Swarm Telepathy: Prefill-Free KV-Cache Transfer Across Families

Abdulrahman Alabdulkareem[1,*] Irene Tenison[1] Yang Zhang[2]
Yada Zhu[2] Boris Katz[1] Dan Gutfreund[2]

[1]MIT CSAIL [2]MIT-IBM Computing Research Lab

Project Page: https://github.com/Ar-Kareem/SwaT

arXiv:2610.15095v1 [cs.LG] 10 Oct 2026

Abstract

LLM multi-agent systems split work across agents that communicate through text. An agent summarizes its findings, another agent receives the text. This constrains communication to discrete tokens and costs redundant compute, limiting bandwidth and capabilities of multi-agent systems. Recent works demonstrate KV-cache transfer between LLMs but they either require the LLMs to be within the same family or cannot function across context, thus requiring the receiver to prefill text. We present SwaT (Swarm Telepathy), the first cross-family KV-cache transfer method that allows prefill-free communication in multi-family multi-agent systems. In an agent-to-agent communication setting, combining SwaT with recent advances in KV-cache compaction [41] beats text-to-text communication in both accuracy and speed, with a mean of 40% higher accuracy and a mean of 4× faster wall-clock time. Experiments show SwaT achieves high performance on 17 benchmarks across 12 LLMs and 6 families. Additionally, given a sender Vision-Language Model (VLM), SwaT is able to transfer the vision capabilities to a text-only receiver LLM exceeding text-to-text communication despite SwaT being trained on text-only data. SwaT keeps both sender and receiver models frozen without fine-tuning and only requires $< 50M$ tokens and < 1 GPU-day to train.

Keywords

Multi-Agent Communication, KV-Cache transfer, Multi-Agent Systems, Cross-Family, Cross-Architecture, Cross-Modality

1 Introduction

Multi-agent systems (MAS) built from large language models (LLMs) have progressed rapidly, with agents that specialize, divide tasks, and pool expertise [21, 25, 27, 32], including systems that drive scientific discovery [9]. In nearly all of these systems agents communicate in natural language: one agent reads its inputs, writes what it found, and the next agent reads that text. This simple protocol has delivered consistent gains over single-model performance [2, 3, 5, 7, 14, 16, 24, 29, 38].

Text is nonetheless a narrow channel. A summary is bounded by what its author chooses to write and by what the reader can reconstruct from it. This process is also costly: the sender must generate the output (decode token by token), and the receiver must consume it from scratch (prefill) before doing any work of its own. Internally, however, the sender has already computed a far richer representation of its context, its key-value cache (KV-cache), which holds every layer's view of every token and represents the entire state an agent requires to generate the next token. The KV-cache is therefore a natural channel for agent communication: the sender agent has already computed it and it is denser than text.

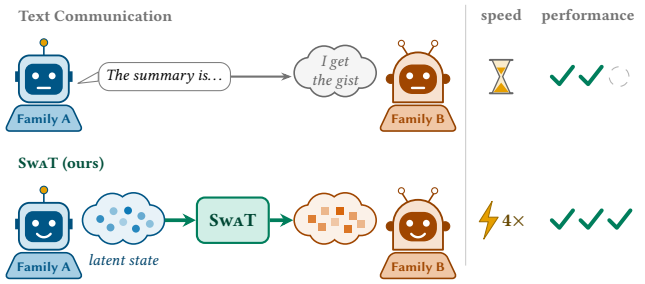


Figure 1: Communication between two LLM agents from different families. Top: communicating in words is slow and the receiver only gets the gist. Bottom: SwaT translates the sender's latent state (KV-cache) to the receiver's latent space. Combined with equal-budget cache compression, SwaT is on average 40% more accurate than text summarization, at 4× the speed.

Recent work confirms the promise of this communication channel when all agents share a single backbone. LatentMAS [40] shows that agents exchanging latent states and caches carry substantially more information per token than agents exchanging text. Similarly, single-model latent chain-of-thought [12] reaches the same conclusion. But these results apply to a single model communicating within copies of itself, where sender and receiver caches have similar geometry.

Recent KV-cache transfer across context relaxes this constraint to models of different sizes and layer counts within one family [4, 6, 13, 26, 28, 33]. However, models in the same family (e.g., Qwen 3 at 4B, 8B, 14B, and 32B) are trained by the same team on similar pipelines and data, and smaller members are often distilled. This leads to the conclusion that a translation between the internal states of such models is a reasonable task. However, in this work we try to tackle the fundamental problem in replicating this protocol in more complex multi-agent systems which mix models of various sizes, architectures, and families while maintaining that the receiver does not prefill any text. This raises the question we address:

> *Can LLMs from different families communicate without text?*

There is reason to believe the answer is yes. The Platonic Representation Hypothesis [15] argues that models trained on the same world converge toward the same representation and are becoming more aligned due to selective pressures. If so, then LLMs built by

*Corresponding author: arkareem@mit.edu

Table 1: Cross-model KV transfer. SwaT is the first method for cross-family KV-cache transfer with no receiver prefill (i.e., the receiver does not read the text).

Paper	Date	Different Sizes	Cross-Family	Skip Prefill
Closed Form [13]	Aug 2026	✓	✗	✓
CacheBridge [26]	Sep 2026	✓	✗	✓
KV Alignment [6]	Jan 2026	✓	✗	✓
Dense Latent [4]	Jun 2026	✓	✗	✓
LCF [28]	May 2026	✓	✗	✗
LCF-X [28]	May 2026	✗	✗	✓
ICaRus [33]	Mar 2026	✗	✗	✓
Cache-to-Cache [8]	Oct 2025	✓	✓	✗
XKV [20]	Aug 2026	✓	✓	✗
SwaT (ours)	—	✓	✓	✓

different organizations with different architectures and data still share what they represent, differing mainly in how they express it. A sender's internal representations would then already contain what a receiver needs, and cross-family communication reduces to learning a mapping function.

If cross-family communication can bypass receiver prefill, it would enable a multitude of optimizations in the field. In a multi-agent system, a child can hand its parent a compacted cache instead of a summary which saves compute on both sides and carries more information. When switching models mid-conversation, the incoming model can inherit the outgoing model's cache instead of re-prefilling the entire context, which skips the cost of switching models mid-context. In routing systems that dispatch each query to the cheapest capable model [39], a router that has already read the query can pass its cache to the chosen model rather than forcing a second read. In general, throughout LLM pipelines there are many moments when a cache for family A's model exists while family B's model is needed for decoding, and our method SwaT (fig. 1) enables these capabilities. Our contributions are:

- We introduce SwaT, a cross-family KV-cache transfer method that allows communication between LLMs without the receiver prefilling any text.
- Across many model–pairs, families, and sizes, an evaluation across 17 benchmarks shows that SwaT is capable of transferring caches with high performance while exceeding the receiver model's own capabilities in certain setups.
- We demonstrate that communication in a multi-agent system is more effective when combining SwaT with KV-cache compaction techniques such as AM [41] compared to textual summaries between agents. This is evaluated across many datasets, budgets, and sender–receiver pairs.
- We demonstrate that despite SwaT only training on English text, it gives the receiver model zero-shot capabilities from the sender such as vision and Chinese capabilities.

2 Background

Consider a multi-agent system with n LLM agents $\{\mathcal{A}^{(1)}, \ldots, \mathcal{A}^{(n)}\}$. Each agent processes inputs in two stages: *prefill*, which reads an input sequence and computes the agent's internal representations (key-value cache or KV-cache), and *decode*, which uses the KV-cache to generate output tokens autoregressively.

For agent $\mathcal{A}^{(i)}$ and input x, the two stages are:

$$C_x^{(i)} = \mathcal{A}^{(i)}_{\text{prefill}}(x), \qquad y^{(i)} \sim \mathcal{A}^{(i)}_{\text{decode}}(C_x^{(i)}). \tag{1}$$

LLMs by default accumulate their cache to prevent the quadratic cost per generated token ($O(|T_{in}|^2 * T_{out})$ where T_{in} denotes the input tokens and T_{out} is the number of generated tokens). Thus, as the generated tokens are processed, the cache extends to $C^{(i)}_{x\|y^{(i)}}$, where $\|$ denotes sequence concatenation. In text-based communication, $\mathcal{A}^{(i)}$ sends $x\|y^{(i)}$ to another agent $\mathcal{A}^{(j)}$, which reads it through its own prefill:

$$C^{(j)}_{x\|y^{(i)}} = \mathcal{A}^{(j)}_{\text{prefill}}(x\|y^{(i)}), \qquad y^{(j)} \sim \mathcal{A}^{(j)}_{\text{decode}}(C^{(j)}_{x\|y^{(i)}}). \tag{2}$$

The information available to $\mathcal{A}^{(j)}$ is therefore mediated by the message $x\|y^{(i)}$ that $\mathcal{A}^{(i)}$ produces. We refer to $\mathcal{A}^{(i)}$ as the sender, and $\mathcal{A}^{(j)}$ as the receiver.

Instead of the above, we seek to communicate without agent $\mathcal{A}^{(j)}$ receiving text by using agent $\mathcal{A}^{(i)}$'s cache $C^{(i)}$. Concretely, for any sender–receiver pair, we seek to learn a mapper $M_{i\to j}$ that transforms $C_x^{(i)}$ into $\widehat{C}_x^{(j)}$ which approximates the receiver's own prefill cache while preserving (and sometimes exceeding) downstream performance:

$$\widehat{C}_x^{(j)} = M_{i\to j}(C_x^{(i)}), \qquad y^{(j)} \sim \mathcal{A}^{(j)}_{\text{decode}}(\widehat{C}_x^{(j)}, q), \tag{3}$$

where q is the receiver's query and $y^{(j)}$ is its answer. Notice how $\mathcal{A}^{(j)}$ was not given the raw input x.

During decoding (such as in eq. (1)), an LLM accesses its previously processed context only through its KV-cache, rather than the raw input. The cache therefore serves as the agent's accumulated internal state, conditioning its subsequent actions. Transferring this cache allows another agent to continue from that state. We call this direct transfer of internal state *telepathy* within the *swarm*, giving our method its name: SwaT (Swarm Telepathy).

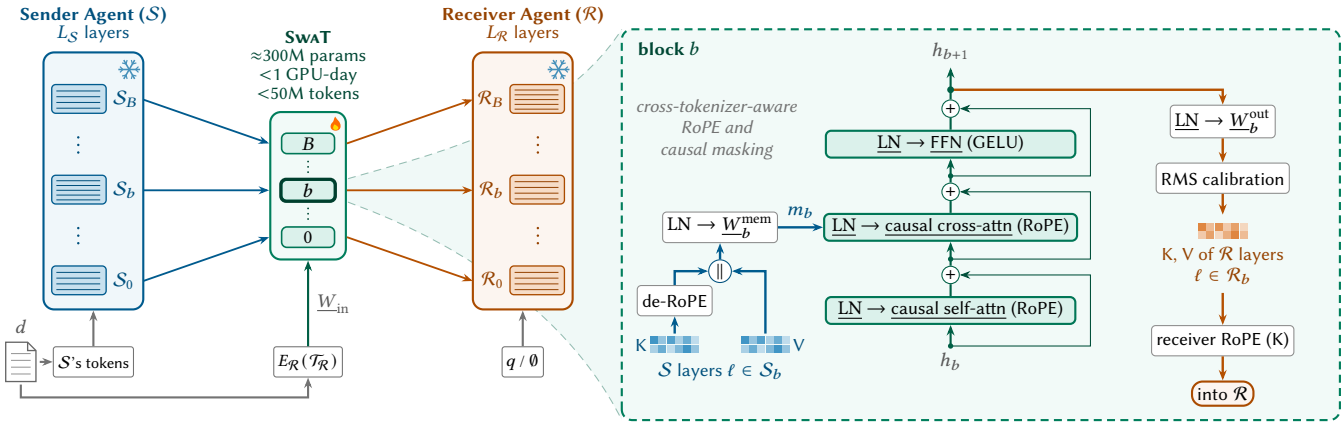


Figure 2: SwAT between a frozen sender agent ($\mathcal{S}$) and receiver agent ($\mathcal{R}$). Left: $\mathcal{S}$ prefills the document d once. SwAT has B blocks where block b cross-attends to $\mathcal{S}$'s KV-cache for the layer group $\mathcal{S}_b$ and outputs $\mathcal{R}$'s KV-cache for the layer group $\mathcal{R}_b$. Only SwAT is trained where underlining marks learned parameters or modules. Right: block b maps h_b to h_{b+1} through causal self-attention on h_b, then causal cross-attention on a linear projection of $\mathcal{S}_b$ (RoPE removed), then an FFN. Positions and the causal mask in the cross-attention module are aligned across the tokenizers as described in section 3.2. The output of b is linearly projected to $\mathcal{R}_b$ with receiver RoPE applied to the keys.

2.1 Summarization vs Attention Matching

In the preceding formulation, agent $\mathcal{A}^{(i)}$ passes the full sequence $x\|y$ to $\mathcal{A}^{(j)}$. In practical multi-agent systems, this sequence contains lengthy intermediate outputs which are unnecessary for the receiver's task. This can be computationally prohibitive and could degrade performance by introducing context pollution [17, 23]. An effective solution is to communicate through summarization.

Text-to-Text communication (T2T) involves the following steps:

Sender task. Agent $\mathcal{A}^{(i)}$ processes input x and achieves its task with generated output y:

$$C_x^{(i)} = \mathcal{A}^{(i)}_{\text{prefill}}(x), \qquad y \sim \mathcal{A}^{(i)}_{\text{decode}}(C_x^{(i)}). \tag{4}$$

At the end of generation, the cache becomes $C^{(i)}_{x\|y}$. In our experiments, the document d from the dataset is $x\|y$.

Summarization. The sender generates a summary s of its findings, conditioned on $C^{(i)}_{x\|y}$ and a summarization instruction p_{sum}:

$$s \sim \mathcal{A}^{(i)}_{\text{decode}}\left(C^{(i)}_{x\|y}, p_{\text{sum}}\right). \tag{5}$$

Receiver prefill. $\mathcal{A}^{(j)}$ reads the summary to construct its cache:

$$C_s^{(j)} = \mathcal{A}^{(j)}_{\text{prefill}}(s). \tag{6}$$

Receiver task. Given a subsequent input $x^{(j)}$, the receiver achieves its task using the prefilled summary:

$$y^{(j)} \sim \mathcal{A}^{(j)}_{\text{decode}}\left(C_s^{(j)}, x^{(j)}\right). \tag{7}$$

Text-to-Text communication significantly reduces the amount of context the receiver must process, but introduces an additional autoregressive generation step at the sender. The information conveyed from $x\|y$ is also limited to what the sender includes in s.

Attention Matching (AM) [41] is an effective KV-cache compaction method that constructs a smaller set of keys and values yet preserves the influence of the original KV-cache. It introduces a hyperparameter U which is the retained cache budget that controls the trade-off between cache size and fidelity.

KV-cache compaction is a natural fit after SwAT: the sender already holds the accumulated cache, and SwAT translates it into a representation the receiver can use directly. Compacting the representation produces a message without generating text or requiring the receiver to prefill text, eliminating steps in eqs. (5) and (6). Specifically, we combine SwAT with AM (SwAT+AM) by translating the sender's cache and compacting it ($\widetilde{C}_U^{(j)}$). Then the transferred and compacted cache ($\widetilde{C}_U^{(j)}$) is given to the receiver to begin decoding:

$$\widetilde{C}_U^{(j)} = \text{AM}_U\left(M_{i\to j}(C^{(i)}_{x\|y})\right), \qquad y^{(j)} \sim \mathcal{A}^{(j)}_{\text{decode}}\left(\widetilde{C}_U^{(j)}, x^{(j)}\right). \tag{8}$$

In T2T, the realized summary length (i.e. the number of tokens generated by $\mathcal{A}^{(i)}_{\text{decode}}(C^{(i)}_{x\|y}, p_{\text{sum}})$) depends on the sender $\mathcal{A}^{(i)}$, its accumulated context $C^{(i)}_{x\|y}$, and the summary prompt p_{sum}. For a comparison at matched budgets, we choose U to match the length of the summary s generated in eq. (5), measured in receiver tokens. This compares T2T and SwAT+AM at comparable cache budgets.

3 Method

For a sender–receiver pair, we train SwAT, which is a transformer mapper $\underline{M_{i\to j}}$, keeping both LLMs frozen. We underline trainable parameters or modules. SwAT's architecture is controlled by hyperparameters block count B, hidden width D, attention-head count H, and feed-forward width F.

For agent $\mathcal{U} \in \{\mathcal{S}, \mathcal{R}\}$, let $L_\mathcal{U}$, $H_\mathcal{U}$, and $\delta_\mathcal{U}$ denote its KV-layer count, KV-head count, and head dimension; let $D_\mathcal{R}$ be the receiver embedding dimension. These dimensions are fixed by the frozen agents. We choose $B \ll \min(L_\mathcal{S}, L_\mathcal{R})$.

Table 2: Zero-shot retention (%, $\text{score}_{\text{SwAT}}/\text{score}_{\text{receiver}}$) across 17 benchmarks.

Sender		Receiver	In-domain (9)			OOD (8)			Mean
			Short Acc	Context Acc	Context F1	Short Acc	Context Acc	Context F1	
Small	→	**Large**							
Qwen 3 4B	→	Llama 3 8B	100.6%	100.2%	110.9%	88.7%	91.6%	98.7%	**98.5%**
Llama 3 8B	→	Qwen 3 14B	89.6%	98.2%	116.1%	86.5%	98.9%	103.5%	**97.5%**
Ministral 3 8B	→	Llama 3.2 11B	104.2%	97.5%	108.5%	89.7%	92.8%	99.0%	**99.4%**
Gemma 3 4B	→	Granite 3.3 8B	95.4%	93.6%	159.8%	82.0%	95.0%	131.4%	**108.5%**
Qwen 3 4B	→	Gemma 2 9B	94.0%	92.4%	104.2%	83.4%	94.5%	96.3%	**94.2%**
Gemma 2 9B	→	Ministral 3 14B	101.9%	95.4%	110.0%	104.2%	95.3%	106.9%	**102.8%**
Gemma 3 4B	→	Llama 3 8B	90.2%	101.6%	106.9%	73.3%	91.9%	99.0%	**92.2%**
Large	→	**Small**							
Gemma 3 12B	→	Qwen 3 4B	104.0%	100.1%	120.5%	97.1%	97.7%	105.8%	**104.6%**
Qwen 3 32B	→	Llama 3 8B	112.9%	104.0%	98.4%	102.5%	95.4%	97.1%	**103.0%**
Granite 3.3 8B	→	SmolLM3 3B	112.4%	103.9%	253.6%	113.4%	114.7%	162.1%	**143.2%**

Layer groups. We partition each agent's KV layers into B contiguous groups. For block $b \in \{0, \dots, B-1\}$,

$$\mathcal{U}_b = \left\{ \ell : \left\lfloor \frac{bL_{\mathcal{U}}}{B} \right\rfloor \le \ell < \left\lfloor \frac{(b+1)L_{\mathcal{U}}}{B} \right\rfloor \right\}, \quad \mathcal{U} \in \{\mathcal{S}, \mathcal{R}\}. \tag{9}$$

Here, $|\mathcal{U}_b|$ denotes the number of layers in group $\mathcal{U}_b$. Block b reads cache from $\mathcal{S}_b$ and predicts cache for $\mathcal{R}_b$, allowing each receiver layer in its group to draw on multiple sender layers.

Mapper inputs. Let S and R be the sender and receiver token counts for document d, respectively, and represent token vectors as columns. We tokenize d with the receiver's tokenizer, obtaining $\mathcal{T}_{\mathcal{R}} = (r_0, \dots, r_{R-1})$, and initialize the hidden states as $h_0 = \underline{W}_{\text{in}} E_{\mathcal{R}}(\mathcal{T}_{\mathcal{R}})$, where $\underline{W}_{\text{in}} \in \mathbb{R}^{D \times D_{\mathcal{R}}}$ acts token-wise and $h_0 \in \mathbb{R}^{D \times R}$. The frozen embedding table lookup $E_{\mathcal{R}}(\mathcal{T}_{\mathcal{R}})$ requires 0 FLOPs. For block b, we remove sender RoPE from its cached keys and concatenate these keys and values across heads and layers in $\mathcal{S}_b$ at each sender token. Applying $\underline{W}_b^{\text{mem}} \in \mathbb{R}^{D \times (2|\mathcal{S}_b| H_{\mathcal{S}} \delta_{\mathcal{S}})}$ produces memory $m_b \in \mathbb{R}^{D \times S}$.

SwaT blocks. Block b updates h_b through causal self-attention, causal cross-attention to m_b, and a feed-forward network, producing h_{b+1}. Each sublayer has its own $\underline{\text{LN}}$ and residual connection. Both attention sublayers learn projections $\underline{W}_b^Q, \underline{W}_b^K, \underline{W}_b^V, \underline{W}_b^O \in \mathbb{R}^{D \times D}$, with separate weights for self- and cross-attention and H heads of dimension D/H. Cross-attention uses m_b as its key/value memory. The FFN uses $\underline{W}_b^{(1)} \in \mathbb{R}^{F \times D}$ and $\underline{W}_b^{(2)} \in \mathbb{R}^{D \times F}$ with GELU. Self-attention uses integer receiver positions; cross-attention uses the character-based RoPE alignment and causal mask described in section 3.2.

Cache outputs. Each block applies its own $\underline{\text{LN}}$ and output projection $\underline{W}_b^{\text{out}} \in \mathbb{R}^{(2|\mathcal{R}_b| H_{\mathcal{R}} \delta_{\mathcal{R}}) \times D}$ to h_{b+1}, predicting keys and values for all layers in $\mathcal{R}_b$. Predicted keys and values are separately rescaled per layer and head to match fixed RMS statistics calibrated from receiver caches on training contexts. We then apply receiver RoPE to the predicted keys at integer receiver positions. Assembling all blocks' outputs yields $\widehat{C}_d^{(j)}$, with one cache entry per receiver token in every receiver KV layer. The receiver processes q on top of this cache and generates its answer.

Sliding-window layers. Some models such as Gemma 2/3 utilize sliding-window layers that only attend to recent tokens and store the cache of only those. To handle a sliding-window layer l_i, if $l_i \in \mathcal{S}_b$ then we force l_i to record all past KV-cache even outside of its window and make that an input to SwAT (l_i's attention equation is not changed). If $l_i \in \mathcal{R}_b$, we make SwAT predict the KV-cache for all tokens in l_i, and afterwards l_i acts normally and drops KV-cache for old tokens that it does not need.

3.1 Training Objectives

We train SwAT with two objectives, keeping both agents frozen. Each training example pairs a document d with a task prompt q and a target continuation y, an answer or reasoning trace. Let $C_d^{\mathcal{S}}$ and $C_d^{\mathcal{R}}$ denote the sender's and receiver's prefill caches of d, so that SwAT predicts the receiver cache as $\underline{M}_{i \to j}(C_d^{\mathcal{S}}, \mathcal{T}_{\mathcal{R}})$. The first objective is to reconstruct the receiver's native cache using RMS-normalized mean squared error:

$$\mathcal{L}_{\text{KV}} = \text{MSE}\left(\frac{\underline{M}_{i \to j}(C_d^{\mathcal{S}}, \mathcal{T}_{\mathcal{R}})}{\sigma}, \frac{C_d^{\mathcal{R}}}{\sigma} \right), \tag{10}$$

where σ is the RMS of $C_d^{\mathcal{R}}$ for keys and values in each layer-head. Keys are compared before RoPE.

The second objective is to optimize the transferred cache for downstream generation using the next-token prediction loss

$$\mathcal{L}_{\text{NLL}} = -\frac{1}{|y|} \sum_{t=1}^{|y|} \log p_{\mathcal{R}}\left(y_t \mid \underline{M}_{i \to j}(C_d^{\mathcal{S}}, \mathcal{T}_{\mathcal{R}}), q, y_{<t} \right). \tag{11}$$

Gradients pass through the frozen receiver to update only $\underline{M}_{i \to j}$.

3.2 Cross-Family Tokenizers

Our alignment strategy preserves both agents' native tokenizations. Sender and receiver tokens interact only through causal

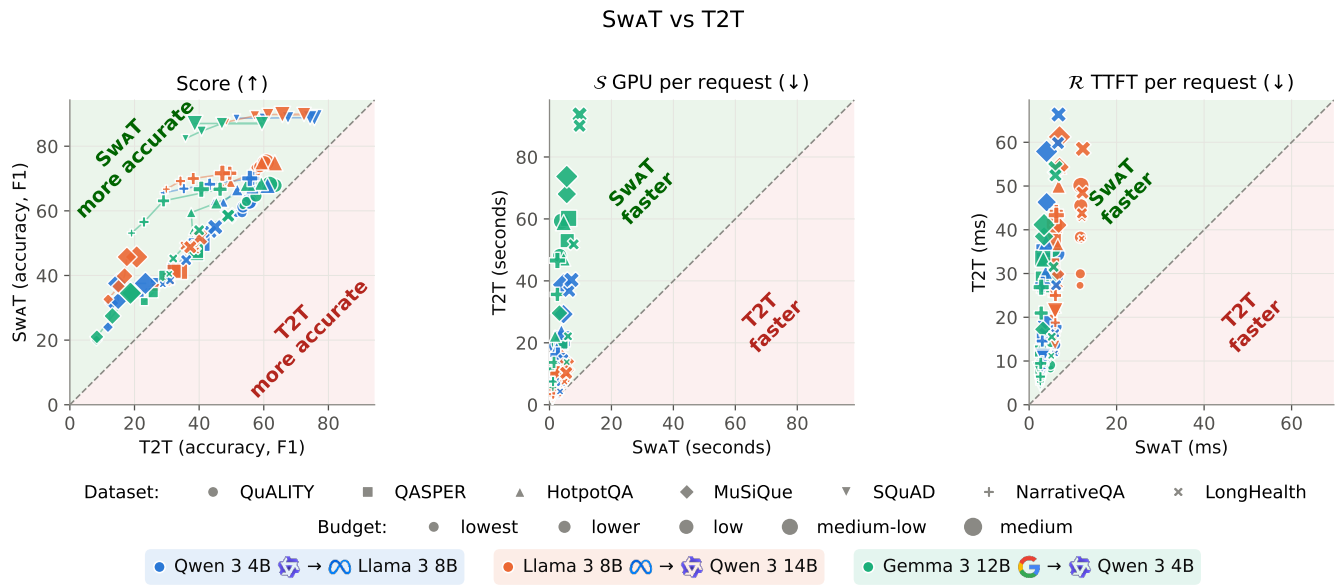


Figure 3: Comparing SwAT (ours) with AM [41] against text-to-text communication (T2T) across three model–pairs, five budgets, and seven datasets. SwAT+AM exceeds T2T in all metrics (task score, sender GPU time, receiver GPU TTFT).

cross-attention, which bridges their representations despite differing sequence lengths and token boundaries. We align positional encodings and construct the causal attention mask by relying on the token character spans, which are defined over the shared characters.

Let document d have D characters $d = (c_1, \ldots, c_D)$ with sender tokens $\mathcal{T}_\mathcal{S} = (s_0, \ldots, s_{S-1})$ and receiver tokens $\mathcal{T}_\mathcal{R} = (r_0, \ldots, r_{R-1})$. For each sender token s_i and receiver token r_j call their character spans $[a_i^\mathcal{S}, b_i^\mathcal{S})$ and $[a_j^\mathcal{R}, b_j^\mathcal{R})$.

RoPE alignment. For sender token s_i, call its midpoint $m_i = (a_i^\mathcal{S} + b_i^\mathcal{S})/2$, and choose the largest j with $a_j^\mathcal{R} \le m_i$ (or $j = 0$ if none exists). Then, its cross-attention RoPE position is

$$p_i = \max\left(0,\ j + \mathrm{clip}_{[0,1]}\left(\frac{m_i - a_j^\mathcal{R}}{b_j^\mathcal{R} - a_j^\mathcal{R}}\right) - \frac{1}{2}\right). \tag{12}$$

Receiver tokens retain integer positions. Tokenizer templates are linearly interpolated. Notice how if both tokenizers are the same, $\mathcal{T}_\mathcal{S} = \mathcal{T}_\mathcal{R}$ and their character spans match, so eq. (12) reduces to $p_i = i$. This alignment strategy avoids prior token alignment methods from Fu et al. [8] where a mismatch between the tokenizers ends up dropping extra tokens causing the model to possibly lose important pieces of information.

Causal masking. Receiver token r_j may attend to sender token s_i only if $b_i^\mathcal{S} \le b_j^\mathcal{R}$, restricting sender-cache access to the text prefix ending at r_j.

4 Experiments

Our experiments address three questions: **(1)** how much of the receiver's own performance survives when it reads SwAT's transferred cache instead of the text? We measure this across 17 datasets and 10 cross-family sender–receiver pairs in both small-to-large and large-to-small directions (section 4.2 and table 2). **(2)** how does cache transfer compare with communicating through text? We compare SwAT, compressed with Attention Matching (AM), against T2T, at matched communication budgets across 7 datasets, 3 cross-family sender–receiver pairs, and 5 different budgets (section 4.3, fig. 3, and table 3). **(3)** can the transferred cache carry capabilities the receiver lacks? Although SwAT is trained only on English text, we test whether it transfers visual understanding from vision-language senders to text-only receivers (section 4.4.1), and whether it transfers specialized text knowledge and capabilities not seen during training (section 4.4.2 and Appendix F).

4.1 Setup

Models. We train SwAT across sender–receiver pairs drawn from six model families: Qwen 3 [36], Llama 3 [11], Gemma 2/3 [30, 31], Ministral 3 [19], Granite 3 [10], and SmolLM3 [1], ranging from 3B to 32B parameters. Because the number of possible sender–receiver pairs grows combinatorially with the number of models, we evaluate 10 pairs chosen so that every pair crosses model families. Seven pairs transfer from a smaller sender to a larger receiver and three from a larger sender to a smaller one.

Training data. The MSE loss (eq. (10)) trains for 5k steps with samples drawn from PG-19 and WikiText-103 and the training splits of QASPER, QuALITY, HotpotQA, and MuSiQue. The NLL loss (eq. (11)) adds the training splits of ARC-Challenge, HellaSwag, WinoGrande, MMLU (auxiliary), and GSM8K with generated reasoning traces. All training data is English text, totalling 44M unique tokens.

Optimization. Each SwAT has $B \in \{7, 8, 9\}$ blocks with $D = 1024$, $H = 16$, and $F = 4096$, giving approximately 300M trainable

Table 3: Performance of SwaT vs T2T over three model–pairs, five budgets, and the mean over seven datasets (QuALITY, QASPER, HotpotQA, MuSiQue, SQuAD, NarrativeQA, LongHealth). All dataset samples are (d, q): document d is given to the sender agent ($\mathcal{S}$) and question q to the receiver agent ($\mathcal{R}$). T2T: $\mathcal{S}$ summarizes d, $\mathcal{R}$ prefills the summary along with q and proceeds to answer. SwaT+AM: the KV-cache of d from $\mathcal{S}$ is transferred (SwaT) and compacted (AM) then given to $\mathcal{R}$, which prefills only q and proceeds to answer. Budgets are set by the T2T summary from $\mathcal{S}$ under five different summary prompts, and SwaT+AM is compressed to a similar budget. Metrics: task score (accuracy or F1), tokens communicated between the agents (Tokens), GPU time of $\mathcal{S}$ per request ($\mathcal{S}$ GPU), and time to first token of $\mathcal{R}$ per request ($\mathcal{R}$ TTFT), both in milliseconds. Change is reported ± bootstrap standard error. Green: SwaT is statistically better than T2T (95% bootstrap CI).

		Qwen3-4B → Llama3-8B			Llama3-8B → Qwen3-14B			Gemma3-12B → Qwen3-4B		
Metric	Budget	T2T	SwaT	Change	T2T	SwaT	Change	T2T	SwaT	Change
	Lowest	*≈16.2× compression*			*≈12.7×*			*≈17.2×*		
Score		34.2	**53.1**	+18.9±4.6	37.2	**58.6**	+21.4±4.5	29.8	**49.0**	+19.3±5.5
Tokens		117	**113**	−3.1±2.1%	161	**149**	−7.4±3.3%	114	**110**	−3.8±2.2%
$\mathcal{S}$ GPU (ms)		3,608	**1,527**	×2.4±0.4	3,675	**1,897**	×1.9±0.2	7,329	**2,094**	×3.5±0.6
$\mathcal{R}$ TTFT (ms)		9.2	**3.8**	×2.4±0.2	21.8	**7.3**	×3.0±0.3	6.7	**3.2**	×2.1±0.1
	Lower	*≈11.0× compression*			*≈9.7×*			*≈12.5×*		
Score		38.1	**55.1**	+17.0±3.7	39.9	**60.7**	+20.8±3.7	31.7	**52.1**	+20.4±5.0
Tokens		181	**166**	−8.2±4.2%	211	**196**	−7.2±3.6%	159	**152**	−4.3±2.3%
$\mathcal{S}$ GPU (ms)		5,387	**1,661**	×3.2±0.5	4,637	**2,049**	×2.3±0.2	10,229	**2,158**	×4.7±0.7
$\mathcal{R}$ TTFT (ms)		12.2	**3.9**	×3.1±0.3	25.4	**7.4**	×3.4±0.3	8.4	**3.2**	×2.6±0.2
	Low	*≈7.5× compression*			*≈7.9×*			*≈6.9×*		
Score		42.7	**57.7**	+14.9±2.6	42.9	**61.8**	+18.9±3.4	37.4	**56.9**	+19.6±4.6
Tokens		271	**246**	−9.1±3.2%	261	**241**	−7.9±2.7%	297	**274**	−7.7±2.8%
$\mathcal{S}$ GPU (ms)		7,826	**1,909**	×4.1±0.6	6,214	**2,265**	×2.7±0.3	19,402	**2,664**	×7.3±1.0
$\mathcal{R}$ TTFT (ms)		16.1	**4.0**	×4.1±0.4	31.6	**7.4**	×4.2±0.5	13.2	**3.4**	×3.9±0.5
	Medium-low	*≈4.1× compression*			*≈5.5×*			*≈3.5×*		
Score		48.6	**61.4**	+12.8±2.1	46.9	**63.9**	+16.9±2.6	48.0	**61.4**	+13.4±3.1
Tokens		571	**448**	−21.5±5.5%	386	**341**	−11.6±6.9%	676	**535**	−20.8±6.0%
$\mathcal{S}$ GPU (ms)		17,170	**2,912**	×5.9±0.5	8,873	**2,874**	×3.1±0.3	42,526	**4,019**	×10.6±1.1
$\mathcal{R}$ TTFT (ms)		30.3	**4.1**	×7.4±1.1	41.5	**7.6**	×5.5±0.7	25.9	**3.6**	×7.2±1.1
	Medium	*≈3.6× compression*			*≈5.4×*			*≈3.3×*		
Score		51.1	**61.8**	+10.7±1.6	47.4	**64.3**	+16.9±2.9	43.3	**61.6**	+18.3±5.4
Tokens		656	**511**	−22.1±6.4%	412	**353**	−14.2±6.8%	779	**572**	−26.5±6.6%
$\mathcal{S}$ GPU (ms)		19,665	**3,258**	×6.0±0.6	9,734	**2,895**	×3.4±0.3	48,709	**4,155**	×11.7±1.4
$\mathcal{R}$ TTFT (ms)		34.0	**4.2**	×8.1±1.3	45.0	**7.6**	×5.9±0.7	29.4	**3.6**	×8.1±1.3
	Overall									
Score		42.9	**57.8**	+14.9±2.5	42.9	**61.9**	+19.0±3.1	38.0	**56.2**	+18.2±4.6
Tokens		293	**254**	−13.1±3.8%	269	**243**	−9.7±4.3%	309	**269**	−13.1±3.4%
$\mathcal{S}$ GPU (ms)		8,752	**2,150**	×4.1±0.5	6,198	**2,360**	×2.6±0.3	19,761	**2,888**	×6.8±0.8
$\mathcal{R}$ TTFT (ms)		18.0	**4.0**	×4.5±0.5	31.8	**7.5**	×4.3±0.4	14.1	**3.4**	×4.2±0.5

parameters. We train with AdamW, weight decay 10^{-3}, gradient clipping at 1.0, all in bf16. We first minimize eq. (10) for 5,000 steps with a cosine learning rate peaking at 3×10^{-4}, then minimize eq. (11) for 24,000 steps at a constant learning rate of 3×10^{-6}.

Evaluation. We evaluate on 17 LLM benchmarks: 9 whose training splits SwaT saw (in-domain) and 8 held out entirely (out-of-domain). For short task benchmarks (ARC-Challenge, HellaSwag, WinoGrande, MMLU, GSM8K, MedQA, OpenBookQA, MATH-500), each sample is a short question. Thus, we consider d to be the task description, question, and multiple choices (if any), while the query q is empty ($\emptyset$). For long context benchmarks (QuALITY, QASPER, HotpotQA, MuSiQue, BoolQ, RACE, LongHealth, SQuAD, NarrativeQA), we have a long context followed by a question. In these tasks, we consider d to be the long context and the query q to be the question and multiple choices (if any). We report accuracy for multiple-choice and math tasks and token F1 for free-form answers, all with greedy decoding. We additionally evaluate the vision experiments on 5 VLM benchmarks: VQAv2, TallyQA, OK-VQA, GQA, and VizWiz. As SwaT was trained only on English text, it has of course never seen these VLM datasets.

Table 4: Test accuracy (%) of 3 VLM → text-only LLM pairs on 5 VQA benchmarks (TQA=TallyQA, VW=VizWiz). ± is bootstrap standard error. *Sender*: VLM answers from the image and question. *T2T*: the receiver LLM answers from the sender VLM's caption. *SwAT*: the sender VLM's KV-cache is transferred to the text-only receiver. SwAT is significantly better than results marked with * (95% paired bootstrap).

Method	VQAv2	TQA	OKVQA	GQA	VW	Avg.
Gemma 3 4B → Granite 3.3 8B						
Sender	45.3*	53.4*	22.0*	31.4*	**19.0**	34.2±.4*
T2T	33.8*	32.0*	15.7*	16.3*	7.7*	21.1±.4*
SwAT	**59.5**	**61.8**	**28.7**	**39.5**	18.3	**41.6±.4**
Gemma 3 4B → Llama 3 8B						
Sender	45.3*	53.4*	22.0*	31.4*	**19.0**	34.2±.4*
T2T	16.4*	12.3*	18.3*	10.9*	8.4*	13.3±.3*
SwAT	**57.5**	**56.4**	**29.7**	**35.5**	18.7	**39.6±.4**
Gemma 3 12B → Qwen 3 4B						
Sender	44.5*	22.4*	**24.4**	28.3*	**20.0**	27.9±.4*
T2T	47.0*	37.6*	19.1*	29.7*	12.2*	29.1±.4*
SwAT	**51.7**	**49.4**	21.8	**33.0**	15.1	**34.2±.4**

Table 5: Zero-shot specialized knowledge transfer from Qwen 3 4B to Llama 3 8B on CMMLU (Chinese MMLU; accuracy, %). Although our training data contained only English text, SwAT transfers most of Qwen's advantage on foreign text never seen during training. Change is relative to Llama 3 8B.

CMMLU subject	Qwen	Llama	SwAT Q→L	
	3 4B	3 8B	Acc.	Change
Arts	90.0	55.6	88.8	+59.6%
Security study	83.7	55.6	77.0	+38.7%
Concept. physics	83.7	45.6	73.5	+61.2%
Prof. psychology	82.8	56.0	77.6	+38.5%
All 67 subjects	69.7	51.2	65.1	+27.1%

4.2 Retention Across Model Pairs

In table 2, we evaluate the retention capabilities of the transferred cache from SwAT compared to if the model perceives the raw text directly. Concretely, for each sender–receiver pair, we calculate the score of the receiver solving the task (d, q) normally by itself (the denominator) and compare it with the score of the sender prefilling d and using SwAT to transfer its KV-cache to the receiver which only sees the raw value of q (where d, q are dataset specific and defined in section 4.1). Across the 10 model–pairs evaluated on 17 benchmarks, the lowest mean retention is $> 90\%$ even in the 'small to large' pairing where the large model has to solve the task using the smaller model's KV-cache compared to using its own KV-cache. In the 'large to small' pairs, the lowest mean retention is $> 100\%$, which is logically plausible as the receiver utilizes the larger model's KV-cache to solve the task instead of relying on its own (smaller model) KV-cache. Table 2 indicates that across our tested benchmarks, a receiver agent viewing the task and solving it through the SwAT translated KV-cache of the sender agent achieves, on average, performance comparable to that of the receiver agent viewing the task itself (in the 'small to large' case), or performance that even slightly exceeds that performance (in the 'large to small' case). Additional details are provided in Appendix C.

4.3 Cache Transfer versus Text Communication

We compare the effectiveness of using SwAT with AM against text-to-text communication (T2T) as described in section 2.1. We extensively compare the two methods across three model–pairs, five budgets, and seven benchmarks. We provide results averaged over the seven benchmarks in table 3 (arithmetic mean for score, geometric mean for other metrics). We additionally report the overall mean across all budgets and datasets, the bootstrapped standard error resampling datasets and their samples, as well as reporting when SwAT+AM is significantly better than T2T (95% bootstrap CI). In addition, we report individual results not averaged per dataset in fig. 3. Additional details are provided in Appendix D.

4.4 Zero-Shot Capabilities

SwAT was only trained on English text, yet we uncover additional capabilities transferred to the receiver model outside of English text. Additional details and capabilities are provided in the supplementary material.

4.4.1 Transferring Vision to Text-Only Receivers. Despite SwAT only training on English text, we want to know whether SwAT can provide the receiver with something it never had: vision. We use Gemma 3 VLMs as senders and text-only LLMs as receivers. We take multiple VLM benchmarks and sample image–question pairs (I, q) from each. The sender looks at the image and writes a short caption P. In T2T, the receiver reads P as text and attempts to answer q. With SwAT, the sender prefills $I \| P$ (where the VLM enriches the KV-cache of P with visual information I) and we take the KV-cache of P and transfer it using SwAT to the receiver which then attempts to answer q. We report these results in table 4. SwAT beats T2T and even beats the sender VLM answering by itself. The trend against T2T holds for many more benchmarks while the trend against the vanilla sender VLM does not hold for some benchmarks especially ones that require detailed image information. Additional details are provided in Appendix G.

4.4.2 Transferring Specialized Knowledge. We next ask whether SwAT can transfer from the sender specialized knowledge that the receiver lacks. Specifically we focus on languages as they occupy their own tokens that aren't in SwAT's English-only training data. Qwen 3 4B is much stronger in Chinese than Llama 3 8B. On all 67 subjects of CMMLU it scores approximately 20 points more than Llama. We let Qwen read each question and pass its cache to Llama through SwAT. We report these results in table 5. This makes Llama close three quarters of the gap to Qwen.

5 Comparisons

Compute. A self-evident but important aspect of SwAT is that it is designed to require much less compute (both in terms of FLOPs and wall-clock time) to transfer the KV-cache from sender to receiver than it would take the receiver to prefill the original text. In table 6

Table 6: Receiver re-prefill vs SwaT FLOP count and latency.

	Qwen 3 4B → Llama 3 8B						Llama 3 8B → Qwen 3 14B					
	FLOPs (TFLOPs)			Latency (ms)			FLOPs (TFLOPs)			Latency (ms)		
Seq Len	Re-prefill	SwaT	Ratio	Re-prefill	SwaT	Speedup	Re-prefill	SwaT	Ratio	Re-prefill	SwaT	Speedup
64	0.89	0.04	×24.7	26.0	2.1	×12.3	1.69	0.04	×45.3	43.5	2.2	×20.2
256	3.59	0.15	×24.4	31.7	2.6	×12.0	6.79	0.15	×44.7	47.8	2.7	×18.0
1K	14.57	0.63	×23.2	52.9	5.7	× 9.3	27.49	0.65	×42.5	95.8	5.7	×16.7
4K	61.57	3.13	×19.7	209.9	21.7	× 9.7	115.11	3.20	×35.9	394.1	22.3	×17.7
16K	299.07	22.40	×13.3	1,054.5	131.4	× 8.0	542.89	22.71	×23.9	1,912.0	134.9	×14.2
32K	738.88	71.19	×10.4	2,763.2	391.6	× 7.1	1,305.68	71.81	×18.2	4,799.9	399.0	×12.0
64K	2,040.71	247.94	× 8.2	7,966.4	1,296.3	× 6.1	3,490.98	249.18	×14.0	13,465.7	1,314.9	×10.2

Table 7: Mean (%, F1) over 17 benchmarks (± SE over 3 training seeds). Other methods are provided cross-tokenizer-aware alignment to function across model families. SwaT exceeds others by ≥20 points in the cross-generation/family cases.

Sender → Receiver	Mode	Closed form [13]	Dense latent [4]	SwaT (ours)
Qwen 3 4B → Qwen 3 8B	Same Family	59.9 ±0.2	**73.5** ±0.1	72.3 ±0.2
Qwen 3 4B → Qwen 2 7B	Cross Gen.	41.0 ±0.5	47.3 ±0.2	**68.1** ±0.1
Qwen 3 4B → Llama 3 8B	Cross Family	30.5 ±2.2	43.5 ±0.3	**63.5** ±0.1

we compare this for sequence lengths ranging from 64 to 64,000. We point out that in the worst case across all sequence lengths, the FLOP count of SwaT is at least ×8 and ×14 less than what is required for a re-prefill by the receiver across the two pairs. In terms of latency, SwaT is approximately ×10 and ×16 faster for short context lengths (< 1K tokens) across the two pairs. For extremely long context lengths up to 64K tokens, the two pairs are still ×6 and ×10 faster respectively. We note that the re-prefill FLOP count and latency significantly increase when going from the 8B receiver to the 14B receiver while SwaT costs roughly the same across the two.

Other methods. Although other KV-cache transfer methods do not target the same capabilities as SwaT (as highlighted in table 1), we implement two recent cross-model and cross-context KV-cache transfer methods (referred to as 'Closed Form' [13] and 'Dense Latent' [4]). Since neither method was designed for cross-family KV-cache transfer, we augment both with cross-tokenizer-aware functionality to work across tokenizers. We report the results of the two methods compared to SwaT on three model–pairs averaged across 17 benchmarks in table 7. Each method and model–pair was trained and evaluated three times with standard error displayed. We note that [13] was given different calibration data across the three seeds to evaluate variance; the two stochastic gradient descent methods were simply given different seeds.

6 Discussion

Limitations. SwaT has several limitations. First, each sender–receiver pair requires a trained mapper. For a multi-agent system with n families, we would need $n(n-1)$ trained mappers, which quickly becomes infeasible. Future work might investigate having a shared latent space to train mappers in and out of it for each agent which would total to only $2n$ mappers being needed. Another limitation is our focus on individual message handoffs instead of repeated transfers or aggregating multiple messages together, both of which introduce challenges as KV-cache degrades and isn't as composable as text. Future work might integrate SwaT with recent KV-cache composition techniques such as Yang et al. [37].

Ethical considerations. SwaT aims to improve the capabilities of multi-agent systems. The current implementation requires the sender's text to be available; thus, text-based monitoring is possible. However, the transferred cache may contain information beyond what is explicit in the text, which highlights the importance of future work investigating the auditability and safety of the benefits of KV-cache transfer while still utilizing the benefits.

7 Related Work

Text communication in multi-agent systems. LLM multi-agent systems have progressed rapidly solving novel challenges [3, 16, 18, 35, 38]. In these systems, communication is primarily in natural language and can occur in various ways to improve the performance of the system such as through debate [2, 3, 7], ensembling [16, 38], and even fine-tuning on communication [29].

Cross-model KV-cache transfer. Many recent works investigate whether KV-cache can be transferred between models. Certain recent works re-use KV-cache within variations of the same model [22, 33]. Other works extend KV-cache transfer across different models within the same family without requiring the receiver to prefill [4, 6, 13, 26]. Other recent works translate KV-cache across families but require the receiver to prefill text [8, 20, 28, 34].

8 Conclusion

We introduce SwaT, a cross-family KV-cache transfer method between LLMs without receiver prefill. Evaluation across many model–pairs and datasets shows the effectiveness of SwaT as a communication protocol in multi-agent systems, achieving more accurate and faster communication compared to text-to-text communication.

References

[1] Elie Bakouch, Loubna Ben Allal, Anton Lozhkov, Nouamane Tazi, Lewis Tunstall, Carlos Miguel Patiño, Edward Beeching, Aymeric Roucher, Aksel Joonas Reedi, Quentin Gallouédec, Kashif Rasul, Nathan Habib, Clémentine Fourrier, Hynek Kydlicek, Guilherme Penedo, Hugo Larcher, Mathieu Morlon, Vaibhav Srivastav, Joshua Lochner, Xuan-Son Nguyen, Colin Raffel, Leandro von Werra, and Thomas Wolf. 2025. SmolLM3: smol, multilingual, long-context reasoner. https://huggingface.co/blog/smollm3.

[2] Chi-Min Chan, Weize Chen, Yusheng Su, Jianxuan Yu, Wei Xue, Shanghang Zhang, Jie Fu, and Zhiyuan Liu. 2024. Chateval: Towards better llm-based evaluators through multi-agent debate. In *International conference on learning representations*, Vol. 2024. 9079–9093.

[3] Justin Chen, Swarnadeep Saha, and Mohit Bansal. 2024. ReConcile: Round-Table Conference Improves Reasoning via Consensus among Diverse LLMs. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Lun-Wei Ku, Andre Martins, and Vivek Srikumar (Eds.). Association for Computational Linguistics, Bangkok, Thailand, 7066–7085. doi:10.18653/v1/2024.acl-long.381

[4] Siyi Chen, Xiaoyan Zhang, Meng Wu, Jonathan Tremblay, Valts Blukis, Stan Birchfield, Rene Vidal, Alvaro Velasquez, Sijia Liu, and Qing Qu. 2026. See What I See, Know What I Think: Dense Latent Communication Across Heterogeneous Agents. arXiv:2606.13594 [cs.MA] https://arxiv.org/abs/2606.13594

[5] Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, et al. 2024. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors, 2023. *URL https://arxiv. org/abs/2308.10848* 6 (2024).

[6] Lucio M. Dery, Zohar Yahav, Henry Prior, Qixuan Feng, Jiajun Shen, and Arthur Szlam. 2026. Latent Space Communication via K-V Cache Alignment. arXiv:2601.06123 [cs.LG] https://arxiv.org/abs/2601.06123

[7] Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. 2023. Improving factuality and reasoning in language models through multiagent debate, 2023. *URL https://arxiv. org/abs/2305.14325* 3 (2023).

[8] Tianyu Fu, Zihan Min, Hanling Zhang, Jichao Yan, Guohao Dai, Wanli Ouyang, and Yu Wang. 2026. Cache-to-Cache: Direct Semantic Communication Between Large Language Models. In *International Conference on Learning Representations*, C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (Eds.), Vol. 2026. 43130–43158. https://proceedings.iclr.cc/paper_files/paper/2026/file/474ada926b331d78f06d95e8913111cc-Paper-Conference.pdf

[9] Ali Essam Ghareeb, Benjamin Chang, Ludovico Mitchener, Angela Yiu, Caralyn J Szostkiewicz, Dmytro Shved, Gavin J Gyimesi, Jon M Laurent, Samantha M Wright, Muhammed T Razzak, et al. 2026. A multi-agent system for automating scientific discovery. *Nature* (2026), 1–3.

[10] IBM Granite Team. 2024. Granite 3.0 Language Models. https://github.com/ibm-granite/granite-3.0-language-models/

[11] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. 2024. The Llama 3 Herd of Models. arXiv:2407.21783 [cs.AI] https://arxiv.org/abs/2407.21783

[12] Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. 2024. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769* (2024).

[13] Taekyung Heo, Rasoul Shafipour, Ritchie Zhao, Maximilian Golub, Mohammad Mahdi Kamani, Ritika Borkar, Makesh Tarun Chandran, Pantea Zardoshti, and Bita Darvish Rouhani. 2026. Cross-Model KV Cache Transfer in LLM Families: A Closed-Form Linear Mapping for Prefill Reuse. arXiv:2608.03893 [cs.LG] https://arxiv.org/abs/2608.03893

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

[15] Minyoung Huh, Brian Cheung, Tongzhou Wang, and Phillip Isola. 2024. The platonic representation hypothesis. *arXiv preprint arXiv:2405.07987* (2024).

[16] Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. 2023. LLM-Blender: Ensembling Large Language Models with Pairwise Ranking and Generative Fusion. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, Toronto, Canada, 14165–14178. doi:10.18653/v1/2023.acl-long.792

[17] Philippe Laban, Hiroaki Hayashi, Yingbo Zhou, and Jennifer Neville. 2026. Llms get lost in multi-turn conversation. In *International Conference on Learning Representations*, Vol. 2026. 54738–54778.

[18] Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. CAMEL: Communicative Agents for "Mind" Exploration of Large Language Model Society. arXiv:2303.17760 [cs.AI] https://arxiv.org/abs/2303.17760

[19] Alexander H. Liu, Kartik Khandelwal, Sandeep Subramanian, Victor Jouault, Abhinav Rastogi, Adrien Sadé, Alan Jeffares, Albert Jiang, Alexandre Cahill, Alexandre Gavaudan, et al. 2026. Ministral 3. arXiv:2601.08584 [cs.CL] https://arxiv.org/abs/2601.08584

[20] Jiyao Liu, Qi Zhang, Yaoyi Jia, Ziwen Kan, and Song Wang. 2026. Dual-Cache Latent Space Communication between Heterogeneous Language Models. arXiv:2608.20617 [cs.AI] https://arxiv.org/abs/2608.20617

[21] Liang Liu, Dong Zhang, Shoushan Li, Guodong Zhou, and Erik Cambria. 2024. Two heads are better than one: Zero-shot cognitive reasoning via multi-llm knowledge fusion. In *Proceedings of the 33rd ACM international conference on information and knowledge management*. 1462–1472.

[22] 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. 2025. DroidSpeak: KV Cache Sharing for Cross-LLM Communication and Multi-LLM Serving. arXiv:2411.02820 [cs.MA] https://arxiv.org/abs/2411.02820

[23] Bhargav Chickmagalur Nanjundappa and Spandan Maaheshwari. 2025. Context Branching for LLM Conversations: A Version Control Approach to Exploratory Programming. arXiv:2512.13914 [cs.SE] https://arxiv.org/abs/2512.13914

[24] Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024. ChatDev: Communicative Agents for Software Development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Lun-Wei Ku, Andre Martins, and Vivek Srikumar (Eds.). Association for Computational Linguistics, Bangkok, Thailand, 15174–15186. doi:10.18653/v1/2024.acl-long.810

[25] Chen Qian, Zihao Xie, YiFei Wang, Wei Liu, Kunlun Zhu, Hanchen Xia, Yufan Dang, Zhuoyun Du, Weize Chen, Cheng Yang, et al. 2025. Scaling large language model-based multi-agent collaboration. In *International Conference on Learning Representations*, Vol. 2025. 41488–41505.

[26] Xingyu Qu, Siyuan Lu, Zhiyu Chen, Sheng Wang, and Tao Lin. 2026. CacheBridge: Efficient Cross-Model KV Cache Transfer. arXiv:2609.00891 [cs.AI] https://arxiv.org/abs/2609.00891

[27] Sumedh Rasal. 2024. LLM Harmony: Multi-Agent Communication for Problem Solving. arXiv:2401.01312 [cs.MA] https://arxiv.org/abs/2401.01312

[28] Maximillian Rossi, Prajwal Raghunath, and Eugene Wu. 2026. Latent Cache Flow: Model-to-Model Communication Without Text. arXiv:2605.22863 [cs.LG] https://arxiv.org/abs/2605.22863

[29] Vighnesh Subramaniam, Yilun Du, Joshua B Tenenbaum, Antonio Torralba, Shuang Li, and Igor Mordatch. 2025. Multiagent finetuning: Self improvement with diverse reasoning chains. In *International Conference on Learning Representations*, Vol. 2025. 10840–10862.

[30] Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. 2025. Gemma 3 Technical Report. arXiv:2503.19786 [cs.CL] https://arxiv.org/abs/2503.19786

[31] Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. 2024. Gemma 2: Improving Open Language Models at a Practical Size. arXiv:2408.00118 [cs.CL] https://arxiv.org/abs/2408.00118

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

[33] Sunghyeon Woo, Jaeeun Kil, Hoseung Kim, Minsub Kim, Joonghoon Kim, Ahreum Seo, Sungjae Lee, Minjung Jo, Jiwon Ryu, Baeseong Park, Se Jung Kwon, and Dongsoo Lee. 2026. ICaRus: Identical Cache Reuse for Efficient Multi-Model Inference. In *International Conference on Learning Representations*. https://mlanthology.org/iclr/2026/woo2026iclr-icarus/

[34] Jin woo Lee, Minkyung Song, Junghyun Oh, Seunghoon Han, Gwangseon Jang, Soyoung Park, and Sungsu Lim. 2026. KV Cache Translation across Heterogeneous Large Language Models. arXiv:2607.28979 [cs.CL] https://arxiv.org/abs/2607.28979

[35] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. 2023. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation. arXiv:2308.08155 [cs.AI] https://arxiv.org/abs/2308.08155

[36] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. 2025. Qwen3 Technical Report. arXiv:2505.09388 [cs.CL] https://arxiv.org/abs/2505.09388

[37] Jingbo Yang, Bairu Hou, Wei Wei, Yujia Bao, and Shiyu Chang. 2026. Kvlink: Accelerating large language models via efficient kv cache reuse. *Advances in Neural Information Processing Systems* 38 (2026), 133797–133824.

[38] Zhangyue Yin, Qiushi Sun, Cheng Chang, Qipeng Guo, Junqi Dai, Xuanjing Huang, and Xipeng Qiu. 2023. Exchange-of-Thought: Enhancing Large Language Model Capabilities through Cross-Model Communication. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, Singapore, 15135–15153. doi:10.18653/v1/2023.emnlp-main.936

[39] Yanwei Yue, Guibin Zhang, Boyang Liu, Guancheng Wan, Kun Wang, Dawei Cheng, and Yiyan Qi. 2025. MasRouter: Learning to Route LLMs for Multi-Agent Systems. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (Eds.). Association for Computational Linguistics, Vienna, Austria, 15549–15572. doi:10.18653/v1/2025.acl-long.757

[40] Jiaru Zou, Ruizhong Qiu, Gaotang Li, Xiyuan Yang, Katherine Tieu, Pan Lu, Ke Shen, Hanghang Tong, Yejin Choi, Jingrui He, et al. 2025. Latent collaboration in multi-agent systems. *arXiv preprint arXiv:2511.20639* (2025).

[41] Adam Zweiger, Xinghong Fu, Han Guo, and Yoon Kim. 2026. Fast KV Compaction via Attention Matching. arXiv:2602.16284 [cs.LG] https://arxiv.org/abs/2602.16284

Appendix

A Code

We provide the code repository at:

https://github.com/Ar-Kareem/SwaT

B AI use

AI-assisted technologies were used for polishing the text, helping with LaTeX format, and for writing code and plotting scripts. AI was not used in the creation of hypotheses or methodologies.

C Additional retention data

In tables 8 and 9 we show the breakdown of all pairs on all datasets of the aggregate table 2. We note that MATH-500 is a highly out-of-distribution task that requires answers formatted as LaTeX expressions while the training data explicitly trains on GSM8K where final answers are numerical values.

Table 8: In-domain (9): per-benchmark receiver score with its own context, SwaT score, and zero-shot retention (italic; SwaT over receiver). Scores in percent. HSwag = HellaSwag; WGrande = WinoGrande; Hotpot = HotpotQA.

		Short					Context			
		ARC-C	HSwag	WGrande	MMLU	GSM8K	QuALITY	QASPER	Hotpot	MuSiQue
Pair	Row	Acc	Acc	Acc	Acc	Acc	Acc	F1	F1	F1
Small → Large										
Qwen 3 4B	Receiver	80.6	77.2	59.2	62.2	75.2	64.1	49.0	67.4	34.6
→ Llama 3 8B	SwaT	85.2	83.6	58.8	62.0	67.6	64.2	51.1	70.6	42.6
	Retention (%)	*105.7*	*108.3*	*99.3*	*99.7*	*89.9*	*100.2*	*104.4*	*104.8*	*123.4*
Llama 3 8B	Receiver	92.6	91.4	69.4	73.2	89.8	80.1	49.1	73.1	43.4
→ Qwen 3 14B	SwaT	78.6	80.6	65.6	60.0	88.6	78.7	56.3	78.2	55.0
	Retention (%)	*84.9*	*88.2*	*94.5*	*82.0*	*98.7*	*98.2*	*114.7*	*106.9*	*126.8*
Ministral 3 8B	Receiver	80.0	80.6	60.6	66.4	79.8	72.9	49.3	71.3	38.6
→ Llama 3.2 11B	SwaT	88.8	86.6	64.2	68.4	74.8	71.0	49.4	71.9	48.0
	Retention (%)	*111.0*	*107.4*	*105.9*	*103.0*	*93.7*	*97.5*	*100.3*	*100.9*	*124.5*
Gemma 3 4B	Receiver	80.4	77.2	62.6	61.0	77.4	67.2	33.3	47.9	24.4
→ Granite 3.3 8B	SwaT	73.4	77.0	63.2	52.0	77.4	62.9	53.7	67.3	43.2
	Retention (%)	*91.3*	*99.7*	*101.0*	*85.2*	*100.0*	*93.6*	*161.6*	*140.6*	*177.2*
Qwen 3 4B	Receiver	90.6	88.2	69.4	67.8	80.0	75.1	45.7	73.4	46.5
→ Gemma 2 9B	SwaT	85.2	85.0	63.0	64.2	75.2	69.4	49.6	71.8	49.4
	Retention (%)	*94.0*	*96.4*	*90.8*	*94.7*	*94.0*	*92.4*	*108.5*	*97.8*	*106.3*
Gemma 2 9B	Receiver	92.0	84.2	66.8	72.4	82.8	80.0	42.2	75.1	51.3
→ Ministral 3 14B	SwaT	88.2	91.8	70.4	66.8	88.4	76.3	51.5	75.4	55.3
	Retention (%)	*95.9*	*109.0*	*105.4*	*92.3*	*106.8*	*95.4*	*121.9*	*100.4*	*107.7*
Gemma 3 4B	Receiver	80.6	77.2	59.2	62.2	75.2	64.1	49.0	67.4	34.6
→ Llama 3 8B	SwaT	70.2	76.0	54.4	51.0	69.0	65.1	50.7	69.6	39.3
	Retention (%)	*87.1*	*98.4*	*91.9*	*82.0*	*91.8*	*101.6*	*103.6*	*103.4*	*113.8*
Large → Small										
Gemma 3 12B	Receiver	87.4	84.6	59.8	63.6	85.0	70.2	47.0	64.1	31.8
→ Qwen 3 4B	SwaT	87.8	90.0	66.4	66.8	82.6	70.3	53.2	71.4	43.6
	Retention (%)	*100.5*	*106.4*	*111.0*	*105.0*	*97.2*	*100.1*	*113.3*	*111.4*	*136.8*
Qwen 3 32B	Receiver	80.6	77.2	59.2	62.2	75.2	64.1	49.0	67.4	34.6
→ Llama 3 8B	SwaT	94.8	92.0	68.2	77.0	66.8	66.7	48.4	67.3	33.4
	Retention (%)	*117.6*	*119.2*	*115.2*	*123.8*	*88.8*	*104.0*	*98.8*	*99.9*	*96.6*
Granite 3.3 8B	Receiver	78.2	71.8	54.4	53.6	73.8	60.9	28.4	30.7	11.5
→ SmolLM3 3B	SwaT	82.8	85.6	68.2	59.2	74.6	63.3	47.8	65.9	43.3
	Retention (%)	*105.9*	*119.2*	*125.4*	*110.4*	*101.1*	*103.9*	*168.4*	*214.6*	*377.8*

Table 9: OOD (8): per-benchmark receiver score with its own context, SwAT score, and zero-shot retention (italic; SwAT over receiver). Scores in percent.

		Short			Context				
		MedQA	OpenBookQA	MATH-500	BoolQ	RACE	LongHealth	SQuAD	NarrQA
Pair	Row	Acc	Acc	Acc	Acc	Acc	Acc	F1	F1
Small → Large									
Qwen 3 4B	Receiver	60.0	76.0	28.4	83.7	81.3	66.8	88.9	74.2
→ Llama 3 8B	SwAT	54.8	79.2	20.0	79.8	76.4	57.0	87.5	73.4
	Retention (%)	*91.3*	*104.2*	*70.4*	*95.3*	*94.0*	*85.4*	*98.4*	*98.9*
Llama 3 8B	Receiver	64.4	90.2	76.6	89.1	89.4	72.0	86.9	72.7
→ Qwen 3 14B	SwAT	54.4	75.6	69.8	88.9	88.0	71.0	90.5	74.8
	Retention (%)	*84.5*	*83.8*	*91.1*	*99.7*	*98.4*	*98.6*	*104.1*	*102.9*
Ministral 3 8B	Receiver	59.8	81.6	49.0	85.0	83.4	71.5	88.3	75.7
→ Llama 3.2 11B	SwAT	59.6	83.0	33.2	82.4	78.3	62.7	89.2	73.5
	Retention (%)	*99.7*	*101.7*	*67.8*	*96.9*	*93.9*	*87.8*	*101.0*	*97.1*
Gemma 3 4B	Receiver	51.6	78.4	61.0	81.9	79.0	60.2	68.5	50.0
→ Granite 3.3 8B	SwAT	44.0	72.2	41.8	82.1	77.2	52.5	86.1	68.5
	Retention (%)	*85.3*	*92.1*	*68.5*	*100.2*	*97.7*	*87.1*	*125.7*	*137.2*
Qwen 3 4B	Receiver	57.2	88.2	51.4	89.1	87.6	68.2	88.6	78.7
→ Gemma 2 9B	SwAT	52.4	79.8	35.0	86.3	83.8	62.0	88.7	72.7
	Retention (%)	*91.6*	*90.5*	*68.1*	*96.8*	*95.7*	*90.8*	*100.2*	*92.3*
Gemma 2 9B	Receiver	62.4	86.2	55.2	85.6	87.7	73.0	83.6	71.6
→ Ministral 3 14B	SwAT	56.4	86.0	67.6	82.2	84.6	68.2	90.7	75.4
	Retention (%)	*90.4*	*99.8*	*122.5*	*96.0*	*96.5*	*93.5*	*108.6*	*105.2*
Gemma 3 4B	Receiver	60.0	76.0	28.4	83.7	81.3	66.8	88.9	74.2
→ Llama 3 8B	SwAT	43.0	69.8	16.0	80.0	77.5	56.8	87.8	73.7
	Retention (%)	*71.7*	*91.8*	*56.3*	*95.6*	*95.2*	*85.0*	*98.7*	*99.2*
Large → Small									
Gemma 3 12B	Receiver	55.0	80.6	71.0	84.3	83.9	68.0	80.7	68.0
→ Qwen 3 4B	SwAT	54.4	82.2	64.2	85.2	82.5	63.7	88.0	69.8
	Retention (%)	*98.9*	*102.0*	*90.4*	*101.2*	*98.3*	*93.7*	*109.0*	*102.6*
Qwen 3 32B	Receiver	60.0	76.0	28.4	83.7	81.3	66.8	88.9	74.2
→ Llama 3 8B	SwAT	71.0	92.4	19.2	81.9	78.3	61.5	86.8	71.6
	Retention (%)	*118.3*	*121.6*	*67.6*	*97.8*	*96.2*	*92.1*	*97.7*	*96.4*
Granite 3.3 8B	Receiver	29.6	74.6	61.8	83.5	74.1	39.2	50.6	44.0
→ SmolLM3 3B	SwAT	48.2	77.4	45.4	81.3	75.1	57.0	84.0	69.6
	Retention (%)	*162.8*	*103.8*	*73.5*	*97.4*	*101.3*	*145.2*	*166.0*	*158.3*

D Additional SwaT+AM details

We provide a comparison of SwaT+AM vs T2T in all seven datasets in tables 10 to 16. We also provide the mean over the datasets in table 17 where we report both the geometric mean of all metrics and a row for the arithmetic mean over the score.

Table 10: Experiment 2, HotpotQA. T2T: sender summarizes to receiver. SwaT+AM: the sender's cache rendered and compressed with Attention Matching. Budgets are determined by the sender agent in T2T given one of five summary prompts and sorted from lowest to highest per pair. Change: score difference in points, relative token difference, and speedup.

		Qwen3-4B → Llama3-8B			Llama3-8B → Qwen3-14B			Gemma3-12B → Qwen3-4B		
Metric	Budget	T2T	SwaT	Change	T2T	SwaT	Change	T2T	SwaT	Change
	Lowest	*≈10× compression ($P_{\text{sum},1}$)*			*≈9× ($P_{\text{sum},1}$)*			*≈10× ($P_{\text{sum},1}$)*		
Score		44.6	**62.8**	+18.2	47.2	**67.9**	+20.6	37.9	**53.8**	+15.9
Tokens		144	**132**	−8.2%	170	**154**	−9.6%	151	**138**	−8.3%
$\mathcal{S}$ GPU (ms)		4,731	**997**	×4.7	3,687	**1,438**	×2.6	9,452	**1,336**	×7.1
$\mathcal{R}$ TTFT (ms)		9.8	**3.6**	×2.7	21.8	**6.5**	×3.4	7.2	**2.9**	×2.5
	Lower	*≈7× compression ($P_{\text{sum},2}$)*			*≈6× ($P_{\text{sum},2}$)*			*≈7× ($P_{\text{sum},2}$)*		
Score		47.5	**62.7**	+15.2	49.8	**68.8**	+19.0	37.5	**59.5**	+21.9
Tokens		207	**188**	−9.3%	235	**231**	−1.7%	209	**198**	−5.3%
$\mathcal{S}$ GPU (ms)		6,631	**1,090**	×6.1	5,162	**1,637**	×3.2	13,511	**1,471**	×9.2
$\mathcal{R}$ TTFT (ms)		12.9	**3.6**	×3.5	27.3	**6.5**	×4.2	9.3	**3.0**	×3.1
	Low	*≈5× compression ($P_{\text{sum},3}$)*			*≈5× ($P_{\text{sum},3}$)*			*≈5× ($P_{\text{sum},3}$)*		
Score		51.8	**66.5**	+14.7	55.9	**71.4**	+15.5	45.3	**62.4**	+17.1
Tokens		297	**263**	−11.2%	335	**277**	−17.4%	344	**277**	−19.5%
$\mathcal{S}$ GPU (ms)		9,132	**1,315**	×6.9	7,403	**1,790**	×4.1	22,165	**1,821**	×12.2
$\mathcal{R}$ TTFT (ms)		17.2	**3.7**	×4.6	36.6	**6.6**	×5.6	13.7	**3.0**	×4.5
	Medium-low	*≈3× compression ($P_{\text{sum},5}$)*			*≈3× ($P_{\text{sum},5}$)*			*≈2× ($P_{\text{sum},4}$)*		
Score		56.5	**67.4**	+10.9	59.4	**74.9**	+15.5	59.2	**68.5**	+9.3
Tokens		628	**439**	−30.0%	499	**461**	−7.6%	767	**692**	−9.7%
$\mathcal{S}$ GPU (ms)		18,038	**2,325**	×7.8	10,604	**2,805**	×3.8	47,056	**4,561**	×10.3
$\mathcal{R}$ TTFT (ms)		30.3	**3.8**	×8.0	50.0	**6.7**	×7.4	27.2	**3.3**	×8.4
	Medium	*≈2× compression ($P_{\text{sum},4}$)*			*≈3× ($P_{\text{sum},4}$)*			*≈2× ($P_{\text{sum},5}$)*		
Score		61.2	**67.9**	+6.7	63.4	**74.9**	+11.4	55.1	**68.5**	+13.4
Tokens		789	**659**	−16.5%	542	**461**	−14.9%	921	**692**	−24.8%
$\mathcal{S}$ GPU (ms)		23,356	**3,847**	×6.1	12,119	**2,805**	×4.3	58,826	**4,561**	×12.9
$\mathcal{R}$ TTFT (ms)		37.2	**3.9**	×9.5	54.6	**6.7**	×8.1	33.3	**3.3**	×10.2

Table 11: Experiment 2, LongHealth. T2T: sender summarizes to receiver. SwaT+AM: the sender's cache rendered and compressed with Attention Matching. Budgets are determined by the sender agent in T2T given one of five summary prompts and sorted from lowest to highest per pair. Change: score difference in points, relative token difference, and speedup.

		Qwen3-4B → Llama3-8B			Llama3-8B → Qwen3-14B			Gemma3-12B → Qwen3-4B		
Metric	Budget	T2T	SwaT	Change	T2T	SwaT	Change	T2T	SwaT	Change
	Lowest	*≈58× compression ($P_{\text{sum},1}$)*			*≈26× ($P_{\text{sum},3}$)*			*≈39× ($P_{\text{sum},1}$)*		
Score		28.7	**37.2**	+8.5	39.0	**47.8**	+8.8	30.8	**40.5**	+9.8
Tokens		130	**129**	−0.5%	308	**303**	−1.5%	203	**202**	−0.4%
$\mathcal{S}$ GPU (ms)		4,299	**3,410**	×1.3	7,481	**4,840**	×1.5	13,791	**5,567**	×2.5
$\mathcal{R}$ TTFT (ms)		11.7	**5.9**	×2.0	38.0	**12.0**	×3.2	11.2	**5.1**	×2.2
	Lower	*≈34× compression ($P_{\text{sum},2}$)*			*≈22× ($P_{\text{sum},1}$)*			*≈25× ($P_{\text{sum},2}$)*		
Score		31.0	**38.5**	+7.5	35.5	**49.2**	+13.8	32.0	**45.2**	+13.2
Tokens		224	**221**	−1.7%	363	**358**	−1.3%	326	**315**	−3.3%
$\mathcal{S}$ GPU (ms)		7,272	**3,680**	×2.0	8,615	**5,004**	×1.7	22,103	**5,975**	×3.7
$\mathcal{R}$ TTFT (ms)		16.4	**6.0**	×2.7	42.9	**12.1**	×3.6	15.5	**5.2**	×3.0
	Low	*≈16× compression ($P_{\text{sum},3}$)*			*≈22× ($P_{\text{sum},2}$)*			*≈10× ($P_{\text{sum},3}$)*		
Score		36.0	**44.8**	+8.8	38.8	**49.2**	+10.5	40.0	**54.0**	+14.0
Tokens		475	**469**	−1.3%	370	**358**	−3.1%	798	**788**	−1.3%
$\mathcal{S}$ GPU (ms)		14,697	**4,465**	×3.3	8,876	**5,004**	×1.8	51,790	**7,802**	×6.6
$\mathcal{R}$ TTFT (ms)		27.3	**6.2**	×4.4	43.8	**12.1**	×3.6	31.5	**5.6**	×5.6
	Medium-low	*≈7× compression ($P_{\text{sum},4}$)*			*≈18× ($P_{\text{sum},5}$)*			*≈6× ($P_{\text{sum},5}$)*		
Score		43.5	**53.5**	+10.0	37.2	**48.8**	+11.5	49.0	**58.5**	+9.5
Tokens		1,194	**1,071**	−10.3%	439	**438**	−0.4%	1,426	**1,313**	−7.9%
$\mathcal{S}$ GPU (ms)		36,800	**6,369**	×5.8	10,281	**5,391**	×1.9	90,082	**9,704**	×9.3
$\mathcal{R}$ TTFT (ms)		59.9	**6.6**	×9.0	48.5	**12.2**	×4.0	52.4	**6.0**	×8.7
	Medium	*≈6× compression ($P_{\text{sum},5}$)*			*≈15× ($P_{\text{sum},4}$)*			*≈6× ($P_{\text{sum},4}$)*		
Score		45.0	**55.0**	+10.0	40.2	**51.7**	+11.5	49.0	**58.5**	+9.5
Tokens		1,323	**1,250**	−5.5%	542	**525**	−3.1%	1,469	**1,313**	−10.6%
$\mathcal{S}$ GPU (ms)		40,319	**7,095**	×5.7	12,653	**5,640**	×2.2	93,712	**9,704**	×9.7
$\mathcal{R}$ TTFT (ms)		66.4	**6.7**	×9.9	58.5	**12.3**	×4.8	54.1	**6.0**	×9.0

Table 12: Experiment 2, MuSiQue. T2T: sender summarizes to receiver. SwaT+AM: the sender's cache rendered and compressed with Attention Matching. Budgets are determined by the sender agent in T2T given one of five summary prompts and sorted from lowest to highest per pair. Change: score difference in points, relative token difference, and speedup.

		Qwen3-4B → Llama3-8B			Llama3-8B → Qwen3-14B			Gemma3-12B → Qwen3-4B		
Metric	Budget	T2T	SwaT	Change	T2T	SwaT	Change	T2T	SwaT	Change
	Lowest	*≈14× compression ($P_{\text{sum},1}$)*			*≈10× ($P_{\text{sum},1}$)*			*≈15× ($P_{\text{sum},1}$)*		
Score		11.8	**24.0**	+12.2	11.8	**32.6**	+20.8	8.2	**20.0**	+11.7
Tokens		168	168	−0.2%	254	**248**	−2.3%	171	**166**	−3.4%
$\mathcal{S}$ GPU (ms)		5,287	**1,568**	×3.4	5,722	**2,290**	×2.5	10,607	**2,441**	×4.3
$\mathcal{R}$ TTFT (ms)		9.9	**3.7**	×2.7	27.0	**6.6**	×4.1	7.8	**2.9**	×2.6
	Lower	*≈8× compression ($P_{\text{sum},2}$)*			*≈8× ($P_{\text{sum},2}$)*			*≈11× ($P_{\text{sum},2}$)*		
Score		13.0	**31.5**	+18.5	15.1	**36.6**	+21.5	8.4	**21.0**	+12.7
Tokens		305	**294**	−3.5%	346	**310**	−10.4%	230	**226**	−1.9%
$\mathcal{S}$ GPU (ms)		8,877	**1,977**	×4.5	6,361	**2,549**	×2.5	14,849	**2,457**	×6.0
$\mathcal{R}$ TTFT (ms)		17.0	**3.7**	×4.6	29.9	**6.6**	×4.5	10.1	**3.0**	×3.4
	Low	*≈7× compression ($P_{\text{sum},3}$)*			*≈6× ($P_{\text{sum},3}$)*			*≈6× ($P_{\text{sum},3}$)*		
Score		15.0	**32.1**	+17.1	16.9	**39.8**	+22.9	13.2	**27.5**	+14.3
Tokens		377	**336**	−10.7%	431	**414**	−3.9%	456	**414**	−9.2%
$\mathcal{S}$ GPU (ms)		9,464	**2,070**	×4.6	9,174	**3,059**	×3.0	29,586	**3,199**	×9.2
$\mathcal{R}$ TTFT (ms)		19.0	**3.8**	×5.0	41.1	**6.8**	×6.1	17.2	**3.2**	×5.5
	Medium-low	*≈3× compression ($P_{\text{sum},5}$)*			*≈4× ($P_{\text{sum},5}$)*			*≈3× ($P_{\text{sum},4}$)*		
Score		14.7	**37.6**	+22.9	17.6	**45.7**	+28.1	19.1	**34.5**	+15.4
Tokens		872	**785**	−10.0%	622	**621**	−0.3%	1,000	**828**	−17.2%
$\mathcal{S}$ GPU (ms)		29,256	**4,519**	×6.5	13,224	**4,078**	×3.2	68,038	**5,578**	×12.2
$\mathcal{R}$ TTFT (ms)		46.3	**4.0**	×11.6	54.3	**6.9**	×7.8	38.5	**3.4**	×11.3
	Medium	*≈3× compression ($P_{\text{sum},4}$)*			*≈4× ($P_{\text{sum},4}$)*			*≈3× ($P_{\text{sum},5}$)*		
Score		23.4	**37.6**	+14.2	20.7	**45.7**	+25.0	18.7	**34.5**	+15.8
Tokens		1,143	**785**	−31.4%	635	**621**	−2.2%	1,098	**828**	−24.7%
$\mathcal{S}$ GPU (ms)		38,766	**4,519**	×8.6	14,258	**4,078**	×3.5	73,704	**5,578**	×13.2
$\mathcal{R}$ TTFT (ms)		57.9	**4.0**	×14.5	61.2	**6.9**	×8.8	41.2	**3.4**	×12.1

Table 13: Experiment 2, NarrativeQA. T2T: sender summarizes to receiver. SwAT+AM: the sender's cache rendered and compressed with Attention Matching. Budgets are determined by the sender agent in T2T given one of five summary prompts and sorted from lowest to highest per pair. Change: score difference in points, relative token difference, and speedup.

		Qwen3-4B → Llama3-8B			Llama3-8B → Qwen3-14B			Gemma3-12B → Qwen3-4B		
Metric	Budget	T2T	SwAT	Change	T2T	SwAT	Change	T2T	SwAT	Change
	Lowest	*≈7× compression ($P_{\text{sum},1}$)*			*≈7× ($P_{\text{sum},1}$)*			*≈9× ($P_{\text{sum},1}$)*		
Score		29.2	**65.6**	+36.4	29.8	**66.7**	+36.9	19.1	**53.1**	+34.0
Tokens		108	**103**	−4.9%	115	**103**	−10.7%	84	**80**	−4.3%
$\mathcal{S}$ GPU (ms)		3,286	**1,085**	×3.0	2,806	**1,023**	×2.7	5,590	**1,083**	×5.2
$\mathcal{R}$ TTFT (ms)		8.3	**3.0**	×2.8	16.8	**5.9**	×2.8	5.2	**2.6**	×2.0
	Lower	*≈5× compression ($P_{\text{sum},2}$)*			*≈5× ($P_{\text{sum},2}$)*			*≈7× ($P_{\text{sum},2}$)*		
Score		35.3	**66.9**	+31.6	34.2	**69.2**	+35.0	22.9	**56.6**	+33.6
Tokens		155	**144**	−7.3%	151	**144**	−4.3%	116	**103**	−11.1%
$\mathcal{S}$ GPU (ms)		4,632	**1,084**	×4.3	3,428	**1,214**	×2.8	7,500	**1,145**	×6.5
$\mathcal{R}$ TTFT (ms)		10.0	**3.0**	×3.3	18.8	**6.0**	×3.1	6.4	**2.6**	×2.5
	Low	*≈4× compression ($P_{\text{sum},3}$)*			*≈4× ($P_{\text{sum},3}$)*			*≈4× ($P_{\text{sum},3}$)*		
Score		43.2	**68.2**	+25.0	38.2	**70.0**	+31.8	29.0	**63.1**	+34.1
Tokens		226	**180**	−20.3%	203	**181**	−11.0%	200	**181**	−9.9%
$\mathcal{S}$ GPU (ms)		7,466	**1,324**	×5.6	5,085	**1,379**	×3.7	13,666	**1,457**	×9.4
$\mathcal{R}$ TTFT (ms)		14.6	**3.0**	×4.8	25.0	**6.0**	×4.2	9.5	**2.6**	×3.6
	Medium-low	*≈2× compression ($P_{\text{sum},4}$)*			*≈2× ($P_{\text{sum},4}$)*			*≈2× ($P_{\text{sum},4}$)*		
Score		56.5	**70.0**	+13.5	49.5	**71.6**	+22.2	46.5	**66.7**	+20.2
Tokens		486	**360**	−26.0%	401	**361**	−9.9%	536	**361**	−32.6%
$\mathcal{S}$ GPU (ms)		15,394	**2,722**	×5.7	10,329	**2,671**	×3.9	35,625	**2,554**	×13.9
$\mathcal{R}$ TTFT (ms)		26.7	**3.1**	×8.6	44.2	**6.2**	×7.1	21.0	**2.8**	×7.6
	Medium	*≈2× compression ($P_{\text{sum},5}$)*			*≈2× ($P_{\text{sum},5}$)*			*≈2× ($P_{\text{sum},5}$)*		
Score		55.7	**70.0**	+14.4	47.2	**71.6**	+24.4	40.7	**66.7**	+26.0
Tokens		537	**360**	−33.0%	416	**361**	−13.3%	693	**361**	−47.9%
$\mathcal{S}$ GPU (ms)		16,040	**2,722**	×5.9	10,028	**2,671**	×3.8	46,570	**2,554**	×18.2
$\mathcal{R}$ TTFT (ms)		27.5	**3.1**	×8.9	43.3	**6.2**	×7.0	26.9	**2.8**	×9.8

Table 14: Experiment 2, QASPER. T2T: sender summarizes to receiver. SwAT+AM: the sender's cache rendered and compressed with Attention Matching. Budgets are determined by the sender agent in T2T given one of five summary prompts and sorted from lowest to highest per pair. Change: score difference in points, relative token difference, and speedup.

		Qwen3-4B → Llama3-8B			Llama3-8B → Qwen3-14B			Gemma3-12B → Qwen3-4B		
Metric	Budget	T2T	SwAT	Change	T2T	SwAT	Change	T2T	SwAT	Change
	Lowest	*≈41× compression ($P_{\text{sum},1}$)*			*≈27× ($P_{\text{sum},1}$)*			*≈41× ($P_{\text{sum},1}$)*		
Score		21.7	**34.3**	+12.6	26.5	**36.3**	+9.8	23.0	**31.9**	+8.9
Tokens		106	106	+0.2%	165	**164**	−0.5%	108	108	−0.2%
$\mathcal{S}$ GPU (ms)		3,269	**2,125**	×1.5	3,850	**2,781**	×1.4	6,953	**3,337**	×2.1
$\mathcal{R}$ TTFT (ms)		8.0	**2.9**	×2.8	19.8	**5.5**	×3.6	5.4	**2.4**	×2.2
	Lower	*≈26× compression ($P_{\text{sum},2}$)*			*≈20× ($P_{\text{sum},2}$)*			*≈29× ($P_{\text{sum},2}$)*		
Score		26.5	**37.8**	+11.4	29.6	**38.2**	+8.6	25.8	**34.7**	+8.9
Tokens		167	167	−0.1%	231	**221**	−4.4%	**151**	152	+0.8%
$\mathcal{S}$ GPU (ms)		5,356	**2,352**	×2.3	5,524	**3,029**	×1.8	9,952	**3,496**	×2.8
$\mathcal{R}$ TTFT (ms)		11.2	**2.9**	×3.8	25.7	**5.5**	×4.6	7.2	**2.5**	×2.9
	Low	*≈16× compression ($P_{\text{sum},3}$)*			*≈16× ($P_{\text{sum},3}$)*			*≈15× ($P_{\text{sum},3}$)*		
Score		32.1	**42.1**	+10.0	30.5	**39.0**	+8.5	29.4	**39.7**	+10.4
Tokens		283	**271**	−4.0%	288	**276**	−4.0%	312	**295**	−5.5%
$\mathcal{S}$ GPU (ms)		8,225	**2,674**	×3.1	7,229	**3,277**	×2.2	19,936	**3,995**	×5.0
$\mathcal{R}$ TTFT (ms)		15.4	**3.0**	×5.1	32.2	**5.6**	×5.8	12.3	**2.6**	×4.7
	Medium-low	*≈7× compression ($P_{\text{sum},5}$)*			*≈11× ($P_{\text{sum},5}$)*			*≈6× ($P_{\text{sum},4}$)*		
Score		38.8	**49.5**	+10.7	32.2	**41.3**	+9.1	39.3	**46.8**	+7.5
Tokens		701	**620**	−11.5%	404	**402**	−0.6%	802	**737**	−8.1%
$\mathcal{S}$ GPU (ms)		19,936	**3,770**	×5.3	8,799	**3,685**	×2.4	52,999	**5,723**	×9.3
$\mathcal{R}$ TTFT (ms)		33.2	**3.3**	×10.1	37.6	**5.7**	×6.5	29.0	**3.0**	×9.6
	Medium	*≈6× compression ($P_{\text{sum},4}$)*			*≈11× ($P_{\text{sum},4}$)*			*≈5× ($P_{\text{sum},5}$)*		
Score		40.9	**49.9**	+9.1	33.9	**41.3**	+7.4	38.9	**47.8**	+8.9
Tokens		**721**	723	+0.4%	414	**402**	−2.8%	945	**884**	−6.4%
$\mathcal{S}$ GPU (ms)		21,283	**4,084**	×5.2	9,656	**3,685**	×2.6	60,182	**6,179**	×9.7
$\mathcal{R}$ TTFT (ms)		35.0	**3.4**	×10.4	41.0	**5.7**	×7.1	33.5	**3.1**	×10.8

Table 15: Experiment 2, QuALITY. T2T: sender summarizes to receiver. SwaT+AM: the sender's cache rendered and compressed with Attention Matching. Budgets are determined by the sender agent in T2T given one of five summary prompts and sorted from lowest to highest per pair. Change: score difference in points, relative token difference, and speedup.

		Qwen3-4B → Llama3-8B			Llama3-8B → Qwen3-14B			Gemma3-12B → Qwen3-4B		
Metric	Budget	T2T	SwaT	Change	T2T	SwaT	Change	T2T	SwaT	Change
	Lowest	*≈42× compression ($P_{\text{sum},1}$)*			*≈30× ($P_{\text{sum},1}$)*			*≈52× ($P_{\text{sum},1}$)*		
Score		51.6	**59.2**	+7.6	57.8	**71.8**	+13.9	53.6	**61.6**	+8.0
Tokens		132	**128**	−2.5%	184	**180**	−2.0%	106	**104**	−2.3%
$\mathcal{S}$ GPU (ms)		3,721	**1,476**	×2.5	4,174	**2,085**	×2.0	7,229	**2,208**	×3.3
$\mathcal{R}$ TTFT (ms)		11.8	**6.0**	×2.0	27.3	**11.6**	×2.3	8.2	**4.9**	×1.7
	Lower	*≈27× compression ($P_{\text{sum},2}$)*			*≈25× ($P_{\text{sum},2}$)*			*≈40× ($P_{\text{sum},2}$)*		
Score		53.3	**59.4**	+6.1	57.5	**73.1**	+15.5	54.6	**62.9**	+8.3
Tokens		204	**200**	−2.4%	218	**216**	−0.8%	137	**135**	−1.3%
$\mathcal{S}$ GPU (ms)		5,085	**1,613**	×3.2	4,903	**2,035**	×2.4	8,950	**2,281**	×3.9
$\mathcal{R}$ TTFT (ms)		13.7	**6.1**	×2.3	29.9	**11.7**	×2.6	9.1	**4.9**	×1.9
	Low	*≈18× compression ($P_{\text{sum},3}$)*			*≈22× ($P_{\text{sum},3}$)*			*≈21× ($P_{\text{sum},3}$)*		
Score		53.6	**61.2**	+7.7	58.8	**73.7**	+14.9	57.6	**64.6**	+7.0
Tokens		303	**299**	−1.1%	254	**246**	−3.5%	263	**257**	−2.3%
$\mathcal{S}$ GPU (ms)		7,699	**1,784**	×4.3	7,131	**2,088**	×3.4	19,700	**2,534**	×7.8
$\mathcal{R}$ TTFT (ms)		17.8	**6.1**	×2.9	38.3	**11.7**	×3.3	14.5	**5.0**	×2.9
	Medium-low	*≈11× compression ($P_{\text{sum},5}$)*			*≈17× ($P_{\text{sum},5}$)*			*≈8× ($P_{\text{sum},4}$)*		
Score		55.6	**62.7**	+7.1	60.2	**74.9**	+14.7	63.5	**67.8**	+4.3
Tokens		528	**490**	−7.2%	336	**318**	−5.3%	769	**675**	−12.2%
$\mathcal{S}$ GPU (ms)		15,439	**2,150**	×7.2	9,097	**2,251**	×4.0	48,189	**3,304**	×14.6
$\mathcal{R}$ TTFT (ms)		29.3	**6.3**	×4.7	45.5	**11.8**	×3.9	29.3	**5.3**	×5.6
	Medium	*≈9× compression ($P_{\text{sum},4}$)*			*≈16× ($P_{\text{sum},4}$)*			*≈6× ($P_{\text{sum},5}$)*		
Score		55.6	**63.1**	+7.4	60.8	**75.1**	+14.3	61.8	**68.2**	+6.4
Tokens		658	**599**	−9.0%	353	**338**	−4.3%	996	**900**	−9.6%
$\mathcal{S}$ GPU (ms)		18,896	**2,363**	×8.0	10,459	**2,265**	×4.6	59,281	**3,867**	×15.3
$\mathcal{R}$ TTFT (ms)		34.5	**6.3**	×5.4	50.1	**11.8**	×4.2	35.5	**5.4**	×6.6

Table 16: Experiment 2, SQuAD. T2T: sender summarizes to receiver. SwaT+AM: the sender's cache rendered and compressed with Attention Matching. Budgets are determined by the sender agent in T2T given one of five summary prompts and sorted from lowest to highest per pair. Change: score difference in points, relative token difference, and speedup.

		Qwen3-4B → Llama3-8B			Llama3-8B → Qwen3-14B			Gemma3-12B → Qwen3-4B		
Metric	Budget	T2T	SwaT	Change	T2T	SwaT	Change	T2T	SwaT	Change
	Lowest	*≈3× compression ($P_{\text{sum},1}$)*			*≈4× ($P_{\text{sum},1}$)*			*≈4× ($P_{\text{sum},1}$)*		
Score		51.6	**88.7**	+37.1	48.4	**87.5**	+39.1	35.8	**82.4**	+46.6
Tokens		64	**60**	−5.5%	60	**47**	−22.9%	50	**47**	−7.2%
$\mathcal{S}$ GPU (ms)		1,852	**1,068**	×1.7	1,272	**935**	×1.4	2,923	**1,220**	×2.4
$\mathcal{R}$ TTFT (ms)		6.2	**3.1**	×2.0	11.5	**5.9**	×1.9	4.0	**2.5**	×1.6
	Lower	*≈3× compression ($P_{\text{sum},2}$)*			*≈3× ($P_{\text{sum},2}$)*			*≈3× ($P_{\text{sum},2}$)*		
Score		60.0	**88.7**	+28.7	57.2	**89.4**	+32.2	40.8	**84.7**	+43.9
Tokens		85	**60**	−29.2%	83	**62**	−25.2%	67	**62**	−7.5%
$\mathcal{S}$ GPU (ms)		2,438	**1,068**	×2.3	1,756	**970**	×1.8	3,956	**1,103**	×3.6
$\mathcal{R}$ TTFT (ms)		7.2	**3.1**	×2.3	13.6	**5.9**	×2.3	4.7	**2.6**	×1.8
	Low	*≈2× compression ($P_{\text{sum},3}$)*			*≈2× ($P_{\text{sum},3}$)*			*≈2× ($P_{\text{sum},3}$)*		
Score		67.5	**88.8**	+21.3	61.3	**89.8**	+28.5	47.0	**87.0**	+40.0
Tokens		104	**90**	−13.3%	105	**93**	−11.3%	98	**93**	−5.1%
$\mathcal{S}$ GPU (ms)		2,994	**1,202**	×2.5	2,264	**1,184**	×1.9	5,679	**1,420**	×4.0
$\mathcal{R}$ TTFT (ms)		7.9	**3.2**	×2.5	15.4	**5.9**	×2.6	5.5	**2.6**	×2.1
	Medium-low	*≈2× compression ($P_{\text{sum},5}$)*			*≈2× ($P_{\text{sum},4}$)*			*≈2× ($P_{\text{sum},4}$)*		
Score		74.3	**88.8**	+14.5	72.4	**89.8**	+17.4	59.4	**87.0**	+27.6
Tokens		169	**90**	−46.4%	171	**93**	−45.7%	178	**93**	−47.6%
$\mathcal{S}$ GPU (ms)		4,782	**1,202**	×4.0	3,632	**1,184**	×3.1	9,586	**1,420**	×6.8
$\mathcal{R}$ TTFT (ms)		10.9	**3.2**	×3.4	21.3	**5.9**	×3.6	8.0	**2.6**	×3.1
	Medium	*≈2× compression ($P_{\text{sum},4}$)*			*≈2× ($P_{\text{sum},5}$)*			*≈2× ($P_{\text{sum},5}$)*		
Score		75.7	**88.8**	+13.1	65.8	**89.8**	+24.0	38.6	**87.0**	+48.4
Tokens		172	**90**	−47.4%	177	**93**	−47.4%	179	**93**	−48.1%
$\mathcal{S}$ GPU (ms)		4,829	**1,202**	×4.0	3,740	**1,184**	×3.2	9,637	**1,420**	×6.8
$\mathcal{R}$ TTFT (ms)		11.1	**3.2**	×3.5	21.5	**5.9**	×3.6	8.0	**2.6**	×3.1

Table 17: Geometric mean performance of SwaT over three sender–receiver pairs, seven datasets and five budgets. T2T and SwaT columns are geometric means over all datasets and budgets; Change is their ratio. Score Δ is the arithmetic mean score difference in points.

	Qwen3-4B → Llama3-8B			Llama3-8B → Qwen3-14B			Gemma3-12B → Qwen3-4B			All pairs		
Metric	T2T	SwaT	Change	T2T	SwaT	Change	T2T	SwaT	Change	T2T	SwaT	Change
Score	38.8	**54.8**	+41.2%	39.2	**59.1**	+50.6%	34.1	**52.8**	+54.9%	37.3	**55.5**	+48.8%
Score Δ (pts)	42.9	**57.8**	+14.9	42.9	**61.9**	+19.0	38.0	**56.2**	+18.2	41.3	**58.6**	+17.4
Tokens	293	**254**	−13.1%	269	**243**	−9.7%	309	**269**	−13.1%	290	**255**	−12.0%
$\mathcal{S}$ GPU (ms)	8,752	**2,150**	×4.1	6,198	**2,360**	×2.6	19,761	**2,888**	×6.8	10,234	**2,447**	×4.2
$\mathcal{R}$ TTFT (ms)	18.0	**4.0**	×4.5	31.8	**7.5**	×4.3	14.1	**3.4**	×4.2	20.1	**4.7**	×4.3

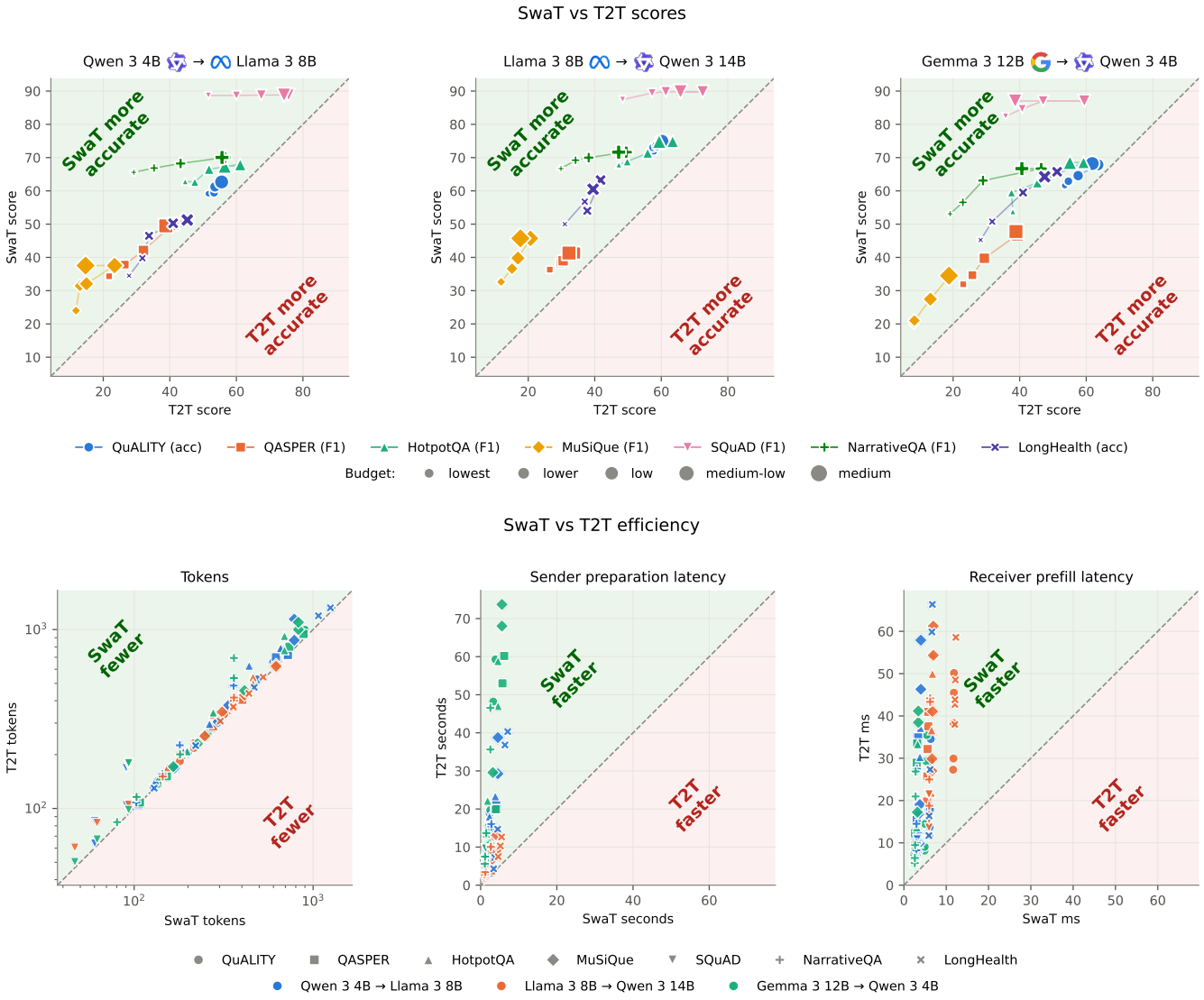


Figure 4: Scatter plot for all individual cells in tables 10 to 16. We note that points in the tokens plot are close to the diagonal by construction as the two methods were picked to have close to matching budgets.

Table 18: Ablations of our SwaT architecture. Each method is compared against the average of two independently trained SwaT models. Differences are in percentage points on the mean benchmark score (accuracy or F1). Metrics calculated over 13 benchmarks with the pair Qwen3-4B → Llama-3-8B.

Ablation	Δ SwaT
SwaT	0
SwaT (w/ c2c [8] token alignment)	−3.20
Headwise MLP	−14.71
Phase 1 only	−36.97
Phase 2 only	−36.96
Dense loss only	−55.11

E Ablation training

In table 18 we provide ablations of our training procedure described in section 3.

F Beyond the Receiver's Context Window

Another zero-shot capability is the ability to provide the receiver with the ability to answer questions about a context beyond its context length. We are able to do this by taking a sender with support for a long context length (such as Qwen 3 which supports a 32k context length) and pairing it with a receiver that only supports an 8k context length. Then, we utilize the sender to generate the KV-cache of a long document, then we chunk the KV-cache into 8k chunks, transfer and compact each one using SwAT+AM, then combine the compacted chunks into a single chunk that the receiver can use. We note that while we do chunk the KV-cache, this is only done after the sender has prefilled the entire document. Thus, each chunk contains cross-dependencies. This is unlike similar capabilities provided by AM which utilize the sender's context to compress each chunk which does not capture cross–chunk dependencies. We show the results for this on a version of HotpotQA where each context was inflated (using distractors from other samples) into 28k tokens thus far exceeding the receiver's 8k limit. We report the results in table 19.

Table 19: Interleaved HotpotQA (500 questions; EM/F1 in %). Sequence lengths is number of text tokens or KV-cache tokens.

Method	Seq. len	EM	F1
Baselines			
Qwen 3 4B	28K	20.4	36.75
Llama 3 8B truncate	8K	5.4	13.32
Llama 3 8B AM_{20}	5.6K	21.2	34.95
SwAT			
Qwen $\rightarrow^{AM_{20}}$ Llama	5.6K	**24.8**	**41.87**

G Additional Vision Transfer

We add an additional ablation to the zero-shot vision transfer in table 20 where we train a SwAT to transfer from a Gemma 3 4B to itself; this measures the effect of the training data on the results. We do see that this ablation does improve the scores but not to the same degree as the Gemma to Granite pair.

Additionally, we show that zero-shot visual transfer is far from perfect. By testing two datasets in table 21 that require heavy visual processing, we show that the sender VLM model alone achieves a much higher accuracy than both T2T and SwAT. This shows that the zero-shot transfer is not transferring full vision capabilities especially on datasets that require such heavy processing.

Table 20: Similar to table 4. Test accuracy (%) for multiple pairs including a control where sender and receiver are the same VLM (Gemma 3 4B → Gemma 3 4B pair).

Method	VQAv2	TQA	OKVQA	GQA	VW	Avg.
Gemma 3 4B → Gemma 3 4B						
Sender	45.3*	53.4*	22.0	31.4*	**19.0**	34.2±.4*
SwAT	**53.8**	**56.6**	**22.4**	**35.3**	**19.0**	**37.4±.4**
Gemma 3 4B → Granite 3.3 8B						
Sender	45.3*	53.4*	22.0*	31.4*	**19.0**	34.2±.4*
T2T	33.8*	32.0*	15.7*	16.3*	7.7*	21.1±.4*
SwAT	**59.5**	**61.8**	**28.7**	**39.5**	18.3	**41.6±.4**
Gemma 3 4B → Llama 3 8B						
Sender	45.3*	53.4*	22.0*	31.4*	**19.0**	34.2±.4*
T2T	16.4*	12.3*	18.3*	10.9*	8.4*	13.3±.3*
SwAT	**57.5**	**56.4**	**29.7**	**35.5**	18.7	**39.6±.4**
Gemma 3 12B → Qwen 3 4B						
Sender	44.5*	22.4*	**24.4**	28.3*	**20.0**	27.9±.4*
T2T	47.0*	37.6*	19.1*	29.7*	12.2*	29.1±.4*
SwAT	**51.7**	**49.4**	21.8	**33.0**	15.1	**34.2±.4**

Table 21: Similar to table 4. Test accuracy (%) for multiple pairs on DocVQA and InfoVQA, which depend on dense text and fine detail in the image.

Method	DocVQA	InfoVQA	Avg.
Gemma 3 4B → Gemma 3 4B			
Sender	**73.3**	**41.6**	**57.4±.8**
SwAT	40.9	26.8	33.8±.8
Gemma 3 4B → Granite 3.3 8B			
Sender	**73.3**	**41.6**	**57.4±.8**
T2T	39.3*	25.5*	32.4±.8*
SwAT	43.4	31.5	37.5±.8
Gemma 3 4B → Llama 3 8B			
Sender	**73.3**	**41.6**	**57.4±.8**
T2T	38.4*	26.6*	32.5±.8*
SwAT	43.3	28.6	36.0±.8
Gemma 3 12B → Qwen 3 4B			
Sender	**78.1**	**46.3**	**62.2±.8**
T2T	48.1	29.3	38.7±.8
SwAT	35.1	23.0	29.0±.7