CACHECODEC: TASK-ORIENTED KV COMPRESSION FOR BANDWIDTH-CONSTRAINED MULTI-LLM COMMUNICATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models are increasingly deployed in collaborative systems, where exchanging key-value (KV) states can bypass autoregressive intermediate-text generation and preserve rich model-internal context. However, KV caches are high-dimensional, model-specific states designed for local inference rather than communication, making direct transmission prohibitively expensive over bandwidth-constrained links. We therefore formulate KV compression as task-oriented communication, preserving downstream-relevant information rather than faithfully reconstructing the transmitted KV states. Based on this perspective, we propose CACHECODEC, a task-oriented codec for efficient inter-LLM KV communication. CACHECODEC first maps high-dimensional KV states into a compact representation message optimized under downstream task supervision, and then applies frequency-domain coding along the sequence dimension to convert this representation into a low-rate bitstream for transmission. Across heterogeneous LLM pairs and downstream tasks, CACHECODEC reduces transmitted payload by $792\times$ while maintaining or improving task accuracy, and achieves $8.64\times$ reduction in end-to-end latency over a 50 Mbps communication link.

1 INTRODUCTION

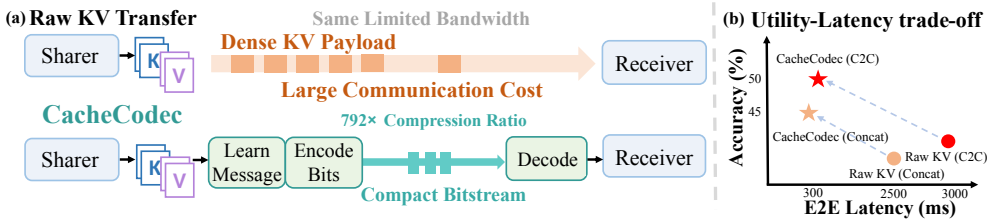


Figure 1: Overview of CACHECODEC results across heterogeneous Sharer–Receiver pairs and cache-integration mechanisms. CACHECODEC reduces communication cost while preserving downstream task utility.

Large language models are increasingly used as collaborative agents for reasoning, role-specialized task execution, and embodied decision making (Hong et al., 2024; Wang et al., 2025; Wu et al., 2023; Du et al., 2023; Zhang et al., 2024). These systems rely on communication to exchange task-relevant information across agents, making the communication interface a key component of multi-LLM collaboration.

We consider communication from a *Sharer*, which provides task-relevant information, to a *Receiver*, which uses it for downstream inference. Most existing multi-agent systems communicate through natural language: a *Sharer* sends a text message, and a *Receiver* tokenizes it as part of its input. This interface is broadly compatible across models, but requires text generation at the Sharer and additional context processing at the Receiver (Fu et al., 2026; Zheng et al., 2026). Recent work has therefore explored more direct communication through model-internal representations, particularly

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.

key-value (KV) cache states (Fu et al., 2026; Dery et al., 2026; Rossi et al., 2026). By reusing already-computed states, cache-level communication can avoid intermediate text generation and repeated context processing. However, directly transmitting KV caches introduces a new challenge: these KV states can be prohibitively expensive to communicate, especially when collaborating models are deployed across separate servers or edge–cloud nodes.

Raw KV caches are not designed as communication representations. They are high-dimensional, continuous, and model-specific representations optimized for the Sharer's own inference. The Receiver, however, may not require all information encoded in these states for downstream prediction. Directly transmitting or faithfully reconstructing the full Sharer cache can therefore spend substantial communication resources on information with limited receiver-side utility. This suggests a different objective for cache-level communication: rather than preserving the Sharer's internal state itself, we seek a compact representation that preserves the information useful to the Receiver.

Existing work addresses complementary parts of this problem. Cross-model cache communication methods learn integration, translation, or fusion mechanisms that make representations from one model usable by another (Fu et al., 2026; Dery et al., 2026; Rossi et al., 2026; Lee et al., 2026). These methods establish the feasibility of communicating through internal states, but primarily focus on cache compatibility rather than minimizing the communication payload. In contrast, KV-cache compression methods reduce memory or serving overhead within a single model through token selection, quantization, or low-rank approximation (Liu et al., 2024b; Hooper et al., 2024; Chang et al., 2025; Liu et al., 2024a). Their goal is typically to preserve the inference behavior of the model that originally produced the cache, rather than determine which information should be retained for a different Receiver. Neither direction therefore directly addresses how to construct and encode a compact message whose content is determined by receiver-side task utility.

To address this problem, we introduce CACHECODEC, a task-oriented codec for bandwidth-efficient communication between heterogeneous LLMs. CACHECODEC separates two complementary problems: *what to communicate* and *how to communicate it*. To determine *what to communicate*, a Latent Channel Compressor (LCC) jointly maps the Sharer's high-dimensional K/V states into a compact representation message. Rather than minimizing reconstruction error with respect to the original KV cache, the latent bottleneck is optimized through the frozen receiving model using downstream task supervision, encouraging its limited capacity to retain information that contributes to Receiver-side prediction. To determine *how to communicate* this representation, a lightweight Frequency Codec transforms the continuous representation along the sequence dimension, quantizes the resulting coefficients, and entropy-codes them into an actual transmittable bitstream. At the Receiver, the bitstream is decoded and mapped into KV states required by the downstream cache-integration mechanism. CACHECODEC therefore reduces communication cost without modifying the underlying LLMs or their existing cache-integration interfaces.

We evaluate CACHECODEC across heterogeneous Sharer–Receiver pairs, multiple downstream reasoning benchmarks, and both concatenation- and fusion-based cache integration. Across these settings, the learned compact representations preserve or improve average downstream task accuracy while substantially reducing the amount of information transmitted. These results suggest that effective cross-model collaboration does not require transmitting the Sharer's full internal state; substantially more compact, task-oriented representations can be sufficient for the Receiver. Our contributions are threefold :

- We formulate inter-LLM KV compression as a *task-oriented communication problem*, where the objective is to preserve Receiver-side utility under a compact communication bottleneck rather than faithfully reconstruct the Sharer's internal KV states.
- We propose CACHECODEC, which separates task-oriented representation formation from bitstream coding: LCC learns a compact Receiver-supervised representations, while the Frequency Codec converts it into a low-rate transmittable bitstream.
- We evaluate CACHECODEC using actual transmitted payload and end-to-end latency across heterogeneous LLM pairs and multiple cache-integration mechanisms, demonstrating an average $792\times$ payload reduction and an $8.64\times$ latency reduction under a 50 Mbps link while maintaining comparable or better average downstream task accuracy.

2 Related Work

2.1 Communication in Multi-LLM Systems

Multi-LLM systems traditionally communicate through natural-language messages, requiring autoregressive generation at the Sharer and repeated processing at the Receiver (Zheng et al., 2026). Recent work instead exchanges model-internal representations. Cross-model cache methods learn integration or translation functions that make KV states produced by one model usable by another (Fu et al., 2026; Dery et al., 2026; Lee et al., 2026). Latent Cache Flow (LCF) further represents K/V information through a compact shared latent channel (Rossi et al., 2026), while Q-KVComm directly studies KV compression for inter-agent communication (Kriuk & Ng, 2026). Related systems such as PrefillShare and KVCOMM focus on cache reuse across models or contexts (Woo et al., 2026; Ye et al., 2026).

These works establish internal model states and mainly address representation alignment or reuse, with less emphasis on Receiver-supervised representations and transmitted bitstreams. Our approach directly targets this communication bottleneck.

2.2 KV Cache Compression and Coding

Existing KV compression methods exploit redundancy along several dimensions. Token-level approaches reduce cache length by retaining attention-critical positions (Zhang et al., 2023; Liu et al., 2023; Li et al., 2024; Xiao et al., 2024), but remove complete token states and primarily target memory-efficient single-model inference. Hidden-space and precision methods instead reduce feature or numerical redundancy through low-rank projection, low-bit quantization, mixed precision, and residual correction (Zhang et al., 2025; Liu et al., 2024b; Hooper et al., 2024; Li et al., 2025; Kang et al., 2024; He et al., 2024). These methods demonstrate substantial compressibility of KV representations, but generally optimize local storage or inference behavior rather than Receiver-side utility in cross-model communication.

Frequency-domain methods such as FreqKV and Fourier Compressor exploit spectral structure for KV compression (Kai et al., 2026; Wang et al., 2026), but mainly target efficient single-model cache representation rather than transmitted bitstreams. CacheCodec instead applies transform coding to a Receiver-supervised latent message, following classical transform-coding principles (Ahmed et al., 1974; Wallace, 1991) for bandwidth-constrained inter-LLM communication.

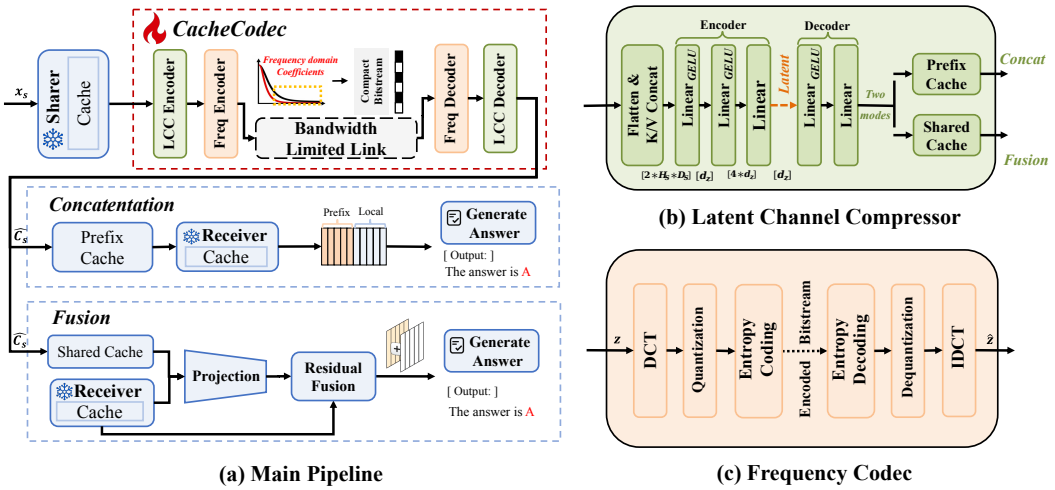
3 Method

3.1 CacheCodec Overview

We consider cache-level communication from a Sharer $\mathcal{M}_s$ to a Receiver $\mathcal{M}_r$, which may differ in architecture, depth, and hidden dimensions. Given a context of S tokens, the Sharer produces layer-wise key and value states $\mathbf{K}_l^s, \mathbf{V}_l^s \in \mathbb{R}^{B\times H_s\times S\times D_s}$, where B is the batch size, H_s is the number of Sharer KV heads, and D_s is the dimension of each head. The cache states from the L transmitted layers are denoted as $\mathcal{C}_s = \{(\mathbf{K}_l^s, \mathbf{V}_l^s)\}_{l=1}^{L}$. These KV states are high-dimensional representations optimized for the Sharer rather than for communication, and a heterogeneous Receiver may not require all of their information. CacheCodec therefore learns a task-oriented compact representations that preserves downstream-relevant information instead of reconstructing the original cache with high numerical fidelity. The communication bottleneck limits message capacity, while Receiver-side supervision determines how this capacity is allocated.

Figure 2 illustrates the overall pipeline. On the Sharer side, the Latent Channel Compressor (LCC) jointly maps each layer of K/V states to a shared latent representation. LCC determines *what to communicate*: it retains information useful for downstream prediction. Then the Frequency Codec determines *how to communicate it*: it encodes the latent representation into a compact bitstream b. At the Receiver, the inverse codec reconstructs latent and maps it to $\widehat{\mathbf{K}}_l^{s\to r}$ and $\widehat{\mathbf{V}}_l^{s\to r}$, whose dimensions are determined by the downstream cache-integration mechanism.

Receiver-side integration. Let $\mathcal{A}$ denote the Receiver-side cache-integration operator. Under virtual-prefix concatenation, the decoded states are appended as an additional prefix during pre-



(a) Main Pipeline (b) Latent Channel Compressor (c) Frequency Codec

Figure 2: Overview of CACHECODEC. The Latent Channel Compressor jointly maps Sharer K/V states into a compact task-oriented latent representation, which the Frequency Codec encodes into a low-rate bitstream. After transmission and decoding, K/V-specific mappings produce cache states for Receiver-side integration.

fill. Under layer-wise KV fusion, as in C2C (Fu et al., 2026), they are fused with the Receiver's native cache at the corresponding layers. CACHECODEC preserves the original integration interface and requires no modification to $\mathcal{A}$.

Given a Sharer context x_s, Receiver input x_r, the reconstructed Sharer cache at the Receiver ($\hat{\mathbf{C}}_s$) and target y, LLMs remain frozen. CACHECODEC parameters θ are optimized solely through the Receiver-side task loss:

$$\min_{\theta} \mathcal{L}_{\text{task}} = -\frac{1}{T}\sum_{t=1}^{T} \log p_{\text{R}}\left(y_t \mid x_r,\, y_{<t},\, \hat{\mathbf{C}}_s(\theta)\right), \tag{1}$$

Thus, the communication bottleneck determines the available message capacity, while task supervision determines how that capacity is used.

Communication operating points. CACHECODEC provides controls at both the representation and coding levels. d_z determines the dimensionality of the communicated representation and therefore affects both representation capacity and communication cost. The quantization configuration controls the precision of the transformed coefficients and introduces the lossy rate–distortion trade-off. Entropy coding is lossless: it changes the realized payload size and coding overhead without altering the reconstructed quantized symbols. Unless otherwise stated, we use $d_z = 128$ with a fixed quantization configuration and report the actual serialized payload produced by the entropy coder.

3.2 LCC: LATENT CHANNEL COMPRESSOR

Observation: low-dimensional head-channel structure. We first transpose each cache tensor and flatten the head and within-head dimensions: $\mathbf{X}_l^K, \mathbf{X}_l^V \in \mathbb{R}^{B\times S\times H_s\times D_s}$. We center the representations, merge the batch and sequence axes, and perform singular value decomposition (SVD). The cumulative spectral energy retained by the leading r singular directions is $E(r) = \frac{\sum_{i=1}^{r}\sigma_i^2}{\sum_{i=1}^{C_s}\sigma_i^2}$. Figure 3 reveals substantial redundancy across the head-channel dimensions, motivating compression of this space before transmission. DeepSeek-V2 and LCF (dee, 2024; Rossi et al., 2026) show that shared latent representations can effectively reduce such representational redundancy. Inspired by this idea, LCC jointly encodes K and V into a compact shared latent space optimized by downstream task supervision.

Task-oriented joint K/V mapping. For each transmitted layer, we concatenate the flattened K and V representations along the channel dimension and apply a nonlinear projection to produce a shared

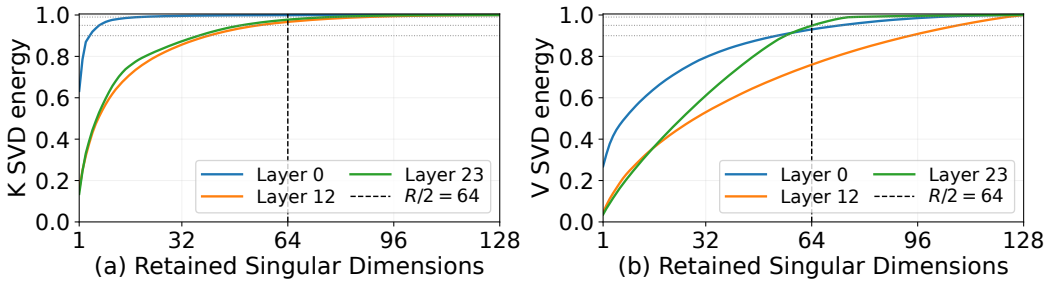


Figure 3: Cumulative SVD energy of K/V head-channel representations across representative layers of Qwen2.5-0.5B-Instruct on OpenHermes2.5 (Teknium, 2023) (sequence length: 128).

representation:

$$\mathbf{Z}_l = f_\theta(\mathbf{X}_l^{KV}), \qquad \mathbf{Z}_l \in \mathbb{R}^{B\times S\times d_z}, \tag{2}$$

where $d_z \ll 2H_sD_s$.

LCC is not trained to reconstruct the original Sharer K/V tensors. Instead, Receiver-side task gradients optimize f_θ through the communication path, encouraging it to retain information useful for downstream prediction. LCC compresses only the head-channel dimension while preserving all S token positions. We denote the latent representations from all transmitted layers as $\mathcal{Z} = \{\mathbf{Z}_l\}_{l=1}^{L}$.

Receiver-side K/V reconstruction. After frequency decoding, the Receiver obtains the recovered representations $\widehat{\mathbf{Z}}_l$. Two K/V-specific output mappings convert it into key and value states:

$$\widehat{\mathbf{X}}_l^{K\to r} = g_K(\widehat{\mathbf{Z}}_l), \qquad \widehat{\mathbf{X}}_l^{V\to r} = g_V(\widehat{\mathbf{Z}}_l). \tag{3}$$

For virtual-prefix concatenation, the mappings produce the Receiver cache width $C_r = H_rD_r$, so the decoded states can be directly consumed as Receiver KV states. For layer-wise fusion, they reconstruct the Sharer-side cache geometry expected by the original fusion module. CACHECODEC therefore adapts the decoded representation to the required interface without modifying the downstream integration mechanism.

3.3 FREQUENCY CODEC

LCC substantially reduces the channel dimensionality of the Sharer cache, but its output $\mathbf{Z}_l \in \mathbb{R}^{B\times S\times d_z}$ remains a dense continuous-valued tensor. Transmitting this tensor directly in floating-point form would still incur considerable communication cost.

Observation: non-uniform sequence-frequency structure. We first examine the frequency distribution of the LCC representation along the sequence dimension. As shown in Figure 4, its spectral energy is concentrated in a relatively small portion of the frequency spectrum. This structure suggests that different frequency components need not be represented with equal precision. We further perform a diagnostic experiment that retains progressively larger low-frequency portions of the DCT spectrum. As shown in Figure 5, retaining only the lowest-frequency 10% of coefficients yields 41.8% accuracy. Increasing this proportion to 25% raises accuracy to 45.6%, close to the 46.8% obtained with the full spectrum. In this setting, most task-relevant information is therefore preserved within the lower-frequency portion, motivating finer quantization for low frequencies.

Sequence-domain DCT. For each transmitted layer l, we apply a one-dimensional DCT-II (Ahmed et al., 1974) independently to each latent channel along the sequence dimension:

$$\begin{aligned} C_{l,n,k,j} &= \alpha_k \sum_{s=0}^{S-1} Z_{l,n,s,j} \cos\left[\frac{\pi}{S}\left(s+\frac{1}{2}\right)k\right], \\ \alpha_k &= \begin{cases} \sqrt{1/S}, & k=0, \\ \sqrt{2/S}, & 1\le k<S. \end{cases} \end{aligned} \tag{4}$$

Here, n indexes the batch element, s the token position, k the frequency index, and j the latent channel. The transformed tensor has the same dimensionality as the input latent. The DCT is

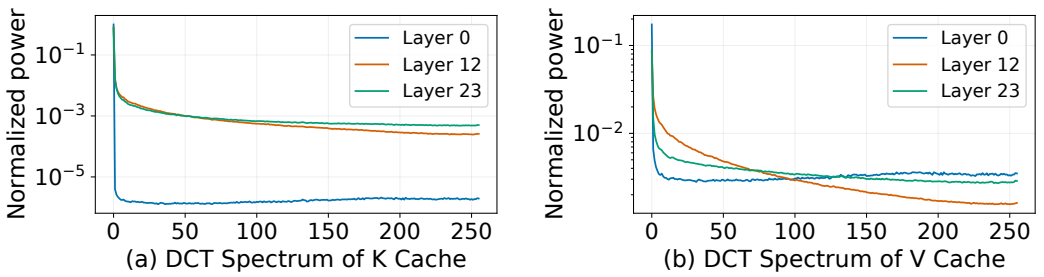


Figure 4: Sequence-domain spectrum of the LCC representations across cache groups in OpenHermes2.5 dataset Teknium (2023), Sequence length = 256.

information preserving and does not directly reduce the payload. It reorganizes sequence variation into frequency-ordered coefficients for subsequent quantization and entropy coding.

Frequency-dependent quantization. The DCT coefficients remain continuous-valued and must be discretized before entropy coding. The quantization step controls the rate–distortion trade-off: larger steps reduce symbol entropy but increase reconstruction error, whereas smaller steps preserve more precision at a higher bit cost. Motivated by the greater downstream importance of low-frequency components, we use a monotonic quadratic schedule that assigns finer quantization to lower frequencies. For frequency index i, we compute

$$q_{l,n,i,j} = \mathrm{round}\left(\frac{C_{l,n,i,j}}{\Delta_i}\right), \qquad \Delta_i = \beta\left[\mathrm{low} + (\mathrm{high} - \mathrm{low})\left(\frac{i}{S-1}\right)^2\right], \tag{5}$$

where low and high specify the relative quantization levels at the two ends of the spectrum, and β controls the overall scale. During training, we implement hard quantization with a straight-through estimator:

$$\tilde{q} = x + \mathrm{sg}(\mathrm{round}(x) - x), \qquad x = C/\Delta, \tag{6}$$

where $\mathrm{sg}(\cdot)$ denotes stop-gradient. This expression is identical to hard rounding in the forward pass but uses the identity gradient in the backward pass. Consequently, downstream task gradients can propagate through the codec to the LCC encoder. Inference uses the same hard-rounded symbols as those transmitted by the actual codec.

Entropy coding and inverse decoding. Quantization produces discrete symbols but does not directly define the transmitted payload. We serialize the quantized symbols and apply lossless entropy coding to obtain the final bitstream b. The entropy coder exploits the empirical distribution of the quantized symbols to reduce the serialized size while preserving them exactly.

At the Receiver, the bitstream is entropy-decoded, dequantized, and inverse-transformed to reconstruct $\widehat{\mathbf{Z}}_l$. The K/V-specific mappings in Section 3.2 then convert the reconstructed representations into the cache states required for Receiver-side integration.

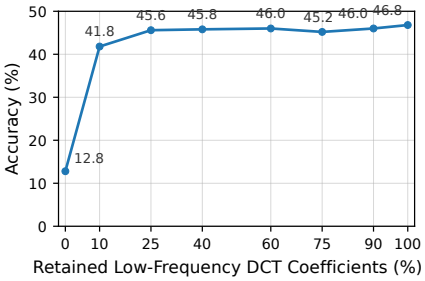


Figure 5: Downstream accuracy when progressively retaining low-frequency DCT coefficients.

LCC and the Frequency Codec therefore address complementary sources of communication cost. LCC reduces the dimensionality of the communicated representation, while the Frequency Codec converts this compact representation into a low-rate discrete bitstream for transmission.

Sharer	Task	Metric	Receiver	Sharer	T2T	Raw		CACHECODEC	
						Concat	C2C	Concat	C2C
Qwen2.5-0.5B	MMLU-Redux	Acc↑	35.2	38.4	38.5	40.3	42.6	39.6	**42.9**
		T_{E2E} ↓	143	117	334	614	494	**207**	277
	OpenBookQA	Acc↑	39.8	45.6	45.8	46.8	**52.8**	47.8	52.0
		T_{E2E} ↓	141	117	313	482	397	**188**	260
	ARC-C	Acc↑	40.5	41.4	41.5	47.7	52.4	48.1	**54.8**
		T_{E2E} ↓	148	131	360	560	438	**206**	277
	C-Eval	Acc↑	30.2	40.4	40.0	36.6	39.7	38.1	**43.0**
		T_{E2E} ↓	135	119	311	581	463	**199**	256
Llama-3.2-1B	MMLU-Redux	Acc↑	35.2	26.5	33.3	38.4	44.4	41.3	**44.8**
		T_{E2E} ↓	143	39	160	1945	1274	239	**175**
	OpenBookQA	Acc↑	39.8	30.6	42.2	43.0	46.8	46.4	**49.6**
		T_{E2E} ↓	141	37	151	1461	975	230	**164**
	ARC-C	Acc↑	40.5	23.8	36.9	41.3	52.0	44.5	**55.3**
		T_{E2E} ↓	148	38	157	1689	1114	253	**164**
	C-Eval	Acc↑	30.2	29.9	28.9	30.7	40.4	34.5	**41.9**
		T_{E2E} ↓	135	29	132	1965	1200	228	**156**
Qwen3-4B-Base	MMLU-Redux	Acc↑	35.2	0.9	35.0	34.8	41.0	38.6	**41.2**
		T_{E2E} ↓	143	292	1240	3359	4313	**222**	233
	OpenBookQA	Acc↑	39.8	2.4	39.2	39.4	48.6	42.4	**51.0**
		T_{E2E} ↓	141	296	1192	2377	3030	205	**203**
	ARC-C	Acc↑	40.5	1.5	41.0	38.0	47.9	42.2	**53.0**
		T_{E2E} ↓	148	297	1241	2807	3621	**227**	238
	C-Eval	Acc↑	30.2	2.3	31.4	30.9	40.1	33.8	**40.4**
		T_{E2E} ↓	135	304	1294	3137	4040	206	**195**

Table 1: Accuracy (%) and end-to-end latency (ms) of different collaboration methods across four benchmarks under a 50 Mbps bandwidth setting.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Model pairs and Collaboration Settings. We evaluate CACHECODEC in a heterogeneous multi-LLM collaboration setting, where a Sharer communicates its KV-cache representations to a Receiver for downstream inference. We fix Qwen3-0.6B (Yang et al., 2025) as the Receiver and pair it with three Sharers: Qwen2.5-0.5B-Instruct (Yang et al., 2024), Llama-3.2-1B (Grattafiori et al., 2024), and Qwen3-4B-Base (Yang et al., 2025). These Sharers span different model families, scales, and training configurations, providing diverse sources of KV representations for evaluating CACHECODEC. Keeping the Receiver fixed isolates variation on the Sharer side and enables controlled evaluation across heterogeneous cache representations.

Cache integration. For each model pair, we evaluate CACHECODEC under two cache integration mechanisms: C2C-style *Fusion* (Fu et al., 2026) and *Concatentation*. In the Fusion setting, the transmitted Sharer cache is integrated with the Receiver's native cache through the original fusion module. In the Concatentation setting, the recovered cache is incorporated as a prefix representation and consumed together with the Receiver input. Importantly, CACHECODEC operates only on the cache communication path and leaves the downstream integration mechanism unchanged, allowing it to be evaluated independently of how the Receiver consumes the communicated cache.

Experimental Protocol. We compare CACHECODEC with text-to-text communication (T2T), Sharer-only, Receiver-only, and the corresponding uncompressed cache-communication baselines. We evaluate multiple-choice accuracy on OpenBookQA (Mihaylov et al., 2018), MMLU-Redux (Gema et al., 2025), ARC-Challenge (Clark et al., 2018), and C-Eval (Huang et al., 2023). We use OpenHermes2.5 (Teknium, 2023) for training, while keeping both the Sharer and Receiver frozen

and optimizing only the learnable communication modules. Additional implementation, training, and evaluation details are provided in Appendix A.1.

Evaluation Metrics. We evaluate CACHECODEC from three complementary perspectives: downstream task utility, communication cost, and end-to-end efficiency. We measure task utility using multiple-choice accuracy, communication cost using the cache compression ratio (CR), and overall system efficiency using end-to-end latency, defined as

$$T_{\text{e2e}} = T_{\text{S,prefill}} + T_{\text{enc}} + T_{\text{trans}} + T_{\text{dec}} + T_{\text{integrate}} + T_{\text{R,prefill}} + T_{\text{gen}}, \tag{7}$$

where $T_{\text{S,prefill}}$ and $T_{\text{R,prefill}}$ denote the Sharer and Receiver prefill time, respectively; T_{enc} and T_{dec} are CACHECODEC encoding and decoding time; T_{trans} is the payload transmission time; $T_{\text{integrate}}$ denotes the Receiver-side cache integration time via concatenation or fusion; and T_{gen} is the Receiver's autoregressive generation time. For example i, let b_i denote the complete serialized bitstream, including decoding metadata, and let S_i^{base} denote the corresponding base-cache size. The cache compression ratio (CR) is defined as

$$CR_{\text{cache}} = \frac{\sum_i S_i^{\text{base}}}{\sum_i |b_i|}, \tag{8}$$

where a larger value indicates a lower payload.

4.2 MAIN RESULTS

Performance. Table 1 shows that CACHECODEC can be integrated into both direct cache concatenation and C2C-style cache fusion, consistently improving downstream performance across different Sharer models. Compared with their uncompressed counterparts, CACHECODEC improves the average accuracy by 0.55%, 3.32%, and 3.48% under concatentation, and by 1.30%, 2.03%, and 1.93% under C2C. These results show that compressed KV caches preserve, and sometimes improve, downstream utility in cross-model integration and compression. These gains are achieved while substantially reducing the end-to-end latency compared with uncompressed cache communication, demonstrating that CACHECODEC improves both task utility and communication efficiency.

End-to-End Efficiency. CACHECODEC achieves consistent end-to-end latency reductions across different Sharer models and cache integration mechanisms. Under the 50 Mbps bandwidth setting in Table 1, The largest reduction is observed with Qwen3-4B-Base, where the larger uncompressed KV payload makes transmission a greater share of the overall latency. CACHECODEC remains effective in even higher bandwidth (e.g. 400Mbps, details in Table 2).

4.3 ABLATION STUDIES

CACHECODEC architecture. Table 3 shows the effect of LCC and FreqCodec. Compared with uncompressed C2C, adding LCC improves the average accuracy by 1.17% with a $7.90\times$ compression factor. Adding FreqCodec further improves accuracy by 2.03% while increasing the compression factor to $871.1\times$.

Latent Channel Dimension. We study the latent channel dimension of LCC to characterize the trade-off between representation capacity and compression efficiency. With LCC alone, using 128 and 256 latent channels improves the average accuracy over the uncompressed C2C baseline by 1.17% and 0.72%, respectively. When integrated into the full CACHECODEC pipeline, the corresponding gains increase to 2.03% and 1.83%. (details in Appendix A.2.1).

Joint K/V compression. We investigate whether the key and value representations should be compressed jointly or independently within LCC. In the joint design adopted by CACHECODEC, K and V are concatenated and encoded into a shared representation, from which K- and V-specific representations are decoded. In contrast, the split design uses separate LCC branches for K and V, with each branch assigned a dimension of $d_z/2$ to keep the total representation budget unchanged. As shown in Appendix A.2.2, these results suggest that jointly modeling K and V provides a small but consistent benefit, supporting the joint design adopted in CACHECODEC. CCA analysis in Appendix A.3 further identifies shared structure between K and V, supporting the use of a shared representation.

Metric	T2T	Raw		CACHECODEC	
		Concat	C2C	Concat	C2C
CACHECODEC Encode (ms)	-	-	-	22	25
Transport Time (ms)	0.007	324	423	0.93	0.3
CACHECODEC Decode (ms)	-	-	-	19	22
Generation (ms)	1054*+140	110	132	112	100
End-to-End Time (ms)	1239	508	659	**218**	**234**

Table 2: Latency breakdown for Qwen3-4B-Base→Qwen3-0.6B at 400 Mbps. *Includes 1054 ms sharer token re-encoding time. End-to-End reports the complete pipeline latency defined in Eq. 7. Sharer/ Receiver prefill and integration are included in End-to-End latency but omitted from the breakdown because they are identical across Raw and CACHECODEC methods. "-" indicates that CACHECODEC encoding or decoding is not applicable.

Method	MMLU-Redux		ARC-C		OpenBookQA		C-Eval		Average	
	Acc	CR(×)	Acc	CR(×)	Acc	CR(×)	Acc	CR(×)	Acc	CR(×)
C2C	44.4	1.00	52.0	1.00	46.8	1.00	40.4	1.00	45.8	1.00
+ LCC	43.9	7.90	53.9	7.89	48.6	7.87	41.2	7.90	46.9	7.90
CACHECODEC	**44.8**	**958.5**	**55.3**	**847.7**	**49.6**	**755.8**	**41.9**	**922.1**	**47.9**	**871.1**

Table 3: Ablation of the main CACHECODEC components for Llama-3.2-1B → Qwen3-0.6B under C2C integration. *C2C*: directly transmits the uncompressed KV cache; *+LCC*: compresses the KV cache into a compact representation; CACHECODEC: further applies FreqCodec to quantize and encode the representation into a transmitted bitstream.

Effect of entropy coding level. We further evaluate CacheCodec under different zlib compression levels. As shown in Appendix A.4, CacheCodec consistently supports different entropy coding settings while maintaining stable downstream accuracy. Higher compression levels reduce the serialized payload, with diminishing returns at more aggressive settings. These results demonstrate that CacheCodec is compatible with different entropy coding configurations without affecting task performance.

4.4 EFFECTIVENESS AND SYSTEMS EVALUATION

Effectiveness across heterogeneous model pairs. As shown in Table 1, CACHECODEC substantially reduces the transmitted payload across heterogeneous Sharer–Receiver pairs. Under both Concatenation and C2C, it maintains or improves average downstream accuracy over the corresponding uncompressed baselines, demonstrating robustness across different model architectures and scales.

End-to-end latency across network bandwidths. Figure A.5 shows that CACHECODEC consistently reduces end-to-end latency under bandwidth-constrained settings, with a break-even bandwidth B^* of 134.4–151.2 Mbps across the four benchmarks. Below B^*, the transmission is faster than raw KV transfer. This advantage extends to higher bandwidths for larger KV payloads: with Qwen3-4B-Base, CACHECODEC still reduces end-to-end latency by 57–64% at 400 Mbps under Concatentation and C2C, as detailed in Table 2.

5 CONCLUSION

We introduced CACHECODEC, a task-oriented KV communication framework for bandwidth-constrained heterogeneous multi-LLM systems. Rather than treating cache transmission as faithful reconstruction of model-internal states, CACHECODEC learns a compact Receiver-oriented representation and further converts it into a low-rate bitstream through frequency-domain coding. Across heterogeneous Sharer–Receiver pairs and cache-integration mechanisms, CACHECODEC substantially reduces communication cost and end-to-end latency while preserving, and in several settings improving, downstream task utility. These results demonstrate the potential of task-oriented latent messages as an efficient communication interface for bandwidth-constrained multi-LLM systems.

AI USE STATEMENT

In preparing this draft, we used generative AI tools for document organization and language editing. Before submission, the authors will review and update this disclosure to cover every applicable use, and they will take responsibility for the final content.

ETHICS STATEMENT

This work studies efficient KV-cache communication for collaboration between large language models. Our experiments use publicly available models and datasets and do not involve human subjects or sensitive personal data. We do not identify any specific ethical concerns introduced by the proposed method beyond those associated with the underlying models and datasets.

REPRODUCIBILITY STATEMENT

We provide the implementation and experimental details necessary to reproduce the results reported in this paper. The CACHECODEC architecture and communication pipeline are described in the main text, while detailed training and evaluation settings, including model configurations, data selection, optimization settings, evaluation protocols, network conditions, and hardware environments, are provided in Appendix A.1.

REFERENCES

Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model, 2024. URL https://arxiv.org/abs/2405.04434.

Nasir Ahmed, T. Natarajan, and K. R. Rao. Discrete cosine transform. *IEEE Transactions on Computers*, C-23(1):90–93, 1974. doi: 10.1109/T-C.1974.223784.

Chi-Chih Chang, Wei-Cheng Lin, Chien-Yu Lin, Chong-Yan Chen, Yu-Fang Hu, Pei-Shuo Wang, Ning-Chi Huang, Luis Ceze, Mohamed S. Abdelfattah, and Kai-Chiang Wu. Palu: KV-cache compression with low-rank projection. In *International Conference on Learning Representations*, 2025.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Lucio M Dery, Zohar Yahav, Henry Prior, Qixuan Feng, Jiajun Shen, and Arthur Szlam. Latent space communication via kv cache alignment. *arXiv preprint arXiv:2601.06123*, 2026.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. *arXiv preprint arXiv:2305.14325*, 2023.

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*, volume 2026, pp. 43130–43158, 2026.

Aryo Pradipta Gema, Joshua Ong Jun Leang, Giwon Hong, Alessio Devoto, Alberto Carlo Maria Mancino, Rohit Saxena, Xuanli He, Yu Zhao, Xiaotang Du, Mohammad Reza Ghasemi Madani, et al. Are we done with MMLU? In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 5069–5096, 2025.

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

Yefei He, Luoming Zhang, Weijia Wu, Jing Liu, Hong Zhou, and Bohan Zhuang. Zipcache: Accurate and efficient KV cache quantization with salient token identification. *Advances in Neural Information Processing Systems*, 37:68287–68307, 2024.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Steven Yau, Zijuan Lin, Liyang Zhou, et al. Metagpt: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations*, volume 2024, pp. 23247–23275, 2024.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W. Mahoney, Yakun Sophia Shao, Kurt Keutzer, and Amir Gholami. KVQuant: Towards 10 million context length LLM inference with KV cache quantization. In *Advances in Neural Information Processing Systems*, volume 37, pp. 1270–1303, 2024.

Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Yao Fu, et al. C-Eval: A multi-level multi-discipline chinese evaluation suite for foundation models. *Advances in Neural Information Processing Systems*, 36: 62991–63010, 2023.

Jushi Kai, Yixuan Wang, Boyi Zeng, Haoli Bai, Bo Jiang, Ziwei He, and Zhouhan Lin. FreqKV: Key-value compression in frequency domain for context window extension. In *International Conference on Learning Representations*, volume 2026, pp. 105568–105587, 2026.

Hao Kang, Qingru Zhang, Souvik Kundu, Geonhwa Jeong, Zaoxing Liu, Tushar Krishna, and Tuo Zhao. GEAR: An efficient KV cache compression recipe for near-lossless generative inference of LLM. *arXiv preprint arXiv:2403.05527*, 2024.

Boris Kriuk and Logic Ng. Q-KVComm: Efficient multi-agent communication via adaptive KV cache compression. In *2026 Second International Conference on Multi-Agent Systems for Collaborative Intelligence (ICMSCI)*, pp. 15–21. IEEE, 2026.

Jin-woo Lee, Minkyung Song, Junghyun Oh, Seunghoon Han, Soyoung Park, Gwangseon Jang, and Sungsu Lim. Mixture-of-translators: Translating KV caches across heterogeneous large language models. *arXiv preprint arXiv:2607.28979*, 2026.

Xing Li, Zeyu Xing, Yiming Li, Linping Qu, Hui-Ling Zhen, Yiwu Yao, Wulong Liu, Sinno Jialin Pan, and Mingxuan Yuan. KVTuner: Sensitivity-aware layer-wise mixed precision KV cache quantization for efficient and nearly lossless LLM inference. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 36451–36485. PMLR, 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*, volume 37, pp. 22947–22970, 2024.

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*, pp. 38–56, 2024a. doi: 10.1145/3651890.3672274.

Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. Scissorhands: Exploiting the persistence of importance hypothesis for LLM KV cache compression at test time. *Advances in Neural Information Processing Systems*, 36:52342–52364, 2023.

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 *Forty-First International Conference on Machine Learning*, 2024b. URL `https://openreview.net/forum?id=L057s2Rq8O`.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2381–2391, 2018.

Maximillian Rossi, Prajwal Raghunath, and Eugene Wu. Latent cache flow: Model-to-model communication without text. *arXiv preprint arXiv:2605.22863*, 2026.

Teknium. Openhermes 2.5: An open dataset of synthetic data for generalist llm assistants, 2023. URL https://huggingface.co/datasets/teknium/OpenHermes-2.5.

Gregory K. Wallace. The JPEG still picture compression standard. *Communications of the ACM*, 34(4):30–44, 1991. doi: 10.1145/103085.103089.

Huanyu Wang, Jushi Kai, Haoli Bai, Lu Hou, Bo Jiang, Ziwei He, and Zhouhan Lin. Fourier compressor: Frequency-domain visual token compression for vision-language models. In *Computer Vision–ECCV 2026: 19th European Conference, Proceedings, Part IX*, pp. 284. Springer Nature, 2026.

Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Y Zou. Mixture-of-agents enhances large language model capabilities. In *International Conference on Learning Representations*, volume 2025, pp. 33944–33963, 2025.

Sunghyeon Woo, Hoseung Kim, Sunghwan Shim, Minjung Jo, Hyunjoon Jeong, Jeongtae Lee, Joonghoon Kim, Sungjae Lee, Baeseong Park, Se Jung Kwon, et al. Prefillshare: A shared prefill module for kv reuse in multi-llm disaggregated serving. *arXiv preprint arXiv:2602.12029*, 2026.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. Autogen: Enabling next-gen llm applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*, 2023.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations*, 2024.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, et al. Qwen2.5-Math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Hancheng Ye, Zhengqi Gao, Mingyuan Ma, Qinsi Wang, Yuzhe Fu, Ming-Yu Chung, Yueqian Lin, Zhijian Liu, Jianyi Zhang, Danyang Zhuo, and Yiran Chen. KVCOMM: Online cross-context KV-cache communication for efficient LLM-based multi-agent systems. *Advances in Neural Information Processing Systems*, 38:17882–17928, 2026.

Hongxin Zhang, Weihua Du, Jiaming Shan, Qinhong Zhou, Yilun Du, Joshua B Tenenbaum, Tianmin Shu, and Chuang Gan. Building cooperative embodied agents modularly with large language models. In *International Conference on Learning Representations*, volume 2024, pp. 19373–19401, 2024.

Rongzhi Zhang, Kuan Wang, Liyuan Liu, Shuohang Wang, Hao Cheng, Chao Zhang, and Yelong Shen. LoRC: Low-rank compression for LLMs KV cache with a progressive compression strategy, 2025. URL https://openreview.net/forum?id=NI8AUSAc4i.

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. H_2O: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36:34661–34710, 2023.

Yujia Zheng, Zhuokai Zhao, Zijian Li, Yaqi Xie, Mingze Gao, Lizhu Zhang, and Kun Zhang. Thought communication in multiagent collaboration. *Advances in Neural Information Processing Systems*, 38:123389–123418, 2026.

A APPENDIX

A.1 TRAINING AND EVALUATING SETTINGS

Table 4: Training and evaluation settings.

Setting	Value
Sharer	Qwen2.5-0.5B-Instruct Qwen3-4B-Base Llama-3.2-1B
Receiver	Qwen3-0.6B
Training dataset	OpenHermes 2.5(Teknium, 2023), first 120K samples
Train/validation split	0.99 / 0.01
Max sequence length	2048 tokens
Layer mapping	Last aligned
Trainable modules	Projector only (LLMs frozen)
Training objective	Response-only causal CE
Optimizer	AdamW
Learning rate	1×10^{-4}
Weight decay	0.01
LR scheduler	Linear
Warmup ratio	0.1
Max grad norm	1.0
Per-device batch size	Concat: 1; C2C: 4
Gradient accumulation	Concat: 16; C2C: 8
Effective batch size	Concat: 128; C2C: 256
Training steps	900
Seed	42
Training hardware	$8\times$ A100 80GB
Evaluation benchmarks	MMLU-Redux (Gema et al., 2025) ARC-Challenge (Clark et al., 2018) OpenBookQA (Mihaylov et al., 2018) C-Eval (Huang et al., 2023)
quant table	low:1 high:8
Decoding	Greedy
Max generate tokens	16
T2T communication limit	64 tokens
Network bandwidth	50 Mbps
Evaluation hardware	$8\times$ L20 48GB

A.2 ADDITIONAL ABLATION EXPERIMENTS RESULTS

A.2.1 ABLATION OF THE LATENT CHANNEL DIMENSION

Table 5: Ablation of the latent channel dimension for Llama-3.2-1B $\rightarrow$ Qwen3-0.6B under C2C integration. We compare 128 and 256 latent channels for both LCC alone and the full CACHECODEC pipeline.

Method	MMLU-Redux	ARC-C	OpenBookQA	C-Eval	Average	CR
C2C	44.4	52.0	46.8	40.4	45.8	1.00
+ LCC-128	43.9	53.9	48.6	41.2	46.9	7.90
+ LCC-256	44.1	52.3	48.8	41.0	46.5	3.95
CACHECODEC-128	**44.8**	**55.3**	49.6	41.9	**47.9**	**871.1**
CACHECODEC-256	43.7	54.7	**49.8**	**42.6**	47.7	435.5

A.2.2 JOINT VERSUS SPLIT OF LATENT CHANNEL COMPRESSOR

Method	MMLU-Redux	ARC-C	OpenbookQA	C-Eval	Average	Params (M)
Split LCC	40.7	52.1	50.6	40.1	45.8	36.87
Joint LCC	**41.2**	**53.0**	**51.0**	**40.4**	**46.4**	42.83

Table 6: Ablation of joint versus separate K/V compression in LCC for Qwen3-4B-Base → Qwen3-0.6B under C2C integration. Joint LCC jointly encodes K and V into a shared latent representation, whereas Split LCC compresses them independently.

A.3 K–V ALIGNMENT IN THE SHARED LATENT SPACE

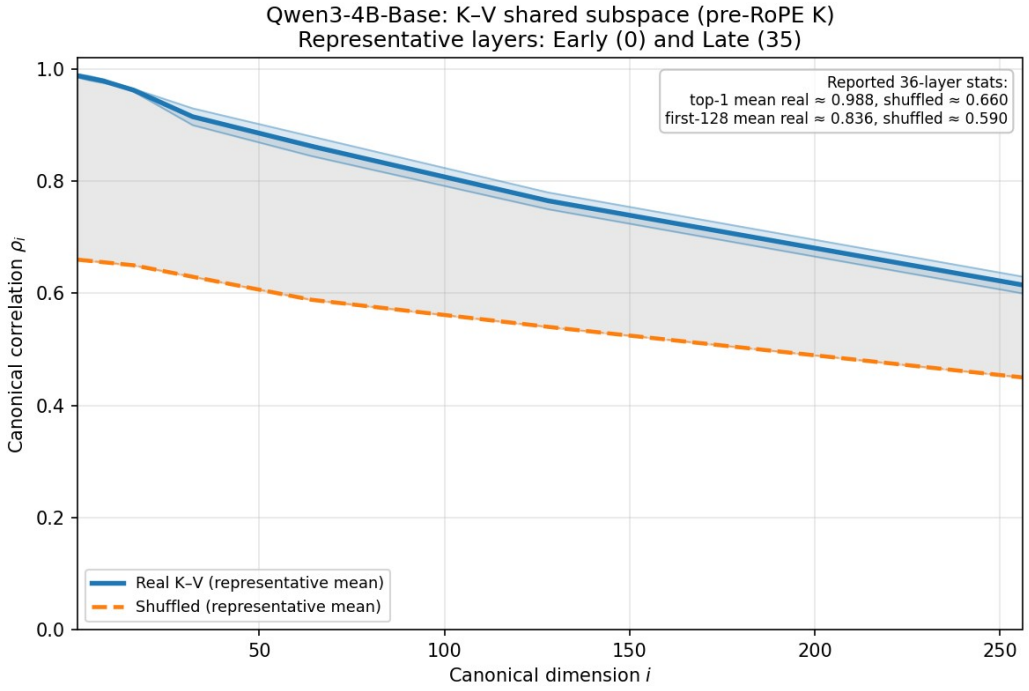


Figure 6: K–V alignment in the shared latent space. Canonical correlation spectrum between the projected K and V representations of LCC, compared with a shuffled K–V baseline. The consistently stronger correlations of the aligned K–V pairs, particularly in the leading canonical dimensions, provide empirical evidence that K and V are mapped into a shared latent subspace.

A.4 Payload-Complexity Trade-off of Entropy Coding

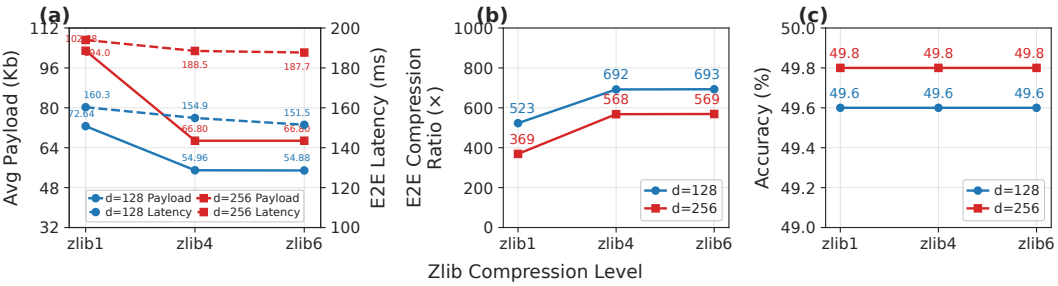


Figure 7: Payload-Complexity Trade-off of Entropy Coding: effect of the zlib compression level for latent widths $d_z = 128$ and $d_z = 256$. (a) Average serialized payload and end-to-end latency. (b) KV-cache compression ratio. (c) Downstream accuracy. Increasing the zlib level from 1 to 4 reduces the payload and improves the compression ratio without changing accuracy, while increasing it further to 6 provides little additional benefit.

A.5 End-to-End Latency Versus Communication Bandwidth

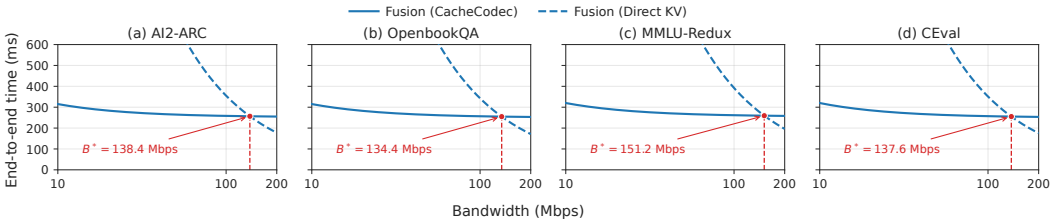


Figure 8: End-to-End latency versus communication bandwidth for the Qwen2.5-0.5B-Instruct → Qwen3-0.6B pair under fusion-based cache integration across four benchmarks. Solid lines denote CacheCodec and dashed lines denote direct KV-cache transmission. Red markers and vertical dashed lines indicate the break-even bandwidth B^* at which the two methods have equal latency; below B^*, CacheCodec achieves lower end-to-end latency.