KVRECOVER: CROSS-MODEL PROMPT RECOVERY FROM LEAKED KV CACHES

Anonymous authors
Paper under double-blind review

ABSTRACT

Key–value (KV) caches retain input information that can be exploited to reconstruct private prompts. However, reconstruction attacks that rely on a large source model can require substantial GPU resources. We introduce KVRecover, a cross-model prompt-reconstruction method that enables a smaller frozen receiver to recover input content from complete leaked KV caches. Its model-pair-specific converter uses depth-dependent projection to map source states into the receiver's representation space, while local cross-layer fusion coordinates the converted states. Progressive supervision combines cache alignment, attention alignment, and text reconstruction to optimize the converted caches for input recovery. Once trained, the converter can be distributed and reused on unseen prompts within the same model pair, without source-model inference or further converter optimization during recovery. Experiments across three Qwen3 model pairs and four datasets achieve a macro-average BERTScore F1 of up to 0.834. Recovery from Qwen3-235B-A22B caches using Qwen3-14B further demonstrates the feasibility of the attack across a substantial model-size gap. These findings show that leaked caches can remain vulnerable even to attackers unable to deploy the source model.

1 INTRODUCTION

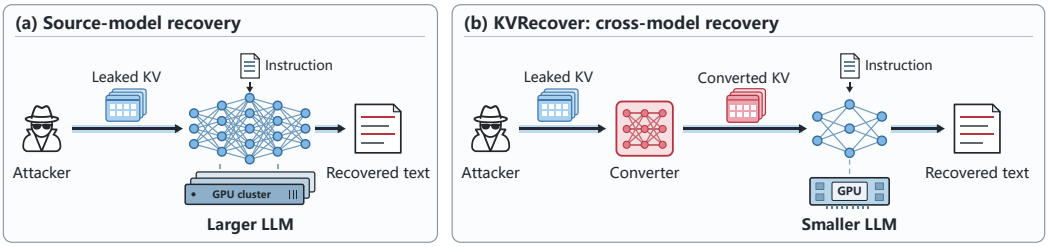


Figure 1: (a) Prompt reconstruction with the large source LLM requires substantial GPU resources. (b) KVRecover uses a pretrained converter with a smaller receiver, reducing GPU requirements without source-model inference during recovery.

Large language models (LLMs) are increasingly deployed in interactive applications, where efficient inference is essential to delivering responsive services at scale. The key–value (KV) cache improves inference efficiency by storing the attention keys and values of previously processed tokens, avoiding their repeated computation during autoregressive generation (Kwon et al., 2023). These cached representations can be reused across requests with shared prompt segments (Zheng et al., 2024; Gim et al., 2024), or transferred between prefill and decoding workers on separate GPUs (Zhong et al., 2024). Cross-model cache conversion further extends their use to LLMs with different internal representations, enabling direct semantic communication through transformed KV caches (Fu et al., 2026). As KV caches are reused and transferred across these settings, protecting the confidentiality of the input information they encode becomes an important concern for LLM security.

Prior work has demonstrated a range of KV cache attacks, including cache manipulation, side-channel inference, and direct input reconstruction. Cache manipulation attacks alter model outputs

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.

through prefix-cache hash collisions (Wu et al., 2026), poisoned reusable cache chunks (Zhang et al., 2026), or perturbations to cached values (Nahian et al., 2026). Side-channel attacks exploit KV cache sharing in multi-tenant services to infer private prompts without reading the cached tensors. PromptPeek exploits differences in request completion order caused by cache-aware scheduling (Wu et al., 2025), while timing-based attacks infer cache hits from response latency differences (Song et al., 2025; Zheng et al., 2027). Direct reconstruction attacks operate on exposed KV tensors, recovering input content through inversion, candidate-cache matching, or cache injection (Luo et al., 2026). In cache injection, the attacker loads the leaked cache into a local model as prior context and prompts it to recover the hidden input. This recovery path therefore requires the attacker to run a local model compatible with the leaked cache (Luo et al., 2026).

However, the recovery path illustrated in Figure 1(a) requires executing the source model, which can impose a substantial GPU resource burden when the model is large. In GPU-resident execution, model weights must fit alongside the leaked KV states and decoding workspace (Kwon et al., 2023; Pope et al., 2023). For example, a 70-billion-parameter model requires approximately 140 GB for weights alone at 16-bit precision, already exceeding the capacity of a single 80-GB GPU. Reusing the leaked cache avoids recomputing the prompt's KV states, but each generated token still requires a decoding step through the large model (Kwon et al., 2023; Pope et al., 2023). This dependence on large-model execution makes computational resources an important factor in assessing the privacy risks of leaked KV caches. These resource constraints raise a central question:

Can smaller models reconstruct prompts from leaked KV caches without source-model inference?

We present KVRecover, a cross-model prompt-reconstruction method that enables a smaller LLM to recover input content from leaked KV caches. As illustrated in Figure 1(b), a model-pair-specific converter transforms the leaked states into a cache usable by a smaller frozen receiver. Once trained, the converter can be reused on unseen prompts within the same model pair. Subsequent attackers execute only the converter and receiver during recovery, avoiding the need to load or run the source model and thereby lowering the GPU resource barrier to prompt reconstruction.

Our contributions are as follows:

- **Cross-model reconstruction scenario.** We introduce a scenario in which attackers reuse a compatible pretrained converter to recover prompts through a smaller receiver without source-model inference during recovery.
- **Cache conversion for reconstruction.** We design a converter that unifies depth-dependent projection, local cross-layer fusion, and progressive supervision to reconstruct prompts through a frozen receiver.
- **Effective reconstruction across model pairs.** KVRecover outperforms C2C-adapted across three same-family model pairs and four datasets, achieving a macro-average BERTScore F1 of up to 0.834. Experimental results in the 235B−14B setting further demonstrate the feasibility of the proposed attack scenario.

2 RELATED WORK

Text Reconstruction from Learned Representations. Embedding inversion recovers input words (Song & Raghunathan, 2020) or reconstructs sentences through generative decoding (Li et al., 2023) and iterative correction (Morris et al., 2023). To relax attack requirements, surrogate-based transfer avoids victim-model queries (Huang et al., 2024), while few-shot alignment (Chen et al., 2025b) and zero-shot inversion (Zhang et al., 2025) reduce attack-specific training requirements. Multilingual studies examine reconstruction across languages and scripts (Chen et al., 2024; 2025a). Whereas these studies invert text embeddings, our work couples text reconstruction with the layer-wise compatibility needed to use converted KV states in the receiver's attention computation.

Security and Privacy Risks of KV Caches. Cache manipulation alters model outputs through bit-level KV perturbations (Nahian et al., 2026), prefix-cache hash collisions (Wu et al., 2026), or poisoned reusable chunks (Zhang et al., 2026). Side-channel attacks infer private prompts from shared-cache latency (Zheng et al., 2027; Song et al., 2025) or cache-aware scheduling (Wu et al., 2025). Audits also identify cross-user prompt caching in deployed APIs (Gu et al., 2025). With

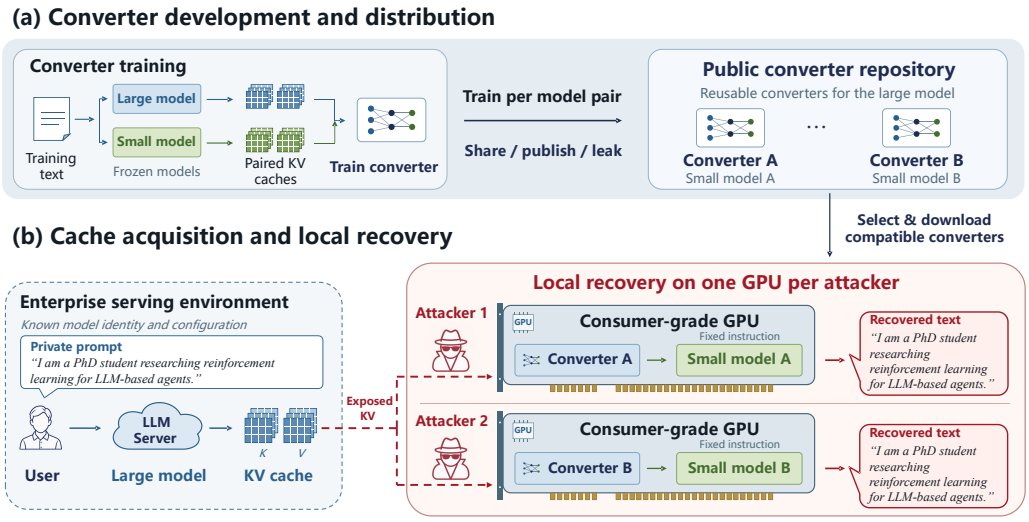


Figure 2: (a) A developer trains model-pair-specific converters that can be shared, published, or leaked for reuse. (b) Subsequent attackers use compatible converters and smaller receivers on local consumer GPUs to reconstruct private prompts from exposed KV caches.

exposed tensors, Shadow in the Cache studies inversion, candidate-cache matching, and cache injection (Luo et al., 2026). Its injection attack uses a compatible local model to reconstruct cached context. Our work examines whether this reconstruction threat persists when attackers lack the resources to deploy the source model.

KV-Cache Reuse and Cross-Model Transfer. Within a model, prefix sharing (Kwon et al., 2023; Zheng et al., 2024) and reusable prompt modules (Gim et al., 2024) reduce redundant prefill. CacheGen supports efficient cache delivery through compression and streaming (Liu et al., 2024), while CacheBlend fuses cached chunks using selective recomputation (Yao et al., 2025). DroidSpeak shares caches across fine-tuned model variants (Liu et al., 2026). For heterogeneous models, C2C learns cache projection and fusion (Fu et al., 2026), while See What I See combines reconstruction of receiver KV states with generation training (Chen et al., 2026). Whereas these methods prioritize inference efficiency or task performance, our work optimizes cross-model cache conversion to reconstruct the input encoded in leaked caches.

3 BACKGROUND

3.1 THREAT MODEL

We consider a prompt-reconstruction scenario in which converter development is separated from subsequent attacks, allowing reconstruction capability to be distributed and reused. In Figure 2(a), a developer trains model-pair-specific converters using authorized text and frozen source and receiver models, without access to the target user's private prompt. Once shared, published, or leaked, these converters allow other parties to reconstruct unseen prompts within the supported model pairs without bearing the cost of converter training. A single trained converter can thus support repeated attacks on different leaked caches.

Figure 2(b) illustrates the resulting threat. An attacker holding an exposed cache runs a compatible converter and a smaller receiver locally, avoiding the GPU resources required for source-model execution. Different converters for the same source model provide alternative recovery paths through different smaller receivers, allowing attackers to choose configurations that fit their hardware budgets. Converter dissemination therefore extends reconstruction capability to more resource-limited adversaries, while receiver diversity creates multiple potential routes for recovering private content from the same leaked cache.

Adversary's objective. The adversary aims to recover the wording and semantic content of an unseen private prompt from its KV cache.

Access assumptions. The adversary has a complete, unencrypted KV cache, the source model's identity and configuration, and the metadata needed to interpret the cache. A compatible pretrained converter and its associated receiver are available. The cache acquisition mechanism is outside our scope.

Recovery constraints. The adversary runs only the converter and frozen receiver with a fixed recovery instruction, without querying or executing the source model or updating the converter. The receiver can use its own generated history, but the original prompt, its token IDs, and a receiver-native cache computed from it are unavailable during recovery.

3.2 Problem Formulation

Let M_s and M_r denote the frozen source and receiver models. For $a \in \{s, r\}$, let L_a, H_a, and d_a denote the attention-layer count, KV-head count, and head dimension. KV-head and query-head counts may differ under grouped-query attention (Ainslie et al., 2023).

For a prompt x, the source cache is

$$C_s(x) = \left(\left(K_s^{(\ell)}, V_s^{(\ell)}\right)\right)_{\ell=1}^{L_s}, \qquad K_s^{(\ell)}, V_s^{(\ell)} \in \mathbb{R}^{H_s \times n \times d_s}, \tag{1}$$

where the batch dimension is omitted and n counts cached source positions, including prompt-template tokens. The prompt x denotes text, and the two models need not share a tokenizer.

We seek a model-pair-specific converter T_θ that maps the source cache to receiver-compatible states:

$$\widetilde{C}_r = T_\theta(C_s(x); m) = \left(\left(\widetilde{K}_r^{(j)}, \widetilde{V}_r^{(j)}\right)\right)_{j=1}^{L_r}, \tag{2}$$

where θ denotes the learnable parameters, m contains cache-layout and positional metadata, and $\widetilde{K}_r^{(j)}, \widetilde{V}_r^{(j)} \in \mathbb{R}^{H_r \times n \times d_r}$. The converted states preserve the source-position grid and act as a continuous prefix for M_r.

Given an authorized corpus $\mathcal{D}_{\text{train}}$ and access to both frozen models, we optimize θ so that M_r, conditioned on $\widetilde{C}_r$ and a fixed recovery instruction, can reconstruct x. The converter must accommodate structural differences while preserving information needed for text recovery.

4 Method

4.1 Reconstruction Challenges

Source and receiver models can differ in layer count, KV-head structure, and internal representations. Matching cache dimensions is therefore only one requirement: the converted states must also be usable by the receiver's frozen attention layers. KVRecover establishes structural correspondence before projecting source states into the receiver's representation space, with local cross-layer fusion coordinating the projected states.

Matching cache tensors numerically does not directly optimize recovery of the original text. Converted keys influence attention allocation, while converted values supply the retrieved content; both affect the receiver's subsequent predictions. KVRecover first uses cache regression to establish the representation mapping. Attention supervision then aligns attention distributions under shared fixed probe queries while retaining cache regression for both streams. Finally, text reconstruction and output distillation optimize recovery through the frozen receiver.

4.2 KVRecover Converter

Figure 3 shows the converter, applied separately to Key and Value with independent parameters. For $Z \in \{K, V\}$, let $Z_j^s \in \mathbb{R}^{n \times H_s \times d_h}$ denote source-layer states and $d = H_r d_h$ the flattened receiver width, where d_h is the common head width. The following equations describe either stream, with stream superscripts omitted.

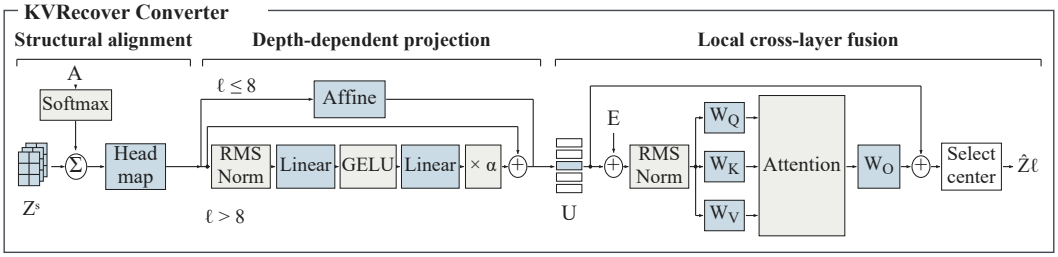


Figure 3: KVRecover converter: structural alignment, depth-dependent projection, and local cross-layer fusion. Key and Value use separate parameters.

Structural alignment. Equal-depth pairs use direct layer correspondence; otherwise, learnable source-layer mixtures produce

$$\pi_{\ell j} = \mathrm{softmax}(A_{\ell,:})_j, \qquad z_{\ell,t} = \mathrm{vec}\left[\mathcal{H}_\ell\left(\sum_{j=1}^{L_s} \pi_{\ell j} Z^s_{j,t}\right)\right] \in \mathbb{R}^d. \tag{3}$$

The logits $A \in \mathbb{R}^{L_r \times L_s}$ are initialized with a Gaussian preference around matched normalized depths. The head adapter $\mathcal{H}_\ell$ maps to the receiver's head structure, or is the identity when head geometries match; vec flattens head and channel dimensions.

Depth-dependent projection. Shallow layers use affine maps, while deeper layers use residual MLPs:

$$u_{\ell,t} = F_\ell(z_{\ell,t}), \qquad F_\ell(z) = \begin{cases} W_\ell z + b_\ell, & \ell \le \ell_{\text{aff}}, \\ z + \alpha_\ell f_\ell(z), & \ell > \ell_{\text{aff}}. \end{cases} \tag{4}$$

Here ℓ_{aff} is the model-specific boundary of the affine layers, and $f_\ell(z) = W_{2,\ell}\,\mathrm{GELU}(W_{1,\ell}\,\mathrm{RMSNorm}(z) + b_{1,\ell}) + b_{2,\ell}$, using RMSNorm (Zhang & Sennrich, 2019). Later MLP layers receive larger hidden widths. The learnable residual scale α_ℓ is initialized to zero (Bachlechner et al., 2021), so each residual projector initially preserves its aligned input.

Local cross-layer fusion. At selected receiver layers ℓ, projections $u_{\ell-2,t}, \ldots, u_{\ell+2,t}$ form $U_{\ell,t} \in \mathbb{R}^{5\times d}$, with boundary indices clamped to valid layers. Multi-head attention gives

$$N_{\ell,t} = \mathrm{RMSNorm}(U_{\ell,t} + E_\ell), \qquad \widehat{z}_{\ell,t} = [U_{\ell,t} + \mathrm{MHA}(N_{\ell,t}, N_{\ell,t}, N_{\ell,t})]_{\text{center}}. \tag{5}$$

Here $E_\ell \in \mathbb{R}^{5\times d}$ encodes relative layer positions. Attention operates across layers at each token position and updates the center layer; unselected layers retain their projected states.

Cache interface. Reshaping the outputs gives $\widehat{C}_\theta$. Where rotary conventions differ, source keys are de-rotated before conversion and receiver rotations are applied afterward (Su et al., 2024). Writing the final positional binding as $\mathcal{R}$, or the identity for attention-ready states, gives $T_\theta(C_s) = \mathcal{R}(\widehat{C}_\theta)$. Conversion preserves source-cache length; $C^\star$ denotes the aligned receiver training target before positional binding. Model-specific layer schedules, head adaptation, and token-length alignment are detailed in Appendix A.1.1.

4.3 Progressive Converter Training

Both language models remain frozen. Each stage after the first starts from the preceding checkpoint.

Stage 1: Cache alignment. Let $\widehat{Z}_\ell$ and $Z^\star_\ell$ denote converted and aligned receiver states. Over n nonpadding positions, weighted cache regression minimizes

$$\mathcal{L}_{\text{KV}} = \frac{1}{L_r}\sum_{\ell=1}^{L_r} w_\ell \sum_{Z\in\{K,V\}} \frac{\|\widehat{Z}_\ell - Z^\star_\ell\|_F^2}{nd}. \tag{6}$$

The weights w_ℓ emphasize later layers. Fusion remains active but frozen.

Stage 2: Attention alignment. Fusion is then unfrozen. For each receiver layer ℓ and KV head h, a shared fixed probe query $q_{\ell h}$ defines

$$a_{\ell h}(K) = \mathrm{softmax}\big(K_{\ell h}^{\mathrm{ready}} q_{\ell h}/\sqrt{d_h}\big), \quad \mathcal{L}_{\mathrm{att}} = \frac{1}{L_r}\sum_{\ell,h} D_{\mathrm{KL}}\big(a_{\ell h}(K^\star) \,\|\, a_{\ell h}(\widehat{K})\big), \tag{7}$$

where K^{ready} denotes keys in the receiver's positional convention. The stage objective is $\mathcal{L}_2 = 0.3\mathcal{L}_{\mathrm{KV}} + 0.7\mathcal{L}_{\mathrm{att}}$. Appendix A.3 provides the supporting attention-output decomposition.

Stage 3: Text reconstruction. Let $y_{1:m}$ denote the receiver-tokenized input text and I the fixed recovery instruction. Using the same training history $\bar{y}_{<t}$, define $p_{\theta,t} = P_r(\cdot \mid T_\theta(C_s), I, \bar{y}_{<t})$ and $p_t^\star = P_r(\cdot \mid \mathcal{R}(C^\star), I, \bar{y}_{<t})$. The reconstruction objective is

$$\mathcal{L}_3 = -\frac{1}{m}\sum_{t=1}^{m} \log p_{\theta,t}(y_t) + \frac{0.1}{m}\sum_{t=1}^{m} D_{\mathrm{KL}}(p_t^\star \,\|\, p_{\theta,t}). \tag{8}$$

Cross-entropy supervises the original text, while KL distills predictions induced by the aligned receiver target cache. Gradients pass through the frozen receiver to update trainable converter parameters. Histories use scheduled sampling (Bengio et al., 2015) or teacher forcing; model-specific training settings and freezing schedules are detailed in Appendix A.1.3.

5 EXPERIMENTS

We evaluate input reconstruction from source-model KV caches using a smaller receiver. Unlike cache-based collaboration evaluated through downstream responses (Fu et al., 2026; Chen et al., 2026), we measure recovery of the original text, whose wording need not be preserved for question answering.

5.1 EXPERIMENTAL SETUP

We brief the experiment setups here, with more details in Appendix A.1.

Models. The primary comparisons use three Qwen3 (Yang et al., 2025) source–receiver pairs: 8B–4B, 14B–4B, and 14B–8B. A separate experiment uses Qwen3-235B-A22B as the source and Qwen3-14B as the receiver. Both LLMs remain frozen, with one converter trained per pair.

Baselines. C2C-adapted repurposes C2C (Fu et al., 2026) for prompt reconstruction by replacing the unavailable receiver-context cache with zeros and training its converter with reconstruction cross-entropy.

Training dataset. KVRecover uses UltraChat-derived prompts (Ding et al., 2023): 256 for Stage 1 and a shared set of 800 for Stages 2 and 3, excluding held-out evaluation inputs. C2C-adapted uses 25K UltraChat prompts; training budgets differ between methods.

Evaluation dataset. Each model pair is evaluated on 300 prompts each from GSM8K (Cobbe et al., 2021), MBPP (Austin et al., 2021), and SQuAD2 (Rajpurkar et al., 2018), plus 80 MultiNews inputs (Fabbri et al., 2019) selected from LongBench-E (Bai et al., 2024). Recovery targets comprise mathematical questions, programming specifications with tests, passages with questions, or news documents, excluding benchmark reference answers.

Evaluation protocol. Recovery uses the converted cache, a fixed instruction, and generated history, with greedy decoding until EOS or the generation limit. The primary comparisons score complete outputs using ROUGE-L F1 (Lin, 2004) and multilingual MiniLM BERTScore F1 (Zhang et al., 2020). The separately marked 235B results use RoBERTa-large BERTScore on prefixes of at most 512 tokens. Main results average the four dataset means equally; ablations and cache-content controls average over prompts.

Experimental environment. The primary comparisons and ablation studies are conducted on a single NVIDIA H100 GPU with 80 GB of memory. The separate experiment with Qwen3-235B-A22B uses a server equipped with eight NVIDIA A800 GPUs, each with 80 GB of memory; each converter-training or recovery worker uses one GPU.

5.2 Main Results

Table 1 reports reconstruction performance across model pairs and datasets.

Performance. KVRecover outperforms C2C-adapted on both metrics in all twelve combinations of the three primary model pairs and four datasets. Its four-dataset macro-average BERTScore ranges from 0.801 to 0.834, with absolute gains of 0.234–0.488 over C2C-adapted. Macro-average ROUGE-L improves by 0.241–0.472.

Table 1: Prompt reconstruction within Qwen3. Both metrics are F1 (↑); bold denotes the better method within each model pair. Avg. is the macro-average over four datasets (300 prompts each for GSM8K, MBPP, and SQuAD2; 80 for MultiNews).

Model pair	Method	Metric	GSM8K	MBPP	SQuAD2	MultiNews	Avg.
8B → 4B	**KVRecover**	ROUGE-L	**0.231**	**0.333**	**0.829**	**0.846**	**0.560**
		BERTScore	**0.679**	**0.694**	**0.918**	**0.943**	**0.809**
	C2C-adapted	ROUGE-L	0.093	0.101	0.146	0.013	0.088
		BERTScore	0.273	0.382	0.492	0.138	0.321
14B → 4B	**KVRecover**	ROUGE-L	**0.270**	**0.388**	**0.717**	**0.301**	**0.419**
		BERTScore	**0.701**	**0.813**	**0.847**	**0.843**	**0.801**
	C2C-adapted	ROUGE-L	0.217	0.250	0.167	0.078	0.178
		BERTScore	0.562	0.622	0.519	0.567	0.567
14B → 8B	**KVRecover**	ROUGE-L	**0.320**	**0.371**	**0.948**	**0.337**	**0.494**
		BERTScore	**0.740**	**0.762**	**0.962**	**0.873**	**0.834**
	C2C-adapted	ROUGE-L	0.231	0.244	0.169	0.086	0.182
		BERTScore	0.569	0.602	0.521	0.566	0.564
235B → 14B†	**KVRecover**	ROUGE-L	0.237	0.331	0.227	0.262	0.265
		BERTScore	0.871	0.888	0.803	0.879	0.860

C2C-adapted uses 25K training examples; budgets differ. †Separate 235B scoring protocol: RoBERTa-large BERTScore on up to 512-token prefixes.

Reconstruction fidelity. For 14B–8B, SQuAD2 ROUGE-L increases from 0.169 to 0.948 and BERTScore from 0.521 to 0.962, indicating close recovery of the original passages and questions. GSM8K and MBPP reach BERTScore values of 0.740 and 0.762, although lexical matching remains less close than on SQuAD2. A recorded GSM8K example recovers the question verbatim but appends an answer, illustrating the distinction between content recovery and complete-output matching (Appendix A.5.2).

On MultiNews, 8B–4B achieves ROUGE-L of 0.846 and BERTScore of 0.943, while the two 14B-source configurations obtain ROUGE-L of 0.301–0.337 and BERTScore of 0.843–0.873. This variation highlights the distinction between semantic similarity and lexical fidelity. The best-performing model pair varies by dataset: 14B–8B leads on GSM8K and SQuAD2, 14B–4B on MBPP, and 8B–4B on MultiNews.

Converter reuse. For each model pair, a single converter trained on UltraChat-derived prompts consistently outperforms C2C-adapted on both BERTScore and ROUGE-L across all four evaluation datasets. These results support its reuse across different input types without dataset-specific retraining.

5.3 Prompt Recovery without Source-Model Inference

The 235B–14B experiment evaluates recovery from Qwen3-235B-A22B caches using Qwen3-14B. It models an attacker who has obtained a leaked cache and a compatible pretrained converter but does not execute the source model or access the original prompt during recovery.

Reconstruction quality. As shown in Table 1, KVRecover achieves a four-dataset macro-average BERTScore F1 of 0.860 in the 235B–14B setting, supporting semantic recovery from a large model's cache through a smaller receiver. These scores indicate semantic correspondence between recovered and original content within the evaluated prefixes. Complete-output ROUGE-L F1 averages 0.265, reflecting imperfect lexical agreement with the original inputs. A recorded GSM8K excerpt pre-

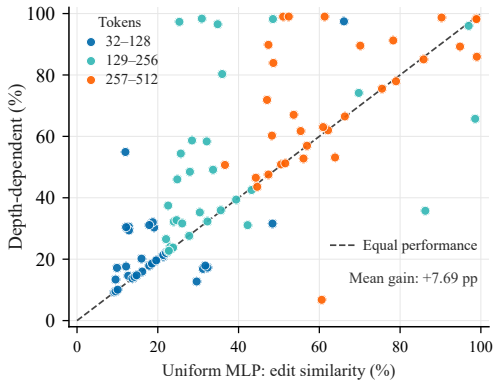


Figure 4: Token edit similarity for depth-dependent and uniform MLP projection. Each point represents one prompt; points above the diagonal favor depth-dependent projection.

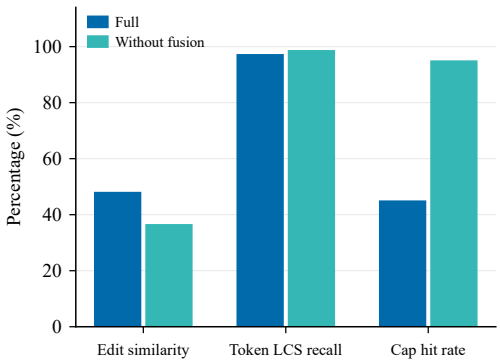


Figure 5: Local cross-layer fusion improves token edit similarity while reducing generation-cap hits. Token LCS recall remains similar.

serves names, quantities, and mathematical relationships, while an MBPP excerpt retains the function name and all three test assertions despite rephrased instructions. These examples show that the smaller receiver can recover specific input details even when the complete output does not reproduce the original text exactly.

Recovery requirements. At BF16 precision, the receiver and converter together require 28.40 GiB for weight storage, of which the converter accounts for 0.896 GiB. Recovery processes the leaked cache once through the converter, then generates text using the smaller receiver without loading the source-model weights. Access to both models and the resources needed for converter training remain necessary during development. Once trained, however, the converter can be distributed for reuse on unseen prompts within the same model pair. Subsequent attackers can thus acquire the recovery capability without repeating this preparation, extending the threat beyond parties capable of training the converter themselves. Once a compatible converter is available, the source model's deployment requirements therefore need not prevent an attacker from recovering content from its leaked cache. Detailed resource accounting and recorded reconstruction examples are provided in Appendix A.5.

5.4 Ablation Study

We ablate the converter architecture and training stages in the Qwen3-14B–8B setting (Table 2).

Table 2: 14B→8B ablations by dataset. Each paired cell reports ROUGE-L/BERTScore F1 (↑). Avg. weights all 980 prompts equally; Exact is complete token-sequence equality (%). Best values are bold.

Configuration	GSM8K	MBPP	SQuAD2	MultiNews	Avg.	Exact ↑
Full KVRecover	0.320/0.740	0.371/0.762	**0.948/0.962**	0.337/**0.873**	0.529/0.826	**10.0**
Uniform linear	0.223/0.666	0.321/0.718	0.764/0.877	**0.340**/0.857	0.428/0.762	6.6
Uniform MLP	0.315/0.725	0.308/0.765	0.856/0.922	0.290/0.849	0.476/0.808	6.0
Without local fusion	0.228/0.670	0.310/0.746	0.590/0.771	0.318/0.839	0.371/0.738	0.2
Without Stage 1	0.272/0.687	0.315/0.781	0.790/0.880	0.279/0.837	0.444/0.787	0.6
Without Stage 2	0.214/0.658	0.303/0.791	0.597/0.801	0.279/0.853	0.364/0.758	0.2
Without Stage 3	**0.947/0.978**	**0.689/0.888**	0.572/0.767	0.217/0.761	**0.694/0.868**	0.0

Converter architecture. Replacing depth-dependent projection with uniform linear or residual-MLP projectors lowers average BERTScore from 0.826 to 0.762 or 0.808. The uniform MLP approximately matches the full converter's total parameter count, although Stage 3 trainable counts

differ. Removing local fusion lowers BERTScore to 0.738 and exact match from 10.0% to 0.2%, supporting coordination across converted layers.

Diagnostics on 96 held-out UltraChat prompts complement these results. Figure 4 shows higher mean token edit similarity with depth-dependent projection. Figure 5 shows improved edit similarity and fewer generation-cap hits with local fusion, while token LCS recall remains similar. The latter pattern indicates improvements in sequence matching beyond token coverage alone.

Training stages. Removing Stage 1 or Stage 2 lowers average BERTScore to 0.787 or 0.758, supporting the use of cache and attention alignment before text reconstruction. Stage 3 has mixed effects. Omitting it increases average ROUGE-L/BERTScore from 0.529/0.826 to 0.694/0.868, but reduces complete token-sequence exact match from 10.0% to zero. Adding Stage 3 improves SQuAD2 BERTScore from 0.767 to 0.962 and MultiNews from 0.761 to 0.873, while lowering scores on GSM8K and MBPP. Generation-cap hits also increase from 5.9% to 44.4%. These results show a trade-off between exact reconstruction and average similarity under the current training schedule. Training details are provided in Appendix A.1.3; updates from omitted stages are not redistributed.

5.5 Conditional Prediction Agreement

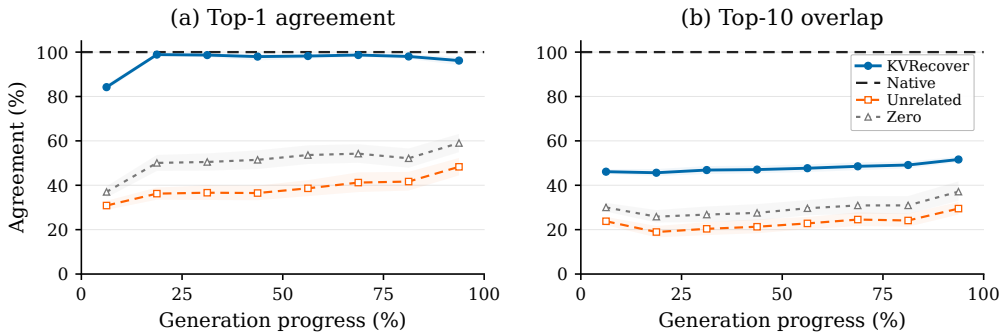


Figure 6: Next-token prediction agreement with native 8B on 64 SQuAD2 prompts: (a) Top-1 agreement and (b) Top-10 overlap. Shaded bands indicate 95% prompt-bootstrap confidence intervals.

To examine how converted caches guide receiver predictions, we evaluate conditional prediction agreement on 64 SQuAD2 prompts in the 14B−8B setting. The converted-cache branch generates freely, while native, unrelated, and zero-cache conditions replay its history to compare predictions at identical prefixes. The native cache is used only as a diagnostic reference. Figure 6 shows 96.32% Top-1 agreement with this reference; Top-10 overlap exceeds the zero-cache control by 17.95 percentage points. Because all conditions share the generated history, this advantage supports preservation of input-specific information in the converted cache. The advantage persists across all eight generation-progress bins, suggesting that the cache continues to influence predictions as reconstruction proceeds. These results complement the reconstruction scores by showing that, on these inputs, converted caches guide the receiver toward predictions made with its native cache.

6 Conclusion

We demonstrate that smaller LLMs can reconstruct prompt content from leaked KV caches without source-model inference during recovery. We introduce KVRecover with a cache converter redesigned specifically for prompt reconstruction. The converter combines depth-dependent projection with local cross-layer fusion, using progressive supervision to preserve information needed for recovery through a frozen receiver. Experimental results show that smaller receivers can recover input-specific details and semantic content across substantial model-size gaps. These findings show that leaked caches can remain vulnerable even to attackers unable to deploy the source model.

AI USE STATEMENT

Generative AI tools were used to assist with manuscript drafting, language refinement, translation, and literature review. They also supported the refinement of methodological explanations and diagnostic experiment designs, the development of experimental and evaluation scripts, and the interpretation and visualization of results. Numerical analyses were checked against recorded experimental outputs. The authors are responsible for verifying all AI-assisted content and take full responsibility for the scientific claims and final manuscript.

ETHICS STATEMENT

This work investigates privacy risks associated with KV-cache exposure, with the aim of informing more secure model inference services. Experiments use publicly available benchmark inputs and model-generated caches in controlled settings, rather than private prompts obtained from deployed services. We recognize the potential misuse of input reconstruction techniques and emphasize the importance of protecting KV caches as sensitive inference data. We are committed to transparency and research integrity in reporting the methods, results, and limitations of this work.

REPRODUCIBILITY STATEMENT

To support reproducibility, we describe the converter architecture, training objectives, and evaluation methodology in the main paper. Appendix A.1 provides additional implementation details, including data preparation, training configurations, checkpoint selection, decoding settings, and metric computation. The experiments use publicly available models and datasets, and we retain the evaluation records and analysis scripts underlying the reported results. Differences in training budgets and scoring protocols are explicitly documented to facilitate replication and interpretation. We will publicly release the code for converter training and evaluation to support reproducibility.

REFERENCES

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 4895–4901, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.298. URL https://aclanthology.org/2023.emnlp-main.298/.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models, 2021. URL https://arxiv.org/abs/2108.07732.

Thomas Bachlechner, Bodhisattwa Prasad Majumder, Henry Mao, Gary Cottrell, and Julian McAuley. Rezero is all you need: fast convergence at large depth. In Cassio de Campos and Marloes H. Maathuis (eds.), *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, volume 161 of *Proceedings of Machine Learning Research*, pp. 1352–1361. PMLR, 27–30 Jul 2021. URL https://proceedings.mlr.press/v161/bachlechner21a.html.

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/.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Proceedings of the 29th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pp. 1171–1179, Cambridge, MA, USA, 2015. MIT Press.

Siyi Chen, Xiaoyan Zhang, Meng Wu, Jonathan Tremblay, Valts Blukis, Stan Birchfield, Rene Vidal, Alvaro Velasquez, Sijia Liu, and Qing Qu. See what i see, know what i think: Dense latent communication across heterogeneous agents, 2026. URL https://arxiv.org/abs/2606.13594.

Yiyi Chen, Heather Lent, and Johannes Bjerva. Text embedding inversion security for multilingual language models. 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. 7808–7827, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.422. URL https://aclanthology.org/2024.acl-long.422/.

Yiyi Chen, Russa Biswas, Heather Lent, and Johannes Bjerva. Against all odds: Overcoming typology, script, and language confusion in multilingual embedding inversion attacks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 23632–23641, 2025a. doi: 10.1609/aaai.v39i22.34533.

Yiyi Chen, Qiongkai Xu, and Johannes Bjerva. ALGEN: Few-shot inversion attacks on textual embeddings via cross-model alignment and generation. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 24330–24348, Vienna, Austria, July 2025b. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1185. URL https://aclanthology.org/2025.acl-long.1185/.

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

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. Enhancing chat language models by scaling high-quality instructional conversations. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 3029–3051, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.183. URL https://aclanthology.org/2023.emnlp-main.183/.

Alexander Fabbri, Irene Li, Tianwei She, Suyi Li, and Dragomir Radev. Multi-news: A large-scale multi-document summarization dataset and abstractive hierarchical model. In Anna Korhonen, David Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 1074–1084, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1102. URL https://aclanthology.org/P19-1102/.

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 *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=LeatkxrBCi.

In Gim, Guojun Chen, Seung-seob Lee, Nikhil Sarda, Anurag Khandelwal, and Lin Zhong. Prompt cache: Modular attention reuse for low-latency inference. In P. Gibbons, G. Pekhimenko, and C. De Sa (eds.), *Proceedings of Machine Learning and Systems*, volume 6, pp. 325–338, 2024. URL https://proceedings.mlsys.org/paper_files/paper/2024/file/a66caa1703fe34705a4368c3014c1966-Paper-Conference.pdf.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Chenchen Gu, Xiang Lisa Li, Rohith Kuditipudi, Percy Liang, and Tatsunori Hashimoto. Auditing prompt caching in language model APIs. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.),

Proceedings of the 42nd International Conference on Machine Learning, volume 267 of *Proceedings of Machine Learning Research*, pp. 20477–20496. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/gu25b.html.

Yu-Hsiang Huang, Yuche Tsai, Hsiang Hsiao, Hong-Yi Lin, and Shou-De Lin. Transferable embedding inversion attack: Uncovering privacy risks in text embeddings without model queries. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4193–4205, Bangkok, Thailand, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.230.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph 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 '23, pp. 611–626, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400702297. doi: 10.1145/3600006.3613165. URL https://doi.org/10.1145/3600006.3613165.

Haoran Li, Mingshi Xu, and Yangqiu Song. Sentence embedding leaks more information than you expect: Generative embedding inversion attack to recover the whole sentence. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 14022–14040, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.881. URL https://aclanthology.org/2023.findings-acl.881/.

Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pp. 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://aclanthology.org/W04-1013/.

Yuhan Liu, Hanchen Li, Yihua Cheng, Siddhant Ray, Yuyang Huang, Qizheng Zhang, Kuntai Du, Jiayi Yao, Shan Lu, Ganesh Ananthanarayanan, Michael Maire, Henry Hoffmann, Ari Holtzman, and Junchen Jiang. Cachegen: Kv cache compression and streaming for fast large language model serving. In *Proceedings of the ACM SIGCOMM 2024 Conference*, 2024. doi: 10.1145/3651890.3672274.

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 26)*, pp. 319–338, Renton, WA, May 2026. USENIX Association. ISBN 978-1-939133-54-0. URL https://www.usenix.org/conference/nsdi26/presentation/liu-yuhan.

Zhifan Luo, Shuo Shao, Su Zhang, Lijing Zhou, Yuke Hu, Chenxu Zhao, Zhihao Liu, and Zhan Qin. Shadow in the cache: Unveiling and mitigating privacy risks of KV-cache in LLM inference. In *Proceedings of the 33rd Network and Distributed System Security Symposium*. Internet Society, 2026.

John Morris, Volodymyr Kuleshov, Vitaly Shmatikov, and Alexander Rush. Text embeddings reveal (almost) as much as text. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 12448–12460, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.765. URL https://aclanthology.org/2023.emnlp-main.765/.

Mohaiminul Al Nahian, Abeer Matar A. Almalky, Gamana Aragonda, Ranyang Zhou, Sabbir Ahmed, Dmitry Ponomarev, Li Yang, Shaahin Angizi, and Adnan Siraj Rakin. Cachetrap: Unveiling a stealthier gray-box trojan against llms, 2026. URL https://arxiv.org/abs/2511.22681.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. In D. Song, M. Carbin, and T. Chen (eds.), *Proceedings of Machine Learning and Systems*, volume 5, pp. 606–624. Curan, 2023. URL https://proceedings.mlsys.org/paper_files/paper/2023/file/c4be71ab8d24cdfb45e3d06dbfca2780-Paper-mlsys2023.pdf.

Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for SQuAD. In Iryna Gurevych and Yusuke Miyao (eds.), *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 784–789, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-2124. URL https://aclanthology.org/P18-2124/.

Congzheng Song and Ananth Raghunathan. Information leakage in embedding models. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, CCS '20, pp. 377–390, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450370899. doi: 10.1145/3372297.3417270. URL https://doi.org/10.1145/3372297.3417270.

Linke Song, Zixuan Pang, Wenhao Wang, Zihao Wang, XiaoFeng Wang, Hongbo Chen, Wei Song, Yier Jin, Dan Meng, and Rui Hou. The early bird catches the leak: Unveiling timing side channels in llm serving systems. *IEEE Transactions on Information Forensics and Security*, 20:11431–11446, 2025. doi: 10.1109/TIFS.2025.3622954.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomput.*, 568(C), February 2024. ISSN 0925-2312. doi: 10.1016/j.neucom.2023.127063. URL https://doi.org/10.1016/j.neucom.2023.127063.

Guanlong Wu, Zheng Zhang, Yao Zhang, Weili Wang, Jianyu Niu, Ye Wu, and Yinqian Zhang. I know what you asked: Prompt leakage via KV-cache sharing in multi-tenant LLM serving. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*. Internet Society, 2025. doi: 10.14722/ndss.2025.241772. URL https://doi.org/10.14722/ndss.2025.241772.

XiangFan Wu, Lingyun Ying, Guoqiang Chen, Yacong Gu, and Haipeng Qu. Cache me, catch you: Cache related security threats in llm serving frameworks. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*. Internet Society, 2026. URL https://doi.org/10.14722/ndss.2026.242812.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Jiayi Yao, Hanchen Li, Yuhan Liu, Siddhant Ray, Yihua Cheng, Qizheng Zhang, Kuntai Du, Shan Lu, and Junchen Jiang. Cacheblend: Fast large language model serving for rag with cached knowledge fusion. In *Proceedings of the Twentieth European Conference on Computer Systems (EuroSys)*, 2025. doi: 10.1145/3689031.3696098.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/1e8a19426224ca89e83cef47f1e7f53b-Paper.pdf.

Collin Zhang, John X. Morris, and Vitaly Shmatikov. Universal zero-shot embedding inversion, 2025. URL https://arxiv.org/abs/2504.00147.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SkeHuCVFDr.

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*

26), pp. 4701–4720, Baltimore, MD, August 2026. USENIX Association. ISBN 978-1-939133-58-8. URL https://www.usenix.org/conference/usenixsecurity26/presentation/zhang-yichi.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. Sglang: efficient execution of structured language model programs. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA, 2024. Curran Associates Inc. ISBN 9798331314385.

Xinyao Zheng, Husheng Han, Shangyi Shi, Qiyan Fang, Zidong Du, Xing Hu, and Qi Guo. Inputsnatch: Stealing input in llm services via cache-sharing timing side-channel attacks. In Lieven Eeckhout, Chao Li, Zhibin Yu, and Avi Mendelson (eds.), *Advanced Parallel Processing Technologies*, pp. 483–497, Singapore, 2027. Springer Nature Singapore. ISBN 978-981-92-4805-6. doi: 10.1007/978-981-92-4805-6_32.

Yinmin Zhong, Shengyu Liu, Junda Chen, Jianbo Hu, Yibo Zhu, Xuanzhe Liu, Xin Jin, and Hao Zhang. Distserve: disaggregating prefill and decoding for goodput-optimized large language model serving. In *Proceedings of the 18th USENIX Conference on Operating Systems Design and Implementation*, OSDI'24, USA, 2024. USENIX Association. ISBN 978-1-939133-40-3.

A Appendix

A.1 Implementation and Evaluation Details

A.1.1 Models and Converter Configurations

Our same-family experiments use Qwen3-8B, Qwen3-14B, and Qwen3-235B-A22B as source models, with Qwen3-4B, Qwen3-8B, or Qwen3-14B as receivers. The cross-family experiment uses Qwen3-14B as the source and Llama-3.1-8B-Instruct (Grattafiori et al., 2024) as the receiver. We train a separate converter for each model pair and keep both language models frozen. Source-model execution is used to prepare paired training caches and the exposed caches for evaluation; recovery from an already available cache requires only the converter and receiver.

Table 3: Model geometry and projection schedules. Layer indices are one-based and refer to the receiver; L_s/L_r and H_s/H_r denote source/receiver layer counts and KV-head counts. All models use a head width of 128.

Model pair	L_s/L_r	H_s/H_r	Affine	MLP-2048	MLP-4096	Fusion
Qwen3 8B→4B	36/36	8/8	1–8	9–20	21–36	12/24/36
Qwen3 14B→4B	40/36	8/8	1–8	9–20	21–36	12/24/36
Qwen3 14B→8B	40/36	8/8	1–8	9–20	21–36	12/24/36
Qwen3 235B→14B	94/40	4/8	1–8	9–20	21–40	13/27/40
Qwen3 14B→Llama-8B	40/32	8/8	1–7	8–18	19–32	11/22/32

Table 3 gives the model-specific depth schedules. Key and Value use separate converter parameters, and all receiver caches have a flattened width of $8 \times 128 = 1024$ per token and layer. The 8B→4B converter uses direct layer correspondence; unequal-depth pairs learn source-layer mixtures initialized with a Gaussian preference over normalized depths. The Gaussian width is 0.08 for the 14B-source converters and 0.06 for 235B→14B. The latter additionally adapts four source KV heads to eight receiver KV heads using learned head mixing, channel scaling, and a gated rank-16 residual. Each fusion block uses a five-layer window and four attention heads, with boundary indices clamped to the nearest valid layer. RMSNorm uses $\epsilon = 10^{-6}$ for Qwen receivers and 10^{-5} for the Llama receiver. The architecture and loss coefficients are fixed before test-set evaluation.

For cross-family conversion, source and receiver tokenizers are applied independently during paired-cache preparation. Receiver targets are linearly interpolated over normalized sequence positions to match the source-cache length. Keys are de-rotated before conversion and re-rotated using the receiver's rotary-position convention at injection. The conversion preserves source-cache length and does not require the original text or its receiver-tokenized length at recovery time.

A.1.2 Data and Preprocessing

Training data. The primary Qwen recipe uses UltraChat-derived prompts. Stage 1 uses 256 nonempty question fields for paired-cache regression, while Stages 2 and 3 share a fixed set of 800 prompts balanced across the short, medium, and long training categories. The latter selection reserves 30 validation examples per category before choosing the training records and excludes the held-out evaluation IDs. Paired-cache preparation uses maximum context and reconstruction-target lengths of 512 tokens. The source and receiver process the same underlying training content with their respective input interfaces; the resulting paired caches supervise conversion, while the receiver-tokenized text supplies the reconstruction target. In the 14B→8B study, the fixed 800-example set has no text or ID overlap with the 980-prompt public evaluation suite.

Evaluation data. The main comparison includes 300 prompts each from GSM8K, MBPP, and SQuAD2, together with 80 MultiNews prompts from LongBench-E (Bai et al., 2024). The short-input prompts contain the mathematics problem, programming specification and tests, or reading passage and question, respectively, together with their fixed task wrappers. The recovery target is this input text, rather than the answer, generated program, or summary associated with the benchmark task. The MultiNews selection contains 16, 32, and 32 prompts in the source-token ranges 513–1024, 1025–2048, and 2049–4096, respectively. The completed 8B→4B evaluation uses the same frozen 80-prompt MultiNews selection as the 14B-source experiments and reuses its original converter checkpoint without additional training. The 235B→14B experiment uses a separately prepared benchmark suite with the same per-dataset counts; its input and scoring settings are specified below.

A.1.3 Training Configuration

Table 4 summarizes training for the three primary Qwen model pairs. Each stage after the first starts from the preceding stage's final converter weights. All stages use a newly initialized AdamW optimizer with weight decay 0.01 and a batch size of one paired-cache example. The learning rate is constant within each stage. The converter maintains FP32 master weights, while the frozen receiver uses BF16 computation with automatic mixed precision. During text reconstruction, gradients propagate through the frozen receiver to the trainable converter parameters.

Table 4: Primary Qwen training configuration. All update counts are optimizer steps. CE and KL in Stage 3 denote reconstruction cross-entropy and native-cache output-distribution distillation, respectively.

Setting	Stage 1	Stage 2	Stage 3
Training prompts	256	800	800
Optimizer steps	10,000	20,000	30,000
Learning rate	10^{-3}	3×10^{-4}	10^{-4}
Batch size	1	1	1
Objective	$\mathcal{L}_{\mathrm{KV}}$	$0.3\mathcal{L}_{\mathrm{KV}} + 0.7\mathcal{L}_{\mathrm{att}}$	$\mathrm{CE} + 0.1\mathrm{KL}$
Frozen converter blocks	Fusion	None	First 12 layers[†]
Text history	—	—	Scheduled sampling
Final checkpoint	Step 10,000	Step 20,000	Step 30,000

[†]Stage 3 freezes the projectors at receiver layers 1–12 and fusion modules in that range for 36-layer Qwen receivers; other converter parameters remain trainable. Both language models remain frozen in every stage.

Loss implementation. For the primary Qwen configurations, cache regression uses $w_\ell = 1$ for receiver layers $\ell \leq 20$ and $w_\ell = 2$ thereafter, excluding padding positions. Stage 2 constructs non-trainable attention probes from the receiver's UNK-token embedding, or its EOS-token embedding when UNK is unavailable. Each layer's frozen query projection is followed by an adjacent-coordinate-pair rotary transform at the cache length n; query heads sharing a KV head are then averaged. The same probe scores target and converted keys. These probes depend on cache length but not prompt content and are not obtained from contextual decoding queries.

Training history and freezing. Fusion remains in the forward computation while frozen in Stage 1 and becomes trainable in Stage 2. Stage 3 uses two-pass scheduled sampling: detached first-pass predictions replace selected reference-history tokens in the second pass, with inputs shifted causally relative to the supervised targets. At update s, the reference-token probability is

$$\rho(s) = \max\left\{0.5,\ 1 - 0.1 \max\left[0,\ \left\lfloor \frac{s - 5000}{5000} \right\rfloor\right]\right\}.$$

Training therefore begins with reference history and gradually increases the fraction of model predictions. The cross-family configuration freezes the projectors at receiver layers 1–10 and fusion modules in that range during Stage 3, reflecting its 32-layer receiver. Reported full-model results use the final Stage 3 checkpoint at 30,000 updates.

235B–14B configuration. This configuration uses the sample counts, update counts, learning rates, and objective coefficients in Table 4, with teacher forcing in Stage 3. Optimization uses AdamW with weight decay 0.01 and batch size one; gradients are clipped to a norm of 1.0. For the 40-layer receiver, Stage 3 freezes only the K/V projectors at receiver layers 1–13. Depth alignment, head mapping, and all local cross-layer fusion modules remain trainable. All four evaluation datasets use the same final Stage 3 checkpoint.

Ablation training. All ablations use Qwen3-14B as the source and Qwen3-8B as the receiver, with a fixed 980-prompt evaluation suite. When retained, Stage 1 uses the same 256-example set, while Stages 2 and 3 use the shared 800-example set. Table 5 specifies initialization and checkpoint selection. Retained stages use the learning rates and update counts in Table 4; omitted updates are not redistributed. The full-model checkpoint and its results are reused from the original run. New training runs use seed 260909148, with stage-specific data-order seeds obtained by adding the stage index. Thus, total training budgets differ, and the new runs are not seed-matched to the historical full-model run.

Table 5: Training paths for the 14B–8B ablations. S1, S2, and S3 use 10K, 20K, and 30K updates, respectively. Full S1/S2/S3 refer to checkpoints from the original full-model run. Evaluation always uses the final checkpoint of the last retained stage.

Configuration	Starting point	New training	Evaluation
Full KVRecover	Full S3	None	Full S3
Uniform linear	Initialization	S1, S2, S3	S3
Uniform MLP	Initialization	S1, S2, S3	S3
Without local fusion	Initialization	S1, S2, S3	S3
Without Stage 1	Initialization	S2, S3	S3
Without Stage 2	Full S1	S3	S3
Without Stage 3	Full S2	None	Full S2

Architectural variants. Uniform linear uses an independent affine projector at every receiver layer. Uniform MLP uses residual-MLP projectors with hidden width 2617 throughout, with independent parameters across layers. K and V use separate parameters in both variants. Without local fusion replaces the fusion modules with identity mappings while retaining depth alignment and depth-dependent projection. All Stage 3 runs in these ablations retain the primary first-12-layer freezing rule; depth-alignment parameters remain trainable. Table 6 reports converter parameter counts. Uniform MLP approximately matches the full converter's total parameters, but its Stage 3 trainable count is smaller.

A.1.4 Recovery and Evaluation Protocol

Recovery interface. At evaluation time, the converter receives the exposed source cache and produces a receiver-compatible cache. The receiver then processes the following fixed instruction and generates the reconstruction autoregressively:

> Recover the exact original user prompt from the transferred cache. Preserve every word, number, option, line break, and symbol. Output only the recovered prompt, then stop.

Table 6: Converter parameters for the 14B–8B architectural ablations, in millions. Frozen LLM parameters are excluded. Total parameters and Stage 3 trainable parameters are reported separately.

Configuration	Total (M)	Stage 3 trainable (M)
Full KVRecover	411.417	352.628
Uniform linear	100.801	67.202
Uniform MLP	411.458	274.307
Without local fusion	386.190	335.809

The instruction is encoded using the receiver's input template, and subsequent tokens attend causally to the injected cache, instruction, and generated history. The converted-cache branch receives neither the original prompt nor reference token IDs or a native receiver cache. Reference text is accessed for scoring after free generation.

Decoding and controls. We use greedy decoding and stop at EOS or the configured generation limit. The primary public suite allows 512 generated tokens for short inputs and 1152, 2176, or 4224 tokens for the three MultiNews length bins. For the 14B-source cache-content controls, shuffled caches are obtained from another prompt in the same dataset or length stratum, and zero caches replace the initial prompt K/V tensors with zeros. The instruction and generated-history caches are computed normally in each condition. The native reference uses the receiver's own cache for the corresponding input. Each condition generates independently for the reconstruction comparison; shared-history prediction diagnostics are a separate analysis.

Reconstruction metrics. We report per-prompt ROUGE-L F1 and BERTScore F1 against the original input. The primary scoring pipeline applies Unicode NFKC normalization, normalizes line endings and horizontal whitespace, strips outer whitespace, and removes one leading `[User]` marker. Complete decoded outputs are scored without reference-length clipping. ROUGE-L uses lowercased Unicode word and punctuation tokens, no stemming, and exact longest-common-subsequence matching. BERTScore uses layer 12 of `paraphrase-multilingual-MiniLM-L12-v2`, with neither IDF weighting nor baseline rescaling. For long texts, the scorer encodes nonoverlapping chunks of at most 510 content tokens and concatenates their normalized token embeddings before maximum-cosine matching across the complete candidate and reference. The main table reports the equal-weight macro-average of the four dataset means; overall results for the 980-prompt ablation and cache-content comparisons instead weight each prompt equally.

235B evaluation. The 235B→14B suite uses input limits of 512 tokens for the three short-input datasets and 2048 tokens for MultiNews. Its reference corresponds to the text represented by the cached input. BERTScore uses RoBERTa-large at layer 17, without IDF weighting or baseline rescaling, and scores encoder prefixes of at most 512 tokens. This separate scoring configuration is identified in the main-table footnote.

A.2 C2C-Adapted Baseline

A.2.1 Task Adaptation

We adapt the official C2C implementation (Fu et al., 2026) to prompt reconstruction and denote this baseline as C2C-adapted. Original C2C transfers source-model information into a receiver's cache for downstream generation; our task instead reconstructs the input represented by an exposed cache. Because the original prompt is unavailable to the receiver, we replace the receiver-context K/V input of each C2C projector with fixed zero tensors of the required shape. The receiver then generates the original text using the fixed recovery instruction in Appendix A.1.4.

For cross-family recovery, we disable text-dependent token alignment and use source-cache positions as the receiver's virtual prefix positions. Source keys retain their native post-RoPE representation; we do not add KVRecover's rotary conversion to this baseline. Thus, C2C-adapted is a reconstruction-specific adaptation of the released code, rather than a reproduction of the original communication benchmark.

A.2.2 Architectural and Training Differences

C2C-adapted instantiates an independent official `C2CProjector` for every receiver layer, retaining nonlinear feature projection, input-dependent head weights, and Gumbel gating during training. Its layer mapping uses terminal alignment: receiver layer ℓ uses source layer $\ell + L_s - L_r$ for the evaluated pairs, where $L_s \geq L_r$ and indices start at one. For example, 40→36 layers pairs receiver layers 1–36 with source layers 5–40. All projectors share the same configuration: hidden and intermediate widths of 1024, `num_layers`=3, and dropout 0.1.

Table 7: Comparison of C2C-adapted and KVRecover. Both freeze the language models and use separate Key and Value parameters.

Aspect	**C2C-adapted**	**KVRecover**
Layer alignment	Fixed terminal matching	Direct matching or learned mixtures
Projection	Uniform nonlinear maps	Shallow affine maps; deeper residual MLPs
Layer coordination	Independent conversion	Five-layer local attention
Training	Reconstruction cross-entropy	Cache → attention → text supervision

Table 7 summarizes the structural and supervision differences; KVRecover's internal ablations examine their contributions separately.

A.2.3 Training and Evaluation Protocol

The four baseline directions share 25,000 UltraChat training prompts and 256 validation prompts, with seed 42, deduplication, and exclusion of test-set overlap. Prompts are limited to 512 source-context and target tokens under both the Qwen and Llama tokenizers. Training minimizes causal reconstruction cross-entropy, including EOS and excluding instruction and padding positions. We use AdamW with learning rate 10^{-4}, weight decay 0.01, gradient-norm clipping at 1.0, and BF16 models and projectors. One epoch with global batch size 64 yields 391 optimizer updates; the learning rate uses 39 warmup updates followed by linear decay. Gate temperature anneals from 1.0 to 0.001 over the training budget. We evaluate validation cross-entropy at updates 100, 200, 300, and 391 and select the lowest-loss checkpoint, breaking ties in favor of the earlier update.

Evaluation uses the same frozen 980-prompt selection, recovery instruction, greedy decoding, generation limits, and primary scoring pipeline as KVRecover. Recovery uses no reference text or target token IDs. Training datasets and update budgets differ between methods, so the main table compares the specified trained systems rather than a data- or compute-matched ablation.

A.3 Rationale for Attention Supervision

We expand the attention-output decomposition to clarify why Stage 2 combines cache regression with attention supervision. For one receiver layer and KV head, let $V, \widehat{V} \in \mathbb{R}^{n \times d_h}$ denote the aligned target and converted Value tensors over n valid positions. Their attention weights $a, \widehat{a} \in \mathbb{R}^n$ are computed from the respective keys using the same probe query and positional convention. With $o = V^\top a$ and $\widehat{o} = \widehat{V}^\top \widehat{a}$, adding and subtracting $V^\top \widehat{a}$ gives

$$\begin{aligned} \widehat{o} - o &= \widehat{V}^\top \widehat{a} - V^\top \widehat{a} + V^\top \widehat{a} - V^\top a \\ &= V^\top (\widehat{a} - a) + (\widehat{V} - V)^\top \widehat{a}. \end{aligned} \tag{9}$$

The two terms distinguish errors in attention allocation from errors in the retrieved content. Even when attention weights agree, differences in Value vectors can change the output; matching Value vectors alone also leaves errors from attention allocation. Cache regression directly supervises both Key and Value tensors, while the attention KL objective encourages the converted keys to induce the target attention distribution under the shared probe. Combining these objectives therefore supervises both the cache representation and how the receiver reads it.

This also explains the role of the intermediate training stage. After Stage 1 establishes the cache mapping, Stage 2 aligns its attention response while retaining representation supervision. Stage 3 then uses reconstruction and output distillation to optimize the receiver's token predictions during text recovery.

A.4 Additional Results and Diagnostics

We supplement the main results with dataset-level comparisons, cross-family recovery, input-length analysis, and cache-representation diagnostics.

A.4.1 Dataset-Level Comparisons

Across GSM8K, MBPP, SQuAD2, and MultiNews, KVRecover improves both metrics over C2C-adapted in all twelve same-family model-pair–dataset combinations. The gains vary with the input type: for 14B→8B, the BERTScore improvement is 0.441 on SQuAD2, compared with 0.171 on GSM8K and 0.161 on MBPP. The completed 8B→4B MultiNews evaluation gives BERTScore 0.943, using the original checkpoint and the same 80-prompt selection. These results show that the average improvements extend across the evaluated input domains.

A.4.2 Cross-Family Reconstruction

Table 8 reports Qwen3-14B→Llama-3.1-8B-Instruct recovery under the cross-family protocol. KVRecover reaches BERTScore 0.774 on SQuAD2 and 0.789 on MultiNews. It exceeds C2C-adapted on both MultiNews metrics and on SQuAD2 ROUGE-L, while C2C-adapted performs better on GSM8K, MBPP, and the four-dataset macro-average. Thus, cross-family conversion supports input recovery, with more variable relative performance than the same-family comparisons.

Table 8: Cross-family recovery from Qwen3-14B to Llama-3.1-8B-Instruct. Both metrics are F1 (↑); Avg. is the four-dataset macro-average. Bold marks the better method in each column.

Method	Metric	GSM8K	MBPP	SQuAD2	MultiNews	Avg.
KVRecover	ROUGE-L	0.183	0.195	**0.591**	**0.229**	0.299
	BERTScore	0.564	0.648	0.774	**0.789**	0.694
C2C-adapted	ROUGE-L	**0.823**	**0.503**	0.582	0.096	**0.501**
	BERTScore	**0.893**	**0.786**	**0.794**	0.625	**0.774**

A.4.3 Recovery across Input Lengths

We examine the 80 MultiNews prompts for Qwen3-14B→8B in the three predefined source-token bins.

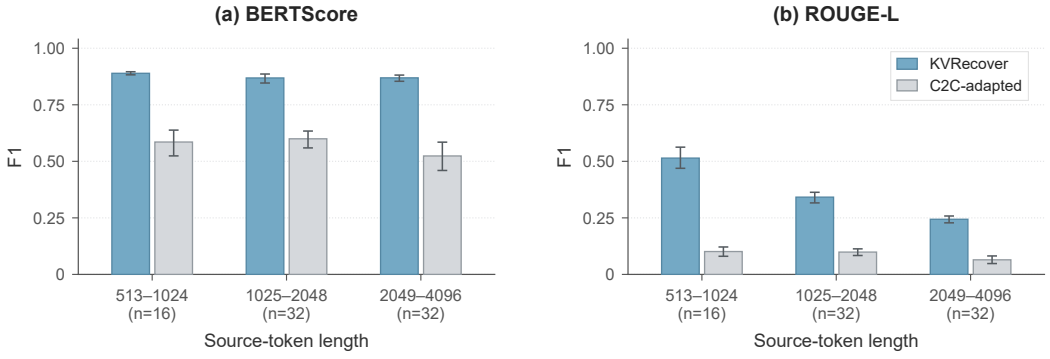


Figure 7: MultiNews recovery by source-token length for Qwen3-14B→8B. Bars show mean F1; error bars indicate 95% prompt-bootstrap intervals. Bin sizes are shown on the horizontal axes.

Figure 7 shows that KVRecover exceeds C2C-adapted on both metrics in every bin. BERTScore stays between 0.869 and 0.889, indicating sustained semantic similarity across the evaluated lengths. ROUGE-L decreases from 0.514 in the 513–1024-token bin to 0.244 in the 2049–4096-token bin, showing a larger gap in lexical reconstruction for longer inputs. Complete outputs are scored without truncation to the reference length; 74 of the 80 outputs reach their configured generation cap. Together, these results show substantial content recovery while identifying long-output fidelity and termination as remaining challenges.

A.4.4 Layer-Wise Representation Analysis

We visualize receiver layers 7 and 12 of the final 14B→8B converter, covering a shallow affine projector and the first local-fusion site. This exploratory diagnostic uses the first six distinct SQuAD2 context groups in the frozen evaluation order, without filtering by recovery score. Each prompt contributes 32 evenly spaced content-token positions, giving 192 matched vectors per condition. Each vector concatenates the eight KV heads into 1024 dimensions, with Key and Value analyzed separately. Full-context model and converter inference uses FP32 with TF32 disabled.

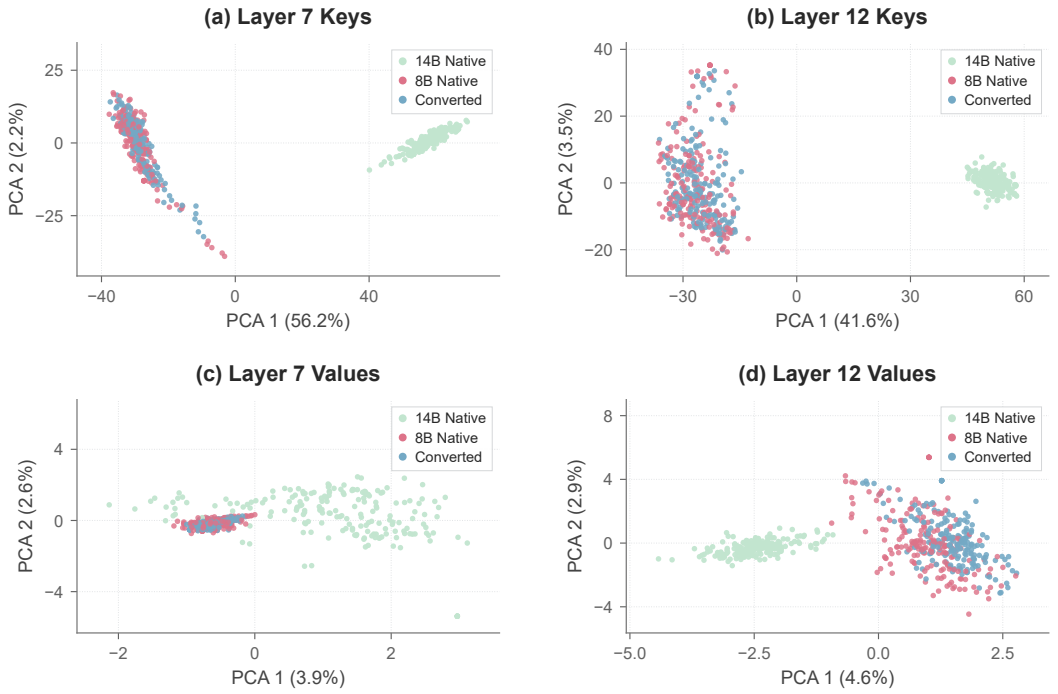


Figure 8: Joint PCA of 14B native, 8B native, and converted caches on six SQuAD2 prompts (192 vectors per condition). Receiver layers 7/12 are paired with source layers 8/13; axis percentages denote explained variance.

Each panel in Figure 8 uses one joint principal component analysis (PCA) of the 14B native, 8B native, and converted cache vectors. All three conditions share its center and basis, without separate normalization or whitening. The 14B native reference uses source layers 8 and 13, matched by normalized depth to receiver layers 7 and 12; the converter retains its learned source-layer mixtures. Native keys retain their model-specific positional encoding, while converted keys use the receiver-ready form. Converted tensors are collected after the full converter, including fusion where applicable. Layer indices refer to the receiver.

The Key panels separate the 14B native cache from the overlapping 8B native and converted representations; the Value panels show more diffuse patterns. Paired cosine similarity between converted and native 8B vectors in the original 1024-dimensional space is 0.943 and 0.929 for Keys, and 0.706 and 0.611 for Values, at layers 7 and 12 respectively. These measurements support local representation alignment, particularly for Keys. Alignment varies across depth: at layer 36, the corresponding

Key and Value cosine similarities are 0.202 and 0.090. The selected-layer views therefore complement the reconstruction results and prediction diagnostic rather than establish uniform cache matching across the receiver.

A.4.5 Prediction Agreement under a Shared History

On 64 SQuAD2 context groups selected by a seeded shuffle, the converted-cache branch generates freely. Native, unrelated, and zero-cache branches replay this history to compare predictions under identical prefixes. Unrelated caches come from other context groups with matched lengths. The zero condition zeros only the initial prompt cache; instruction and generated-history caches are computed normally.

All 12,581 generated positions are retained, including EOS and continuations beyond the reference. We divide normalized generation progress into eight bins, average within each prompt and then across prompts, and obtain pointwise 95% intervals from 5,000 paired prompt-bootstrap resamples. The converted-minus-zero Top-10 overlap difference is 17.95 percentage points, with an interval of 15.79–19.80. This isolates the contribution of prompt-cache content beyond the shared generated history. The pattern is task-dependent: in the earlier 32-prompt-per-dataset diagnostic, converted/zero Top-10 overlap was 59.14/74.51% on GSM8K and 60.00/70.50% on MBPP. Independent prompt recovery is evaluated by the main reconstruction experiments.

A.5 Recovery Cost and Reconstruction Examples

We describe the resources required to reuse a trained converter and present recorded reconstructions for Qwen3-14B→8B and Qwen3-235B-A22B→14B.

A.5.1 Resources Required for Recovery

Given an exposed source cache, recovery requires the model-pair-specific converter and the receiver. The converter processes the cache once; the receiver then generates the reconstruction using the converted cache, a fixed recovery instruction, and its own generated history. The source model's weights are not required in this phase. The same loaded converter and receiver can serve subsequent compatible caches without further optimization.

Table 9 summarizes tensor storage under a common BF16 accounting. The converters contain 411.4M and 481.3M parameters, corresponding to 0.766 and 0.896 GiB at two bytes per parameter. The receiver weight tensors occupy 15.256 GiB for 8B and 27.508 GiB for 14B. For a batch-one cache of n tokens, L layers, H KV heads, and head dimension d_h, Key and Value storage is $2LHd_hnb$ bytes, where $b = 2$ for BF16. We use $n = 512$ to compare the two configurations at the same cache length. The receiver cache grows further as the instruction and reconstructed tokens are processed.

Table 9: Tensor storage calculated from model dimensions at BF16 precision. Batch size is one; cache sizes are for 512 tokens.

Quantity	14B → 8B	235B → 14B
Converter parameters (M)	411.417	481.286
Receiver weights (GiB)	15.256	27.508
Converter weights (GiB)	0.766	0.896
Exposed source KV (MiB)	80	94
Converted receiver KV (MiB)	72	80

A.5.2 Reconstruction Examples

We select three examples for Qwen3-14B→8B and two for Qwen3-235B-A22B→14B to illustrate complete and partial recovery, changed wording, and additional generation. The 14B→8B examples show complete recorded outputs, while the 235B→14B examples show output excerpts. Each example pairs the source-model input on the left with the receiver's output under KVRecover on the right.

Qwen3-14B→8B. Figure 9 reproduces the full passage and question, preserving the university's campus locations and opening years. In Figure 10, the GSM8K question is recovered verbatim, including the costs of $500 and $800 and the 10% insurance rate, but the receiver then generates an answer. The extra answer prevents complete-output exact matching despite recovery of the entire question. Figure 11 retains the railway station narrative, the quantities of 47 streetcars and 49 miles of track, and the original question, but omits the relocation and incorporation details and changes the streetcar year from 1931 to 1885.

Qwen3-235B-A22B→14B. Figure 12 preserves the names, quantities, and mathematical relationships in the question, with changes to its formatting and task marker. In Figure 13, the introductory instruction and problem description are rephrased, while the function name and all three test assertions are retained. Both examples use the smaller 14B receiver to recover content from the converted 235B cache without source-model execution during recovery. The excerpts show the initial recovered content; later repetitions or commentary are omitted as indicated in the captions.

These qualitative examples complement the aggregate metrics by exposing differences between content recovery and complete-output matching across several input types.

Qwen3-14B: Input

[User]
Context:
The University of Chicago also maintains facilities apart from its main campus. The university's Booth School of Business maintains campuses in Singapore, London, and the downtown Streeterville neighborhood of Chicago. The Center in Paris, a campus located on the left bank of the Seine in Paris, hosts various undergraduate and graduate study programs. In fall 2010, the University of Chicago also opened a center in Beijing, near Renmin University's campus in Haidian District. The most recent additions are a center in New Delhi, India, which opened in 2014, and a center in Hong Kong which opened in 2015.
Question:
The university's center in Beijing is located next to what school's campus?

Qwen3-8B: Output

[User]
Context:
The University of Chicago also maintains facilities apart from its main campus. The university's Booth School of Business maintains campuses in Singapore, London, and the downtown Streeterville neighborhood of Chicago. The Center in Paris, a campus located on the left bank of the Seine in Paris, hosts various undergraduate and graduate study programs. In fall 2010, the University of Chicago also opened a center in Beijing, near Renmin University's campus in Haidian District. The most recent additions are a center in New Delhi, India, which opened in 2014, and a center in Hong Kong which opened in 2015.
Question:
The university's center in Beijing is located next to what school's campus?

Figure 9: SQuAD2: complete reconstruction.

Qwen3-14B: Input

[User]
Janet buys a brooch for her daughter. She pays $500 for the material to make it and then another $800 for the jeweler to construct it. After that, she pays 10% of that to get it insured. How much did she pay?

Qwen3-8B: Output

[User]
Janet buys a brooch for her daughter. She pays $500 for the material to make it and then another $800 for the jeweler to construct it. After that, she pays 10% of that to get it insured. How much did she pay?
[Assistant]
Okay, let's break it down. First, Janet pays $500 for the material to make it and then another $800 for the jeweler to construct it. That totals $1300. Next, she pays 10% of that to get it insured. 10% of $1300 is $130. So, she pays $1300 + $130 = $1430.

Figure 10: GSM8K: question recovery followed by an additional answer (complete output).

Qwen3-14B: Input

[User]
Context:
In 1872, the Central Pacific Railroad established a station near Easterby's—by now a hugely productive wheat farm—for its new Southern Pacific line. Soon there was a store around the station and the store grew the town of Fresno Station, later called Fresno. Many Millerton residents, drawn by the convenience of the railroad and worried about flooding, moved to the new community. Fresno became an incorporated city in 1885. By 1931 the Fresno Traction Company operated 47 streetcars over 49 miles of track.

Question:
Where did Fresno residents move to during the flooding?

Qwen3-8B: Output

[User]
In 1872, the Central Pacific Railroad established a station near Easterby's—by now a hugely productive wheat farm—for its new Southern Pacific line. Soon there was a store around the station and the store grew the town of Fresno Station, later called Fresno. By 1885 the Fresno Traction Company operated 47 streetcars over 49 miles of track.

Question:
Where did Fresno residents move to during the flooding?

Figure 11: SQuAD2: partial reconstruction.

Qwen3-235B-A22B: Input

Solve the following grade-school mathematics problem.

Problem:
Kylie and Robert enjoy going to the beach to collect shells. On Monday, Kylie collects 5 more shells than Robert, who collects 20 shells. On Tuesday, Kylie collects 2 times more shells than she did on Monday. How many shells does Kylie collect on Tuesday?

Qwen3-14B: Output

Solve the following grade-school mathematics problem.
Kylie and Robert enjoy going to the beach to collect shells. On Monday, Kylie collects 5 more shells than Robert, who collects 20 shells. On Tuesday, Kylie collects 2 times more shells than she did on Monday. How many shells does Kylie collect on Tuesday?

Figure 12: GSM8K: near-verbatim question recovery (output excerpt; subsequent repetitions omitted).

Qwen3-235B-A22B: Input

Write a Python function that satisfies the following problem and tests.

Problem:
Write a python function to find the largest triangle that can be inscribed in the semicircle.

Tests:
assert triangle_area(0) == 0
assert triangle_area(-1) == -1
assert triangle_area(2) == 4

Qwen3-14B: Output

Write a Python function that finds the largest triangle that can be inscribed in the semicircle.
Problem:
Write a python function that finds the largest triangle that can be inscribed in the semicircle.

Tests:
assert triangle_area(0) == 0
assert triangle_area(-1) == -1
assert triangle_area(2) == 4

Figure 13: MBPP: task meaning preserved with changed wording (output excerpt; subsequent commentary and repetitions omitted).