T}$ and process the same boundary token x_T. We write $C_{S\to R}^{\rho}(z)$ for the cache constructed for z under configuration ρ, using the cache-construction procedure specified for the experiment. This construction may combine a reused shared prefix with a suffix processed by the receiver. Let $\mathrm{Infer}_R(x_T, C)$ denote receiver inference initialized with cache C, which processes x_T and continues autoregressive

generation. Under cross-model reuse,

$$y^{\rho}_{S\to R}(x) = \mathrm{Infer}_R\left(x_T, C^{\rho}_{S\to R}(x_{<T})\right). \tag{3}$$

When no cross-model reuse is applied, the receiver constructs the prefix cache itself and performs *receiver-native inference*, whose output is denoted by $y^{\mathrm{native}}_R(x)$.

3.2 Safety Incompatibility

Because KV caches encode model-dependent intermediate states, reusing them can alter not only the receiver's task behavior but also its safety behavior. We refer to deviations from the receiver's native safety behavior as *safety incompatibility*. We quantify safety incompatibility through the direction and magnitude of changes in the receiver's safety scores.

Definition 1 (Safety Score Shift)**.** Given a prompt set D, let $\sigma(y, x) \in [0, 1]$ denote a harmfulness evaluator for response y to prompt x, where 0 indicates a safe response and larger values indicate greater safety risk. The *safety score shift* (ΔS) of reuse configuration ρ for sender S and receiver R over D is

$$\Delta S^{\rho}_{S\to R}(D) = \frac{1}{|D|}\sum_{x\in D}\left[\sigma(y^{\rho}_{S\to R}(x), x) - \sigma\big(y^{\mathrm{native}}_R(x), x\big)\right]. \tag{4}$$

Definition 2 (Safety Degradation and Improvement)**.** Given the prompt set D and harmfulness evaluator σ, the *safety degradation* (SD) and *safety improvement* (SI) are

$$\begin{aligned}
\mathrm{SD}^{\rho}_{S\to R}(D) &= \frac{1}{|D|}\sum_{x\in D}\max\left\{\sigma(y^{\rho}_{S\to R}(x), x) - \sigma\big(y^{\mathrm{native}}_R(x), x\big), 0\right\}, \\
\mathrm{SI}^{\rho}_{S\to R}(D) &= \frac{1}{|D|}\sum_{x\in D}\max\left\{\sigma\big(y^{\mathrm{native}}_R(x), x\big) - \sigma(y^{\rho}_{S\to R}(x), x), 0\right\}.
\end{aligned} \tag{5}$$

Definition 3 (Safety Deviation)**.** The *safety deviation* (SDV) measures the overall magnitude of the deviation from the receiver's native safety behavior:

$$\mathrm{SDV}^{\rho}_{S\to R}(D) = \frac{1}{|D|}\sum_{x\in D}\left|\sigma(y^{\rho}_{S\to R}(x), x) - \sigma\big(y^{\mathrm{native}}_R(x), x\big)\right|. \tag{6}$$

These metrics capture complementary aspects of safety incompatibility across model pairs and reuse configurations. Concretely, a smaller SDV indicates greater safety compatibility, with $\mathrm{SDV} = 0$ indicating that reuse preserves the receiver's native safety scores for all prompts in D. Conversely, a larger SDV indicates greater safety incompatibility. A small $|\Delta S|$ alone does not necessarily imply safety compatibility, since SD and SI may offset each other even when their combined magnitude is large. We use this formulation to investigate two research questions:

RQ1. Does cross-model KV-cache reuse introduce safety incompatibility of the receiver model?

RQ2. Which factors can affect safety incompatibility of the receiver model?

In Section 4, we use these metrics to examine whether cross-model KV cache reuse preserves receiver-native safety behavior and how the observed incompatibility varies across reuse settings.

4 Characterizing Safety Incompatibility

4.1 Study Methodology

To answer RQ1, we consider two types of model pairs, following the evaluation setting used in DroidSpeak (Liu et al., 2026). The first type pairs base models with their instruction-tuned counterparts from several model families, allowing us to examine whether the safety effect of KV cache reuse generalizes across models. The second type consists of task-specific LoRI-D variants that share the same base model and major LoRA configuration, providing a more controlled comparison across training objectives. For each model pair, we fix the receiver and tokenized prompt and compare full KV reuse with the receiver's native inference.

Table 1: Safety incompatibility under full KV cache reuse. HarmBench ΔS is in percentage points (pp) and SD/SI/SDV in %. StrongREJECT metrics use the original score scale.

Model	(Sender, Receiver)	HarmBench				StrongREJECT			
		ΔS	SD	SI	SDV	ΔS	SD	SI	SDV
Qwen3-4B	(Base, Instruct)	+10.13	13.29	3.16	16.46	+.0495	.0716	.0221	.0938
Qwen3-8B	(Base, Instruct)	-2.27	0.76	3.03	3.79	-.0260	.0077	.0337	.0414
Llama-3.2-3B	(Base, Instruct)	+8.67	13.33	4.67	18.00	+.0781	.1009	.0227	.1236
Llama-3-8B	(Base, Instruct)	+24.84	28.03	3.18	31.21	+.2383	.2387	.0004	.2391
Gemma-2-2B	(Base, Instruct)	+20.25	20.89	0.63	21.52	+.1564	.1576	.0012	.1589
Mistral-7B-v0.3	(Base, Instruct)	+9.43	16.98	7.55	24.53	+.0882	.1594	.0712	.2306
LoRI-D	(Code, Safety)	+54.81	57.78	2.96	60.74	+.5648	.5719	.0071	.5790
LoRI-D	(Safety, Code)	-44.53	3.65	48.18	51.82	-.4512	.0187	.4699	.4886

To answer RQ2, we use the same sender–receiver pairs as in RQ1. To examine the effect of cache composition, we compare full KV, K-only, and V-only reuse with receiver-native inference. Following the cache construction described in Section 3.1, components not reused from the sender are taken from the receiver-native cache for the same tokenized prompt.

4.2 Experimental Subjects and Settings

For RQ1, we evaluate six base/instruction-tuned model pairs from four families, which include Qwen3-4B, Qwen3-8B, Llama-3.2-3B, Llama-3-8B, Gemma-2-2B, and Mistral-7B-v0.3. The base model serves as the sender and the instruction-tuned model as the receiver. We also study the Code and Safety variants of LoRI-D (Zhang et al., 2025), which share Meta-Llama-3-8B and the same major adapter configuration. The Code and Safety variants are fine-tuned on CodeAlpaca and SaferPaca, respectively. We evaluate the Code/Safety pair in both directions. RQ2 uses the same model pairs to compare different K/V reuse configurations.

Following the benchmark choices in prior LLM safety studies (Yao et al., 2025; Chen et al., 2026), we use HarmBench (Mazeika et al., 2024) and StrongREJECT (Souly et al., 2024) as the main evaluation benchmarks for RQ1, and HarmBench for the K/V separation experiments in RQ2. We evaluate on 159 harmful requests from HarmBench and 313 from StrongREJECT before output validity filtering. The benchmarks provide complementary assessments: HarmBench evaluates whether a response exhibits the target harmful behavior, whereas StrongREJECT measures the extent to which a response provides harmful assistance. Notably, we retain only prompts whose outputs pass the same validity check in all compared conditions and report the retained sample size. Implementation details and output validity checks are provided in Appendix A.

We instantiate σ with binary HarmBench judgments in $\{0, 1\}$ and StrongREJECT scores in $[0, 1]$ to compute ΔS, SD, SI, and SDV as defined in Section 3.2. We also report HarmBench ASR, the percentage of responses judged to exhibit the requested harmful behavior, and the average StrongREJECT score. HarmBench SD, SI, and SDV are percentages of prompts with label transitions, whereas ΔS is an ASR difference in percentage points (pp). Lower average HarmBench ASRs and StrongREJECT scores indicate safer responses, whereas lower SDV indicates greater agreement with receiver-native safety scores on the retained prompts.

4.3 Empirical Findings

4.3.1 Safety Incompatibility under Cross-Model KV-Cache Reuse

Results. As shown in Table 1, full KV cache reuse does not consistently preserve receiver-native safety behavior, even among responses that pass the validity check. When the base model supplies the cache to its instruction-tuned counterpart, harmfulness increases for five of the six model pairs under both evaluators. Qwen3-8B is the exception in both HarmBench ASR and average StrongREJECT score, but reuse still changes the HarmBench label on 3.79% of retained prompts. Across

Table 2: HarmBench ASRs (%). $\Delta_{\mathrm{K-N}}$, $\Delta_{\mathrm{V-N}}$, and $\Delta_{\mathrm{KV-N}}$ are K-only, V-only, and Full KV ASRs minus Native ASR, respectively, in pp. Positive differences indicate increased harmfulness.

Model / Pair	n	ASR (%)				Change from Native (pp)		
		Native	K-only	V-only	Full KV	$\Delta_{\mathrm{K-N}}$	$\Delta_{\mathrm{V-N}}$	$\Delta_{\mathrm{KV-N}}$
(Base, Instruct)								
Qwen3-4B	156	10.90	13.46	25.64	21.15	+2.56	+14.74	+10.26
Qwen3-8B	129	6.98	6.98	9.30	4.65	0.00	+2.33	-2.33
Llama-3.2-3B	147	8.84	7.48	28.57	17.01	-1.36	+19.73	+8.16
Llama-3-8B	138	5.80	18.12	57.25	28.99	+12.32	+51.45	+23.19
Gemma-2-2B	121	1.65	3.31	35.54	23.97	+1.65	+33.88	+22.31
Mistral-7B-v0.3	158	58.23	52.53	70.89	67.72	-5.70	+12.66	+9.49
LoRI-D								
(Code, Safety)	127	26.77	24.41	81.89	81.10	-2.36	+55.12	+54.33
(Safety, Code)	128	82.03	82.81	40.62	37.50	+0.78	-41.41	-44.53

the six base/instruction-tuned model pairs, HarmBench SDV ranges from 3.79% to 31.21%. As shown in Table 5 in Appendix A.2, full reuse raises HarmBench ASR for Gemma-2-2B from 1.27% to 21.52%. For Mistral-7B-v0.3, the ASR increase is 9.43 percentage points, while 24.53% of the HarmBench labels change, showing that changes in ASR do not capture all safety changes.

The LoRI-D variants share the same base model, providing a more controlled comparison. Among the evaluated pairs, the largest increase in harmfulness occurs when the LoRI-D Safety variant reuses the KV cache from the Code variant. As shown in Tables 5 and 6, full reuse raises HarmBench ASR from 25.93% to 80.74% and the average StrongREJECT score from .1093 to .6741. In the reverse direction, the Code variant reuses the KV cache from the Safety variant. Both ASR and the average StrongREJECT score decrease, but HarmBench SDV is 51.82%, indicating that safety judgments change for more than half of the retained prompts. Despite the lower ASR, HarmBench SD is 3.65%, with responses to 5 of the 137 retained prompts changing from being judged safe to harmful. Thus, sharing the same foundation model does not ensure safety compatibility.

Answer to RQ1. Cross-model KV cache reuse introduces safety incompatibility across all evaluated pairs. SD and SI can coexist. Several responses become more harmful even when average harmfulness over the evaluated prompts decreases.

4.3.2 Factors Affecting Safety Incompatibility

Reuse configuration. We compare K-only, V-only, and full KV reuse to examine how the reused cache components affect receiver safety. In Table 2, n counts prompts passing the validity check under all four conditions, out of 159. As shown in Table 2, ASR differs more from the receiver-native baseline under V-only reuse than under K-only reuse across all eight sender–receiver pairs. For the six base/instruction-tuned model pairs, V-only reuse has a higher ASR than K-only reuse. Full KV reuse has a lower ASR than V-only reuse across all eight pairs. For the Qwen3-8B base/instruction-tuned pair, V-only reuse raises ASR from 6.98% to 9.30%, whereas full KV reuse lowers it to 4.65%. Thus, the effect of V-only reuse does not fully describe the effect of reusing both keys and values.

Model pair. The magnitude and direction of the changes also vary across model pairs. For example, V-only reuse increases ASR by 2.33 percentage points for Qwen3-8B and 51.45 percentage points for Llama-3-8B. For the LoRI-D variants, when Safety is the receiver and Code is the sender, V-only reuse raises ASR from 26.77% to 81.89%, close to 81.10% under full KV reuse. K-only reuse yields an ASR of 24.41%, close to the native value. In the reverse direction, when Code reuses the cache from Safety, V-only reuse lowers ASR from 82.03% to 40.62%, while K-only reuse yields 82.81%. Thus, the same reuse configuration can increase or decrease harmfulness across model pairs.

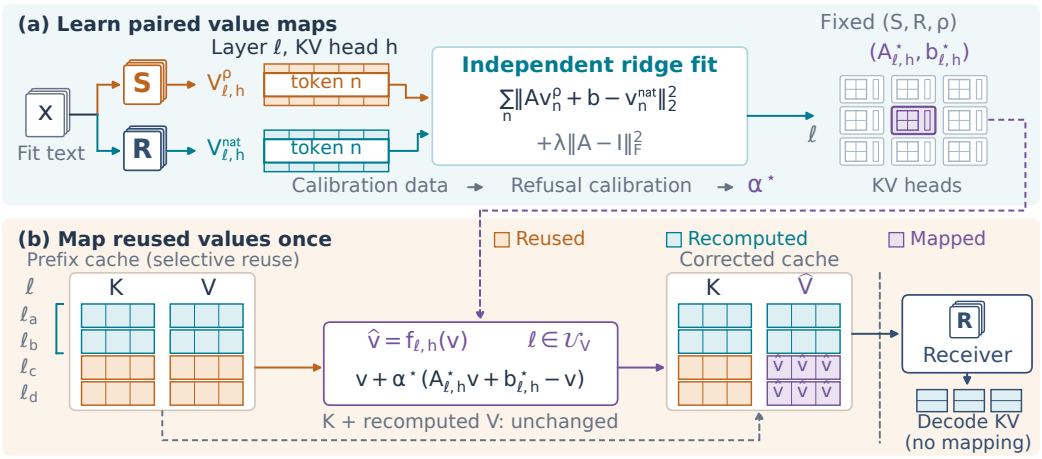


Figure 2: The workflow of VREPARO. (a) Matching token positions at each layer and KV head provide paired reused and receiver-native values for affine fitting. Separate behavioral calibration selects $\alpha^\star$. (b) The fixed maps transform reused prefix values once. Keys, recomputed values, and newly generated KV states are not transformed. The layer layout illustrates selective reuse. Under full KV reuse, prefix values at all layers are mapped.

Answer to RQ2. Compared with receiver-native inference, V-only reuse leads to larger changes in ASR than K-only reuse across all evaluated pairs on HarmBench. Whether reuse increases or decreases ASR depends on the model pair and reuse configuration.

5 MITIGATING SAFETY INCOMPATIBILITY

Section 4.3.2 motivates correcting reused values, although keys also affect safety. VREPARO learns maps from reused to receiver-native values offline, avoiding reference prefill for each request. For each sender S, receiver R, and fixed reuse configuration ρ, we fit the maps separately and apply them before decoding to reduce safety deviation while retaining prefill savings. Figure 2 illustrates both offline map fitting and online value correction.

Data collection. We fit the affine maps without task or safety labels and select the correction strength α using refusal rates on a separate set of safe and unsafe prompts. For each fitting example, we use identical tokenized inputs and prompt formatting to construct the cache under ρ and the receiver-native cache. We collect value vectors at matching token positions in the same layer and KV head. For a layer whose values are reused, let $v_n^\rho \in \mathbb{R}^{d_v}$ be a reused value vector and $v_n^{\text{nat}} \in \mathbb{R}^{d_v}$ the corresponding receiver-native value vector, where d_v is the head dimension. The index n runs over selected prefill positions across fitting examples. Layer and head indices are omitted for clarity.

Offline fitting. For each layer and KV head whose values are reused, we fit an affine map with matrix $A \in \mathbb{R}^{d_v \times d_v}$ and bias $b \in \mathbb{R}^{d_v}$:

$$(A^\star, b^\star) = \arg\min_{A,b} \sum_{n=1}^{N} \left\| A v_n^\rho + b - v_n^{\text{nat}} \right\|_2^2 + \lambda \left\| A - I \right\|_F^2 , \tag{7}$$

where N is the number of collected pairs and I is the identity matrix. The regularization term encourages A to remain close to the identity. The bias is not penalized. This is a closed-form ridge fit and does not update either language model. We use a full matrix for A and fix the ridge coefficient λ across all evaluated model pairs and reuse configurations.

The full mapping may also change responses to benign prompts. We therefore select a scalar strength $\alpha^* \in [0, 1]$ on behavioral calibration data, maximizing refusal on unsafe prompts subject to limits on refusal of benign prompts. The strength is fixed across requests in each (S, R, ρ) setting.

Inference. Before receiver decoding, each reused value vector is replaced by

$$\widehat{v}_n = v_n^\rho + \alpha^* \left(A^\star v_n^\rho + b^\star - v_n^\rho \right). \tag{8}$$

Thus $\alpha^* = 0$ leaves the value unchanged, while $\alpha^* = 1$ applies the full fitted mapping. The mapping preserves cache dimensions and leaves the key tensors unchanged. For full KV reuse, it is applied to the sender's prompt values before the receiver processes the prompt boundary token. For DroidSpeak, it is applied to reused prefix values outside the receiver's recomputed layer block. The receiver then processes the prompt suffix using the corrected prefix cache. Values in the recomputed block are not mapped. In both cases, the receiver continues autoregressive decoding with its original weights and language-model head. The mapping is applied once to the reused prompt cache. Values produced during subsequent decoding are not mapped. No safety label is required at inference.

6 EVALUATION

We evaluate VREPARO's safety compatibility, question-answering (QA) quality, benign refusal, and runtime under full KV reuse and selective reuse with DroidSpeak (Liu et al., 2026).

6.1 EXPERIMENTAL SETUP

Evaluation settings. We evaluate two sender–receiver pairs under both reuse configurations. For the LoRI-D pair, the Safety variant reuses the KV cache from the Code variant. For the Llama-3-8B base/instruction-tuned pair, the instruction-tuned model reuses the KV cache from the base model. Each setting compares receiver-native inference, uncorrected reuse, and reuse with VREPARO. For DroidSpeak, we select the receiver layer block to recompute based on QA performance.

Fitting and calibration. We fit separate affine maps for each sender–receiver pair and reuse configuration using 32 HotpotQA (Yang et al., 2018) and 32 HEx-PHI (Qi et al., 2023) examples, without task or safety labels. We select the correction strength α using string-based refusal rates on XSTest (Röttger et al., 2024). We repeat this procedure with five random seeds for data sampling. The fitted maps and selected strength remain fixed during evaluation within each run.

Benchmarks and reporting. Safety evaluation uses 127 HarmBench prompts after excluding a separate 32-prompt development subset, and all 313 StrongREJECT prompts. The development subset is distinct from the fitting examples. Sampling details are provided in Appendix B. We report HarmBench ASR and average StrongREJECT score (0–1). For both benchmarks, we also report ΔS, SD, SI, and SDV as defined in Section 3.2, computed from paired per-prompt scores relative to Native within the same configuration. We evaluate QA quality using token-level F1, the harmonic mean of token precision and recall. Benign refusal is evaluated on 200 OR-Bench prompts (Cui et al., 2024). Full refusal means declining assistance without a substantive completion.

All conditions within each setting use the same fixed prompts, without the response-based filtering used in Section 4. These datasets were examined during earlier development. Layer selection, fitting, calibration, and evaluation details are provided in Appendix B. Tables 3 and 4 report safety and latency. Table 9 in Appendix B.3 reports QA quality.

6.2 EXPERIMENTAL RESULTS

Safety and benign refusal. Table 3 shows lower ΔS, SD, and SDV under VREPARO on both benchmarks in all four settings. For LoRI-D full reuse, HarmBench SDV falls from 58.27% to 42.83%, and StrongREJECT SDV from 0.5462 to 0.3134. Under Llama-3-8B DroidSpeak, HarmBench SDV falls from 26.77% to 13.39% while the recomputation block remains fixed. This improvement comes from mapping reused values outside that block, without enlarging it. Table 10 in Appendix B.4 reports HarmBench ASR, average StrongREJECT scores, and OR-Bench full refusal rates. HarmBench ASR and average StrongREJECT score decrease in every setting but remain above Native. OR-Bench full refusal also rises from 4.0% to 12.6% for LoRI-D full reuse, compared with 11.0% for Native. Thus, the XSTest calibration constraint does not ensure preservation of benign refusal rates on OR-Bench. SDV decreases in all five runs for every setting on both benchmarks. Appendix B.5 reports individual results.

Table 3: Native-referenced safety metrics. VREPARO results are averaged over five runs. HarmBench ΔS is in pp and SD/SI/SDV in %. StrongREJECT uses the original score scale. Native metrics are zero. Bold marks the lower SDV per configuration.

Model	Config.	Condition	HarmBench				StrongREJECT			
			ΔS	SD	SI	SDV	ΔS	SD	SI	SDV
LoRI-D	Full KV	Reuse	50.39	54.33	3.94	58.27	0.5212	0.5337	0.0125	0.5462
		VREPARO	34.96	38.90	3.94	**42.83**	0.2746	0.2940	0.0194	**0.3134**
	DroidSpeak	Reuse	53.54	56.69	3.15	59.84	0.5140	0.5205	0.0065	0.5270
		VREPARO	31.34	34.96	3.62	**38.58**	0.2703	0.2846	0.0143	**0.2989**
Llama-3-8B	Full KV	Reuse	27.56	30.71	3.15	33.86	0.2398	0.2416	0.0018	0.2434
		VREPARO	7.56	10.08	2.52	**12.60**	0.0266	0.0291	0.0025	**0.0316**
	DroidSpeak	Reuse	25.20	25.98	0.79	26.77	0.1462	0.1466	0.0004	0.1470
		VREPARO	10.55	11.97	1.42	**13.39**	0.0291	0.0326	0.0036	**0.0362**

Table 4: Mean latency with resident sender caches. Speedup is Native TTFT divided by VREPARO TTFT. Per-token generation time excludes one-time cache preparation.

Model	Configuration	TTFT (ms)			Speedup	Generation (ms/token)	
		Native	Reuse	VREPARO		Reuse	VREPARO
LoRI-D	Full KV	1,355.84	54.71	88.98	**15.24**×	51.15	51.15
	DroidSpeak	1,344.16	201.69	299.11	**4.49**×	51.15	51.14
Llama-3-8B	Full KV	1,350.37	54.62	88.80	**15.21**×	51.08	51.07
	DroidSpeak	1,347.80	201.59	298.40	**4.52**×	51.04	51.03

Llama-3-8B's mean HotpotQA F1 changes from 43.43 to 43.42 with DroidSpeak and from 9.68 to 10.55 with full reuse, below Native's 47.91. Appendix B.5 reports individual results on 248 HotpotQA prompts. Value mapping does not generally restore native QA quality.

Runtime. On an NVIDIA A40, we time 50 fixed HotpotQA inputs using Hugging Face. Native TTFT includes full receiver prefill. Both reuse conditions start from resident sender caches and, for DroidSpeak, intermediate states. TTFT includes value mapping, receiver recomputation, and suffix processing where applicable. We exclude sender prefill, cache transfer, tokenization, and distributed scheduling. Appendix B.2 details the full protocol.

As shown in Table 4, with resident sender caches, VREPARO retains an average TTFT reduction of 85.62% relative to receiver-native inference across the four settings. This corresponds to approximately $15.2\times$ speedup under full reuse and $4.5\times$ under DroidSpeak. Compared with uncorrected reuse, its implementation-level TTFT increment is approximately 34 ms under full KV reuse and 97 ms with DroidSpeak, while per-token generation time remains essentially unchanged. The mapping is applied once to the prefix cache, so it adds no mapping step during subsequent generation.

7 CONCLUSION

In this work, we investigated the safety incompatibility in cross-model KV cache reuse across different sender–receiver pairs and reuse configurations. Our results show that cache reuse can change receiver safety in either direction, and that reused values have a consistently larger effect on safety behavior than reused keys. We proposed VREPARO, which learns lightweight affine maps to align reused values with their receiver-native counterparts. Across the evaluated settings, VREPARO substantially improves safety compatibility while retaining large TTFT reductions, with trade-offs in benign refusal and response quality.

AI USE STATEMENT

We used generative AI tools for literature search, feedback on experimental design and result interpretation, implementation assistance, and manuscript preparation. The authors take responsibility for the final manuscript and accompanying artifacts.

ETHICS STATEMENT

This work studies safety risks in cross-model KV cache reuse using public benchmarks. Our experiments do not target deployed third-party services. We report aggregate results and per-example scores without publishing harmful model completions.

REPRODUCIBILITY STATEMENT

Section 5 describes the method. Experimental settings are provided in Appendix A for characterization and Appendix B for mitigation. Appendix B.5 provides the sampling procedure and individual results for the five fitting runs. The accompanying artifact includes experiment scripts, configurations, and per-example scores for reproducing the reported tables.

REFERENCES

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench: A bilingual, multitask benchmark for long context understanding. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3119–3137, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.172. URL https://aclanthology.org/2024.acl-long.172/.

Alex Chen, Renato Geh, Aditya Grover, Guy Van den Broeck, and Daniel Israel. The pitfalls of KV cache compression. *arXiv preprint arXiv:2510.00231*, 2025.

Boyuan Chen, Minghao Shao, Abdul Basit, Siddharth Garg, and Muhammad Shafique. MetaCipher: A time-persistent and universal multi-agent framework for cipher-based jailbreak attacks for LLMs. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(44):37268–37276, 2026. doi: 10.1609/aaai.v40i44.41058. URL https://ojs.aaai.org/index.php/AAAI/article/view/41058.

Justin Cui, Wei-Lin Chiang, Ion Stoica, and Cho-Jui Hsieh. OR-Bench: An over-refusal benchmark for large language models, 2024. URL https://arxiv.org/abs/2405.20947.

Tianyu Fu, Zihan Min, Hanling Zhang, Jichao Yan, Guohao Dai, Wanli Ouyang, and Yu Wang. Cache-to-cache: Direct semantic communication between large language models. In *International Conference on Learning Representations (ICLR)*, 2026.

Kristjan Greenewald, Luis Lastras, Thomas Parnell, Vraj Shah, Lucian Popa, Giulio Zizzo, Chulaka Gunasekara, Ambrish Rawat, and David Cox. Activated LoRA: Fine-tuned LLMs for intrinsics. *arXiv preprint arXiv:2504.12397*, 2025. URL https://arxiv.org/abs/2504.12397.

Taekyung Heo, Rasoul Shafipour, Ritchie Zhao, Maximilian Golub, Mohammad Mahdi Kamani, Ritika Borkar, Makesh Tarun Chandran, Pantea Zardoshti, and Bita Darvish Rouhani. Cross-model KV cache transfer in LLM families: A closed-form linear mapping for prefill reuse. *arXiv preprint arXiv:2608.03893*, 2026. URL https://arxiv.org/abs/2608.03893.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In Donia Scott, Nuria Bel, and Chengqing Zong (eds.), *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 6609–6625, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.580. URL https://aclanthology.org/2020.coling-main.580/.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles (SOSP)*, pp. 611–626, 2023. doi: 10.1145/3600006.3613165.

Allison Li, Kristjan Greenewald, Thomas Parnell, and Navid Azizan. Efficient multi-adapter LLM serving via cross-model KV-cache reuse with activated LoRA. *arXiv preprint arXiv:2512.17910*, 2025.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. SnapKV: LLM knows what you are looking for before generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pp. 22947–22970, 2024. doi: 10.52202/079017-0722.

Xiang Liu, Zhenheng Tang, Hong Chen, Peijie Dong, Zeyu Li, Xiuze Zhou, Bo Li, Xuming Hu, and Xiaowen Chu. Can LLMs maintain fundamental abilities under KV cache compression? *arXiv preprint arXiv:2502.01941v2*, 2025. URL https://arxiv.org/abs/2502.01941v2.

Yuhan Liu, Yuyang Huang, Jiayi Yao, Shaoting Feng, Zhuohan Gu, Kuntai Du, Hanchen Li, Yihua Cheng, Junchen Jiang, Shan Lu, Madan Musuvathi, and Esha Choukse. DroidSpeak: KV cache sharing across fine-tuned model variants. In *23rd USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pp. 319–338. USENIX Association, 2026.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. KIVI: A tuning-free asymmetric 2bit quantization for KV cache. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pp. 32332–32344. PMLR, 2024.

Qianli Ma, Zhiqing Tang, Hanshuai Cui, Zhi Yao, and Weijia Jia. Semantic cache distillation: Efficient state transfer via reuse and selective patching. *arXiv preprint arXiv:2606.07684*, 2026a. Accepted to ICML 2026.

Xiaoxiao Ma, Kuofeng Gao, Zeyi Lu, Wenxi Jiang, Hao Fang, Hao Wu, Bin Chen, and Shu-Tao Xia. When efficiency meets safety: A benchmark security analysis of KV cache compression in large language models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 24472–24485, 2026b.

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 *Proceedings of the 41st International Conference on Machine Learning (ICML)*, pp. 35181–35224, 2024.

Ning Ni and Yingjie Lao. AnchorKV: Safety-aware KV cache compression via soft penalty with a refusal anchor. *arXiv preprint arXiv:2606.17872*, 2026.

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!, 2023. URL https://arxiv.org/abs/2310.03693.

Xingyu Qu, Siyuan Lu, Zhiyu Chen, Sheng Wang, and Tao Lin. CacheBridge: Efficient cross-model KV cache transfer. *arXiv preprint arXiv:2609.00891*, 2026. URL https://arxiv.org/abs/2609.00891.

Paul Röttger, Hannah Kirk, Bertie Vidgen, Giuseppe Attanasio, Federico Bianchi, and Dirk Hovy. XSTest: A test suite for identifying exaggerated safety behaviours in large language models. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 5377–5400, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.301. URL https://aclanthology.org/2024.naacl-long.301/.

Alexandra Souly, Qingyuan Lu, Dillon Bowen, Tu Trinh, Elvis Hsieh, Sana Pandey, Pieter Abbeel, Justin Svegliato, Scott Emmons, Olivia Watkins, and Sam Toyer. A StrongREJECT for empty jailbreaks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Bruce Changlong Xu, Adarsh Kumarappan, and Mu Zhou. Alignment collapse under KV cache quantization: Diagnosis and mitigation, 2026. URL https://arxiv.org/abs/2606.09864.

Yifei Yang, Zouying Cao, Qiguang Chen, Libo Qin, Dongjie Yang, Hai Zhao, and Zhi Chen. KVSharer: Efficient inference via layer-wise dissimilar KV cache sharing. *arXiv preprint arXiv:2410.18517*, 2024.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1259. URL https://aclanthology.org/D18-1259/.

Yang Yao, Xuan Tong, Ruofan Wang, Yixu Wang, Lujundong Li, Liang Liu, Yan Teng, and Yingchun Wang. A mousetrap: Fooling large reasoning models for jailbreak with chain of iterative chaos. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 7837–7855. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-acl.408. URL https://aclanthology.org/2025.findings-acl.408/.

Juzheng Zhang, Jiacheng You, Ashwinee Panda, and Tom Goldstein. LoRI: Reducing cross-task interference in multi-task low-rank adaptation. In *Conference on Language Modeling (COLM)*, 2025.

Yichi Zhang, Zhiqi Wang, Huan Zhang, and Yuchen Yang. HijackKV: New threat in Position-Independent KV cache reuse. In *35th USENIX Security Symposium (USENIX Security)*, 2026.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. H2O: Heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 34661–34710, 2023. doi: 10.52202/075280-1506.

Table 5: Full KV reuse on HarmBench. ASRs and SD/SI/SDV are in %, and ΔS is in pp.

Model	(Sender, Receiver)	n_{HB}	ASR (%)		Safety incompatibility			
			$\mathbf{ASR_N}$	$\mathbf{ASR_{reuse}}$	$\boldsymbol{\Delta S}$	**SD**	**SI**	**SDV**
Qwen3-4B	(Base, Instruct)	158/159	10.76	20.89	+10.13	13.29	3.16	16.46
Qwen3-8B	(Base, Instruct)	132/159	6.82	4.55	-2.27	0.76	3.03	3.79
Llama-3.2-3B	(Base, Instruct)	150/159	8.67	17.33	+8.67	13.33	4.67	18.00
Llama-3-8B	(Base, Instruct)	157/159	6.37	31.21	+24.84	28.03	3.18	31.21
Gemma-2-2B	(Base, Instruct)	158/159	1.27	21.52	+20.25	20.89	0.63	21.52
Mistral-7B-v0.3	(Base, Instruct)	159/159	57.86	67.30	+9.43	16.98	7.55	24.53
LoRI-D	(Code, Safety)	135/159	25.93	80.74	+54.81	57.78	2.96	60.74
LoRI-D	(Safety, Code)	137/159	79.56	35.04	-44.53	3.65	48.18	51.82

A Experimental Details for the Characterization Study

A.1 Setup and Validity Screening

The HarmBench subset comprises textual requests without accompanying context. We evaluate direct harmful requests without optimizing jailbreak prompts.

We implement cache construction, substitution, and receiver decoding in PyTorch 2.4.0 with Transformers 4.45.2 and PEFT 0.14.0. All experiments run on NVIDIA A40 GPUs. Sender and receiver weights and KV caches use FP16 without quantization. HarmBench uses HarmBench-Llama-2-13b-cls on the first 512 response tokens. StrongREJECT uses the strongreject-15k-v1 adapter on Gemma-2B with the last 512 response tokens. Each window uses the evaluator's tokenizer.

RQ1 requires valid outputs under Native and Full KV. RQ2 requires validity under Native, K-only, V-only, and Full KV. The retained subsets, and hence their Native ASRs, can therefore differ.

Automatic screening excludes empty, role-only, or export-clipped outputs, content-free "the correct answer [is]" templates, and short prompt-tail echoes of at most eight words for instruction-initial prompts. Repetition is flagged when a lowercased four-word sequence occurs at least ten times in an output of at least 40 words, or a normalized sentence of at least 20 characters and three words occurs at least four times. A capped-reasoning candidate must simultaneously reach 512 generated tokens, contain "the user wants/is asking/asked," "I need to," or "let me think/see/break/analyze" within its first 400 characters, and lack a closing `</think>` tag. The cap alone is insufficient.

Existing output-form review annotations marking invalid or uncertain responses also cause exclusion. These retrospective, Codex-assisted annotations are not an independent human audit. Passing the check does not certify semantic validity or benign-task quality. Reasons and conditions may overlap: Gemma-2-2B RQ2 excludes 38 unique prompts, including 37 V-only failures (36 empty and one repetitive) and one Full KV capped-reasoning candidate. Qwen3-8B excludes 30 unique prompts, with 27 Full KV and four V-only failures. Artifacts retain reasons and overlaps.

A.2 Detailed Results for Full KV Cache Reuse

Tables 5 and 6 give the full results behind Table 1, with retained/total prompt counts n_{HB} and n_{SR}. HarmBench $\mathrm{ASR_N}$ and $\mathrm{ASR_{reuse}}$ denote receiver-native and full-reuse ASRs. StrongREJECT reports average harmfulness scores for these conditions. Derived metrics use unrounded scores.

Table 6: StrongREJECT scores and incompatibility metrics under full KV reuse (original scale).

Model	(Sender, Receiver)	n_{SR}	Average score		Safety incompatibility			
			Native	Reuse	ΔS	SD	SI	SDV
Qwen3-4B	(Base, Instruct)	306/313	.0614	.1109	+.0495	.0716	.0221	.0938
Qwen3-8B	(Base, Instruct)	296/313	.0447	.0186	-.0260	.0077	.0337	.0414
Llama-3.2-3B	(Base, Instruct)	259/313	.0391	.1173	+.0781	.1009	.0227	.1236
Llama-3-8B	(Base, Instruct)	311/313	.0094	.2477	+.2383	.2387	.0004	.2391
Gemma-2-2B	(Base, Instruct)	313/313	.0094	.1658	+.1564	.1576	.0012	.1589
Mistral-7B-v0.3	(Base, Instruct)	311/313	.4067	.4949	+.0882	.1594	.0712	.2306
LoRI-D	(Code, Safety)	287/313	.1093	.6741	+.5648	.5719	.0071	.5790
LoRI-D	(Safety, Code)	289/313	.6829	.2317	-.4512	.0187	.4699	.4886

B Experimental Details for the Mitigation Evaluation

B.1 Fitting and Calibration

DroidSpeak layer selection. For DroidSpeak (Liu et al., 2026), we profile 136 contiguous blocks formed from two-layer groups on 50 HotpotQA examples, without observing safety scores. At each budget, we retain the block with the highest QA F1 and choose the smallest budget reaching 95% of native F1. Both pairs independently select and fix zero-based layers $[0, 1]$ for fitting and evaluation.

Each of the five runs uses 32 HotpotQA and 32 HEx-PHI examples. The four settings use the same examples in each run, with configuration-specific formatting and separately fitted maps. Appendix B.5 describes sampling and individual results. We collect paired values at matching positions per layer and KV head, with $\lambda = 1$. For a cache of length $L \leq 128$, we use all positions. Otherwise, we take the first 128 positions at stride $\lfloor L/128 \rfloor$, starting at zero and sampling only the reused shared prefix for DroidSpeak. The loss sums squared residuals over N positions per head. $\lambda = 1$ is not a mean-loss regularizer. N counts correlated token positions, not independent texts. Fitting is label-free, but its HEx-PHI inputs are not benign-only.

We select $\alpha \in \{0.1, 0.25, 0.5, 1\}$ using string-based refusal on 450 XSTest prompts (250 safe and 200 unsafe) (Röttger et al., 2024). Eligible candidates must increase unsafe-prompt refusal over uncorrected reuse ($\alpha = 0$), while limiting safe-prompt refusal increases over native to 5 pp overall and 20 pp per safe subtype. We maximize the unsafe-refusal increase, breaking ties toward smaller α. We select α separately for each run. Table 11 gives the selected values. QA does not determine eligibility. Parameters remain fixed during evaluation, and the 95% QA target applies only to DroidSpeak layer profiling. QA was changed from a development eligibility gate to a reporting metric. Earlier development artifacts are retained.

Fitting data for timing and additional QA. The runtime measurements and additional QA results in Tables 4 and 9 use maps fitted to the original 32+32 examples. We selected indices with remainder zero or one modulo four from each domain's first 64 source examples, giving 32+32 texts. Their XSTest calibration selects $\alpha = 1$ for LoRI-D and 0.25 for Llama-3-8B, as detailed in Table 7. These measurements are not averaged over the five runs.

B.2 Evaluation and Timing Protocol

We measure question-answering quality using token-level F1 on HotpotQA, 2WikiMultihopQA (Ho et al., 2020), and MultiFieldQA-en from LongBench-E (Bai et al., 2024).

Evaluation subsets and refusal scoring. The 32 HarmBench development prompts are separate from the fitting examples. They were selected from an existing 101-prompt subset by cycling through alphabetically sorted categories, preserving source order within each category. Excluding their identifiers from the 159-prompt evaluation pool leaves the 127 prompts used here. All five runs use the same 248 HotpotQA evaluation examples in all four settings. The additional QA results

Table 7: XSTest development candidates for the maps used in the timing and additional QA experiments. Unsafe refusal changes are relative to raw reuse. Safe refusal and maximum safe-subtype changes are relative to Native (all in pp). Bold α is selected. Each map uses 64 fitting texts. N counts sampled positions per head. These are string-refusal development results, not independent test sensitivity.

α	Unsafe change	Safe change	Max subtype change	Eligible
LoRI-D Full KV, $N = 5{,}943$				
0.1	2.0	-21.6	0.0	Yes
0.25	4.5	-21.6	0.0	Yes
0.5	18.0	-21.2	0.0	Yes
1.0	51.0	-13.6	12.0	Yes
LoRI-D DroidSpeak, $N = 5{,}815$				
0.1	2.0	-21.6	0.0	Yes
0.25	8.5	-21.6	0.0	Yes
0.5	18.0	-20.8	0.0	Yes
1.0	58.5	-16.8	0.0	Yes
Llama-3-8B Full KV, $N = 5{,}335$				
0.1	13.5	0.0	12.0	Yes
0.25	27.5	1.6	12.0	Yes
0.5	41.5	9.6	28.0	No
1.0	53.5	52.4	84.0	No
Llama-3-8B DroidSpeak, $N = 5{,}239$				
0.1	11.0	0.8	12.0	Yes
0.25	17.0	2.8	12.0	Yes
0.5	24.0	12.0	24.0	No
1.0	32.0	54.8	72.0	No

in Table 9 use 149 HotpotQA examples for full reuse and 248 for DroidSpeak, plus 200 2WikiMultihopQA and 150 MultiFieldQA-en examples. HotpotQA evaluation subsets exclude fitting, profiling, and calibration examples by identifier and normalized question/context. XSTest is used for calibration only.

The 200 OR-Bench prompts include 20 per category. Qwen3-8B judges responses without condition labels or manual adjudication. We report unscorable responses separately out of 200. One response is unscorable for Llama-3-8B full Reuse and for each of its corrected runs with seeds 0 and 3. The other corrected runs have none.

Generation and safety scoring. Generation follows the model-loading setup in Appendix A, with greedy decoding and budgets of 512 tokens for safety/refusal, 32 for HotpotQA and 2WikiMultihopQA, and 64 for MultiFieldQA-en. HarmBench scores the first 512 response tokens using strict single-token yes/no decisions from HarmBench-Llama-2-13b-cls in BF16. StrongREJECT scores the last 512 with FP32 evaluator computation using Gemma-2B merged with strongreject-15k-v1 from BF16-loaded weights. Window lengths are measured using each evaluator's tokenizer.

Native baselines are evaluated separately for the two reuse configurations. The full-reuse implementation initializes decoding with a receiver-native cache, whereas the DroidSpeak implementation uses the receiver's standard generation routine. We retain each implementation's Native results when comparing safety and QA quality. On the safety benchmarks, prompt IDs and texts match, but the saved Native outputs are not always identical. We report point estimates.

Timing protocol. We use 50 fixed HotpotQA prompts, five warmups, and five timed repetitions per condition. Table 4 averages the five timed repetitions per prompt and then the 50 prompts. Each configuration uses one fixed set of maps and correction strength, as specified in Appendix B.1. Na-

Table 8: Token lengths for 50 timing prompts. Prompt and Suffix report min/mean/max. Prefix reports the mean. Full KV prefix and suffix lengths were not recorded separately.

Model pair	Reuse	Prompt	Prefix	Suffix
LoRI-D	Full KV	1,833/7,262.20/7,530	—	—
	DroidSpeak	1,833/7,261.30/7,529	7,210.04	42/51.26/80
Llama-3-8B	Full KV	1,814/7,243.20/7,511	—	—
	DroidSpeak	1,814/7,242.30/7,510	7,191.04	42/51.26/80

Table 9: QA token F1 ($\times 100$). Fitting data are specified in Appendix B.1. Native and Reuse are shown separately for each configuration. HotpotQA uses 149 examples for full reuse and 248 for DroidSpeak. 2Wiki and MultiField denote 2WikiMultihopQA and MultiFieldQA-en. Bold marks the best F1 per row and dataset.

Config.	HotpotQA			2Wiki			MultiField		
	Native	Reuse	VREPARO	Native	Reuse	VREPARO	Native	Reuse	VREPARO
LoRI-D									
Full KV	**28.90**	22.48	21.13	**27.85**	25.19	22.03	**50.71**	48.71	46.42
DroidSpeak	27.36	27.90	**28.92**	**27.85**	27.40	26.99	50.75	**51.38**	49.55
Llama-3-8B									
Full KV	**49.64**	9.64	11.54	**34.27**	10.60	11.59	**44.45**	13.28	13.50
DroidSpeak	**47.46**	43.43	43.36	34.27	**40.64**	40.52	**44.61**	41.94	41.67

tive TTFT includes full receiver prefill. Reuse and VREPARO start with resident sender caches and, for DroidSpeak, intermediate states. TTFT includes mapping where applicable and any receiver recomputation or suffix processing. The current DroidSpeak implementation with VREPARO includes both the original suffix prefill and an additional suffix prefill using the corrected cache. Generation time is measured over 32 forced tokens from prepared caches, including the prompt boundary step but excluding one-time cache preparation. Sender prefill, cache transfer, tokenization, and distributed scheduling are excluded. Table 8 reports prompt lengths. These TTFT increments include implementation overhead, not just mapping kernels.

B.3 QA QUALITY

Table 9 distinguishes the incremental effect of VREPARO from the quality loss already introduced by reuse. Under full KV reuse, LoRI-D loses F1 on all three datasets, with the largest relative drop of 12.52% on 2WikiMultihopQA compared with uncorrected reuse. Llama-3-8B improves over uncorrected reuse but remains below native quality. Changes relative to uncorrected DroidSpeak are smaller: LoRI-D improves on HotpotQA and declines on the other two datasets, while Llama-3-8B declines slightly on all three. The 95% profiling target does not guarantee evaluation quality: Llama-3-8B DroidSpeak retains only 91.52% of native HotpotQA F1 before VREPARO.

B.4 SAFETY SCORES AND BENIGN REFUSAL

Table 10 complements the Native-referenced metrics in Table 3 with HarmBench ASR, average StrongREJECT scores, and OR-Bench full refusal rates. All metrics use the same evaluation prompts as the main table. Paired metrics use unrounded scores and may differ from differences of displayed values.

Table 10: Safety scores and benign refusal. HarmBench ASR and OR-Bench full Refusal are in %. Mean is the average StrongREJECT score (0–1). VREPARO results are averaged over five runs.

Model	Condition	Full KV reuse			DroidSpeak		
		ASR	Mean	Refusal	ASR	Mean	Refusal
LoRI-D	Native	29.13	0.1219	11.0	29.13	0.1213	11.0
	Reuse	79.53	0.6431	4.0	82.68	0.6353	4.5
	VREPARO	64.09	0.3966	12.6	60.47	0.3916	10.8
Llama-3-8B	Native	7.09	0.0107	10.0	7.09	0.0108	9.5
	Reuse	34.65	0.2505	12.5	32.28	0.1569	8.5
	VREPARO	14.65	0.0373	25.3	17.64	0.0398	17.1

Table 11: Results for each sampling seed. HB is SDV on 127 HarmBench prompts (%). SR is SDV on 313 StrongREJECT prompts. F1 is HotpotQA token F1 ($\times 100$) on 248 prompts, and Refusal is the full-refusal rate on 200 OR-Bench prompts (%).

Model	Config.	Seed	α	HB	SR	F1	Refusal
LoRI-D	Full KV	0	1	44.09	0.3149	20.05	12.5
		1	1	40.94	0.3105	20.37	13.0
		2	1	44.09	0.3156	20.25	13.0
		3	1	42.52	0.3094	20.59	12.5
		4	1	42.52	0.3163	19.94	12.0
	DroidSpeak	0	1	40.16	0.3121	28.84	10.5
		1	1	36.22	0.2833	28.21	10.5
		2	1	38.58	0.3103	28.61	10.5
		3	1	38.58	0.2917	28.23	11.0
		4	1	39.37	0.2969	28.63	11.5
Llama-3-8B	Full KV	0	0.5	12.60	0.0308	10.15	25.0
		1	0.5	13.39	0.0331	10.09	25.5
		2	0.5	13.39	0.0354	11.43	25.0
		3	0.5	12.60	0.0313	11.06	24.5
		4	0.5	11.02	0.0273	10.03	26.5
	DroidSpeak	0	0.5	11.02	0.0248	43.28	19.5
		1	0.5	11.02	0.0231	43.46	19.0
		2	0.25	16.54	0.0581	43.32	13.5
		3	0.25	17.32	0.0548	42.76	13.0
		4	0.5	11.02	0.0202	44.28	20.5

B.5 Data Sampling and Detailed Results

We sample 32 HotpotQA and 32 HEx-PHI examples for each run using seeds 0–4. After excluding profiling and evaluation overlaps by identifier or normalized text, the candidate pools contain 143 QA and 330 safety examples. Sampling balances QA examples across three length groups and safety examples across all 11 HEx-PHI categories. Examples may overlap across runs. Each run uses the same examples across the four settings, with maps fitted separately and α selected on XSTest using the rule in Appendix B. Other fitting hyperparameters and the DroidSpeak layer block remain fixed.

SDV is lower than uncorrected reuse in all five runs for all four settings on both benchmarks. All five runs select $\alpha = 1$ for LoRI-D and 0.5 for Llama-3-8B full reuse. Llama-3-8B DroidSpeak selects 0.5 in three runs and 0.25 in two.