MMA: EFFICIENT LATENT COMMUNICATION FOR PARALLEL LARGE LANGUAGE MODEL AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large Language Models (LLMs) process context through a sequential token interface. However, current multi-agent systems naturally involve parallel branches with disjoint evidence or private contexts. Existing agent communication methods through text or sequence-valued latent space incur decoding re-prefill overhead and introduce communication costs that grow with context length. In this paper, we propose *Multi-Modal Attention* (MMA), a latent communication framework for frozen parallel LLMs. Each branch retains its context and Key Value (KV) cache locally, exchanging only current-token hidden states at sparse synchronization layers. Furthermore, a latent-fusion aggregator produces a shared next-token distribution for all branches. Because MMA communicates current-token states rather than sequence-length-dependent memories, its payload is independent of input length, and the same interface naturally extends to multiple branches, heterogeneous model widths, and different input modalities. Evaluated on information retrieval benchmarks spanning different context lengths, MMA matches or outperforms other communication baselines while remaining efficient with significantly less communication payload. Its performance advantages are most pronounced in long-context, multi-branch retrieval. These results support MMA as a scalable communication interface for parallel multi-agent inference.

1 INTRODUCTION

Modern multi-agent systems (MASs) are distributed computational structures: multiple Large Language Model (LLM) agents may operate concurrently, interact with different resources, and contribute to a common objective (Li et al., 2023; Hong et al., 2024; Wu et al., 2024). Their workflows therefore often contain parallel branches. Across branches, agents may assume complementary roles, retrieve from different sources, or operate on local and potentially private information (Qian et al., 2024; Wang et al., 2025; Liu et al., 2026). Each branch can process its evidence independently, but producing the final answer often requires integrating information across branches. Serializing these contexts into one ordered prompt collapses this parallel structure, weakens context isolation, and places the entire long-context processing burden on one model (Liu et al., 2024a). This creates a fundamental mismatch between the sequential, concatenation-based interface of LLMs and the inherently parallel, multi-branch organization of MASs.

Existing inter-LLM communication methods attempt to tackle this mismatch by preserving parallel local processing and combining branch information when coordination is required. Text-mediated communication is model-agnostic and interpretable, but requires intermediate decoding and re-prefill and may compress branch evidence into lossy messages (Li et al., 2023; Qian et al., 2024; Wu et al., 2024; Wang et al., 2025; Zhuge et al., 2024). KV-cache communication retains richer token-level representations, but its sequence-valued payload grows with context length and branch count (Fu et al., 2026; Shi et al., 2026; Zou et al., 2026; Liu et al., 2026; 2024b). Compact activation communication avoids both full decoding and full-cache transfer, yet existing designs primarily support one-way transfer rather than coordination throughout joint decoding (Ramesh & Li, 2025). Moreover, these interfaces have been studied primarily with homogeneous, text-only branches, leaving communication across heterogeneous input modalities (e.g., text and image) underexplored.

To address the above research gap, we propose the *Multi-Modal Attention* (MMA), a latent communication framework for parallel frozen LLMs. Each branch processes its own context under a

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.

shared query while retaining its KV cache locally. During autoregressive decoding, a directionally gated cross-branch attention mixer synchronizes current-token hidden states at a sparse set of transformer layers. A latent-fusion aggregator then combines the final branch states into a shared next-token distribution. The selected token is returned to every branch for the next decoding step. MMA therefore integrates distributed evidence throughout generation without serializing intermediate responses. Since MMA exchanges only current-token states, its communication payload is independent of input context length. As additional branches are introduced, even with heterogeneous input modalities, they can participate in the same shared decoding process through latent communication, without generating separate responses that require subsequent synthesis.

We evaluate MMA on long-context retrieval and multimodal reasoning benchmarks. MMA matches or outperforms other communication baselines while maintaining low latency and compact communication payload. Our contributions are threefold. **First,** we introduce MMA for joint inference over distributed contexts. Multiple LLM branches exchange current-token hidden states and jointly determine each next token, producing one shared answer rather than separate responses that require subsequent synthesis. **Second,** we make this collaboration communication-efficient. Sparse synchronization exchanges only compact current-token states while keeping contexts and KV caches local, making the per-step payload independent of input length. **Third,** we establish MMA's generality across branch configurations. The proposed MMA mixer naturally extends the settings to multiple branches, heterogeneous model widths, and different input modalities within the same framework. Beyond these contributions, a preliminary privacy study in Appendix suggests that MMA may have better privacy protection, since its narrow latent channel could limit the exposure of branch-local information unrelated to the shared task.

2 Related Work

Text-mediated multi-agent collaboration. Natural language is the prevailing interface for coordinating LLM agents. AutoGen organizes agents through conversational message, GPTSwarm represents agent workflows as optimizable computation graphs, and Mixture-of-Agents conditions each layer of agents on the textual outputs of the preceding layer (Wu et al., 2024; Zhuge et al., 2024; Wang et al., 2025). While this interface is interpretable and model-agnostic, each communication edge requires a sender to decode a message and a receiver to process it again. Such costs grow as more branches contribute messages and generally require a final synthesis agent call to unify the answers. Our proposed MMA instead coordinates branches within one shared decoding trajectory, without generating intermediate text.

Sequence-valued latent communication. Several methods bypass textual messages by communicating token-indexed internal states. C2C learns to project and fuse a sender's KV cache into a receiver, KVComm transmits KV states from selected layers, and LatentMAS exposes latent reasoning states and their associated cache to a receiving model (Fu et al., 2026; Shi et al., 2026; Zou et al., 2026). Parallel Synthesis maps independently produced worker caches into a downstream synthesizer and adapts that synthesizer to the resulting non-sequential cache interface (Liu et al., 2026). These approaches retain richer representations than decoded summaries, but their communication remains sequence-valued. The payload grows with the transmitted cache length and the number of branches. In contrast, our MMA keeps token histories local and exchanges only one current-token state per branch at sparse layers, making its payload independent of the input context length.

Compact activation communication. Activation Communication (AC) communicates a sender's last-token activation through a directional intervention in the receiver (Ramesh & Li, 2025). ThoughtComm similarly uses a compact carrier: it compresses agents' post-generation hidden states into sparse latent thoughts and injects selected thoughts as learned prefixes in a subsequent collaboration round (Zheng et al., 2025). Both avoid transmitting full KV caches, but their communication remains intervention- or round-based: AC performs a one-way activation graft, while ThoughtComm communicates only after agents have generated their current responses. Our MMA instead synchronizes all branches recurrently within a shared autoregressive trajectory, using gated residual attention without replacing their native states.

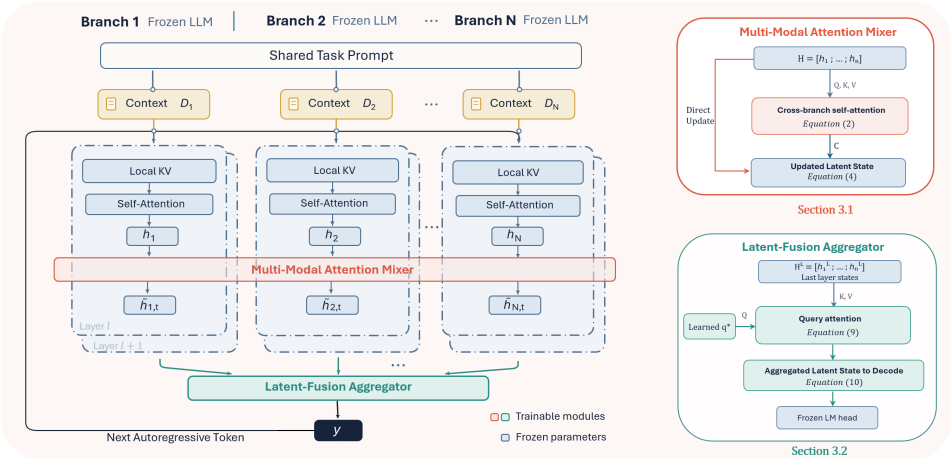


Figure 1: Architecture of the proposed MMA framework. Left: parallel frozen LLM branches with disjoint contexts communicate current token embedding at selected layers and output shared next token. Right: the cross-branch MMA mixer and latent-fusion aggregator. Blue denotes frozen components, while red and green denote trainable modules.

Collaborative decoding and model composition. Collaborative decoding and agent communication are closely related, both coordinating independent information toward a shared answer. FiD independently encodes passages and combines their sequence representations through decoder cross-attention (Izacard & Grave, 2021). PCW processes context windows separately and allows a single decoder to attend across their combined KV caches (Ratner et al., 2023). Both of them share the same underlying principle of the sequence-valued communication methods discussed above. MMA can likewise be viewed as collaborative decoding, but coordinates model branches through compact current-token exchange. CALM likewise composes frozen models through learned cross-attention to inject an augmenting model's sequence representations into an anchor model (Bansal et al., 2024). But MMA instead enables bidirectional exchange among branches with separate contexts, restricting communication to current-token states while keeping sequence memories local

3 Methodology

Figure 1 illustrates the overall MMA framework. Given a shared task prompt, N frozen LLM branches process disjoint local contexts $\{D_i\}_{i=1}^{N}$ in parallel while retaining their context-dependent KV caches locally. At each selected synchronization layer, the MMA mixer collects only the branches' current-token states, applies cross-branch self-attention, and returns a gated residual update to each branch. After the final layer, a latent-fusion aggregator uses a learned query to attend over the final branch states and combines the resulting summary with a designated anchor state. The corresponding frozen LM head produces a shared next-token distribution, and the selected token is returned to every branch to continue the same autoregressive trajectory.

3.1 Multi-Modal Attention Mixer

For clarity, we first consider branches with a common hidden dimension d. At decoding step t, let $\mathbf{h}_{i,t}^{\ell} \in \mathbb{R}^d$ denote the current-token state of branch i at synchronization layer ℓ. MMA collects the states from all N branches:

$$\mathbf{H}_t^{\ell} = \left[\mathbf{h}_{1,t}^{\ell}; \ldots; \mathbf{h}_{N,t}^{\ell}\right] \in \mathbb{R}^{N \times d}. \tag{1}$$

The branch axis is treated as a short sequence, and multi-head self-attention is applied across branches rather than across token positions:

$$\mathbf{C}_t^{\ell} = \mathrm{MHA}_{\ell}\left(\mathbf{H}_t^{\ell}, \mathbf{H}_t^{\ell}, \mathbf{H}_t^{\ell}\right), \tag{2}$$

where $\mathbf{c}_{i,t}^{\ell}$ denotes the cross-branch attention output for branch i. This path allows each branch to retrieve task-relevant information from the current states of the complete branch set without accessing their context tokens or KV caches.

In parallel, MMA provides an explicit directional communication path. For each ordered sender–receiver pair, a learned gate $g^{\ell}_{i\leftarrow j,t} \in [0,1]$ controls the contribution of sender j to receiver i. The direct update is

$$\mathbf{d}^{\ell}_{i,t} = \mathbf{W}^{\ell}_{d}\left(\frac{1}{N-1}\sum_{j\neq i} g^{\ell}_{i\leftarrow j,t}\mathbf{h}^{\ell}_{j,t}\right). \tag{3}$$

Because the gates are directional, $g^{\ell}_{i\leftarrow j,t}$ and $g^{\ell}_{j\leftarrow i,t}$ are learned independently. This enables MMA to represent asymmetric information flow even when all branches use the same backbone.

The attention and direct paths are combined through a residual update:

$$\widetilde{\mathbf{h}}^{\ell}_{i,t} = \mathbf{h}^{\ell}_{i,t} + g^{\ell}_{i,t}\mathbf{c}^{\ell}_{i,t} + \mathbf{d}^{\ell}_{i,t}, \qquad g^{\ell}_{i,t} \in [0,1], \tag{4}$$

where $g^{\ell}_{i,t}$ is a learned factor controlling the overall attention update received by branch i. The attention path performs content-dependent aggregation over the branch set, while the direct path preserves explicit sender-to-receiver communication. Their residual form retains each branch's native representation and introduces only a learned communication delta.

The mixer is applied only at a sparse set of synchronization layers. We select the synchronization schedule using the ablation presented in Appendix A.6.

Heterogeneous hidden dimensions. When branch i has native hidden dimension d_i, its current-token state is first mapped into a common d_c-dimensional communication space:

$$\mathbf{z}^{\ell}_{i,t} = \mathbf{P}^{\ell}_{i}\mathbf{h}^{\ell}_{i,t}, \qquad \mathbf{z}^{\ell}_{i,t} \in \mathbb{R}^{d_c}. \tag{5}$$

The states $\mathbf{Z}^{\ell}_{t} = [\mathbf{z}^{\ell}_{1,t}; \ldots; \mathbf{z}^{\ell}_{N,t}]$ are processed by the same mixer described above. To avoid replacing a native state with a potentially lossy reconstruction, MMA projects only the resulting communication residual back to each branch:

$$\Delta\mathbf{z}^{\ell}_{i,t} = \widetilde{\mathbf{z}}^{\ell}_{i,t} - \mathbf{z}^{\ell}_{i,t}, \tag{6}$$

$$\widetilde{\mathbf{h}}^{\ell}_{i,t} = \mathbf{h}^{\ell}_{i,t} + \mathbf{R}^{\ell}_{i}\Delta\mathbf{z}^{\ell}_{i,t}, \tag{7}$$

where $\mathbf{R}^{\ell}_{i}$ maps the shared-space residual back to the native dimension d_i.

3.2 Latent-Fusion Aggregator

The latent-fusion aggregator follows the same branch-attention principle as the MMA mixer, but replaces the branch-conditioned queries with a single learned query $\mathbf{q}^{*}$. Let

$$\mathbf{H}^{L}_{t} = \left[\mathbf{h}^{L}_{1,t}; \ldots; \mathbf{h}^{L}_{N,t}\right] \in \mathbb{R}^{N\times d} \tag{8}$$

denote the final current-token states of the N branches. The learned query attends over these states to obtain a global answer representation:

$$\mathbf{z}_{t} = \mathrm{MHA}_{\mathrm{fuse}}\left(\mathbf{q}^{*}, \mathbf{H}^{L}_{t}, \mathbf{H}^{L}_{t}\right). \tag{9}$$

Whereas the MMA mixer returns one updated state for every branch, the latent-fusion aggregator produces a single state for next-token prediction.

We designate one branch a as the decoding branch and fuse the aggregated representation into its native final state through a gated residual:

$$\mathbf{f}_{t} = \mathbf{h}^{L}_{a,t} + \mathbf{g}^{f}_{t} \odot \mathbf{W}_{\triangle}\mathbf{z}_{t}, \tag{10}$$

where $\mathbf{g}^{f}_{t}$ is a learned fusion gate. The frozen LM head of the decoding branch then produces the shared next-token distribution:

$$p(y_t \mid y_{<t}, D_{1:N}, q) = \mathrm{softmax}\left(\mathrm{LMHead}_{a}(\mathbf{f}_{t})\right). \tag{11}$$

Only this decoding branch exposes an output distribution; logits from the remaining branches are not combined or released. The selected token is nevertheless returned internally to every branch so that all branches continue along the same autoregressive trajectory.

Separating communication from decoding allows the output-producing branch to be selected according to the deployment scenario, such as assigning answer generation to a trusted participant while the other branches contribute only latent information. For homogeneous branches with compatible hidden and vocabulary spaces, changing the designated decoding branch of a trained checkpoint of MMA has negligible effect on performance, as shown in Appendix A.7.

4 EXPERIMENTS

We evaluate MMA as a mechanism for parallel communication among LLM agents. In Section 4.2, our primary setting considers two homogeneous branches that share the same frozen LLM and question, but receive non-overlapping subsets of the context. We then investigate the generality of MMA in three extensions: First, we investigate how performance changes as the number of branches increases in Section 4.3. Then, Section 4.4 investigates cooperation between heterogeneous settings, including model implementation, context loading, and input modality. Finally, we simulate long context communication with distractors and its influence in communication payload in Section 4.5.

4.1 SETUP AND EVALUATION PROTOCOL

MMA uses the sparse-late-4 directional mixer: one eight-head self-attention layer communicates across branches at four late backbone layers. A one-layer, four-head next token aggregator combines each branch's hidden states for autoregressive answer generation. All backbone parameters remain frozen; only the mixer and aggregator are optimized.

Benchmarks. We evaluate MMA on three multi-document QA benchmarks spanning long, medium, and short context regimes. *Loong* evaluates long-context reasoning where multiple relevant documents jointly determine a structured answer (Wang et al., 2024), with an average context length of 23.4K tokens. *MultiHop-RAG* requires reasoning across retrieved news articles to produce a single short answer (Tang & Yang, 2024), with an average context length of 5.58K tokens. *HotpotQA* represents the short-context setting, in which answers are inferred from two supporting passages (Yang et al., 2018), averaging 241 tokens. *MultiModalQA* tests cross-modal reasoning over image evidence paired with text or table evidence (Talmor et al., 2021).

We evaluate on 80 *Loong* examples, each containing 3–5 QA pairs (326 QA units in total), and 300 examples each from *MultiHop-RAG*, *HotpotQA* and *MultiModalQA*. For *Loong*, we report both example-level perfect accuracy, which requires all QA pairs to be correct, and QA-unit F1.

Models and baselines. In this experiment, we use the most popular open-source Qwen3 models (Yang et al., 2025). We compare MMA with the following recent representative baselines:

- **Concatenation.** A single model receives the union of the evidence from all branches. This serves as an reference assuming all evidence can be collected and assembled into one prompt. In practical multi-agent systems, such centralization may be infeasible due to privacy restrictions or limited knowledge of and access to other branches' contexts.
- **Text MAS.** Two workers independently process disjoint evidence subsets and communicate their findings through text to a synthesis agent, which produces the final answer without directly accessing the original evidence (Zhuge et al., 2024; Zou et al., 2026).
- **LatentMAS.** The sender and receiver process disjoint evidence subsets. The sender communicates its latent reasoning states and KV cache to the receiver, which generates the final answer (Zou et al., 2026).
- **KVComm.** The two models receive disjoint evidence subsets and communicate by sharing KV states from a Gaussian-selected 30% of the model layers (Shi et al., 2026).
- **Activation Communication (AC).** The sender and receiver independently process disjoint evidence subsets. At an intermediate layer, the sender's final prompt-token activation replaces the receiver's corresponding activation for communication. (Ramesh & Li, 2025).
- **Cache-to-Cache (C2C).** The sender and receiver process disjoint evidence subsets, and a learned projector maps the sender's KV cache into the receiver's representation space for fusion (Fu et al., 2026).
- **Parallel-Synthesis.** Two workers process disjoint evidence subsets in parallel and communicate the KV states of their generated messages to a LoRA-fintuned synthesis model. A learned cache mapper calibrates and combines the branch caches (Liu et al., 2026).

All methods use the same questions, evidence partitions, and decoding protocol. Experiments are conducted on a single NVIDIA RTX 5090, with latency measured end-to-end, including communication and answer generation. Implementation details can be found at Appendix A.1.

Table 1: Two-branch evaluation on *Loong*, *MultiHop-RAG*, and *HotpotQA*. *Loong* reports manually audited perfect accuracy and QA-pair unit F1; *MultiHop-RAG* and *HotpotQA* report strict exact-match accuracy. Performance values are percentages; Lat. latency is seconds per example. Bold performance values indicate the strongest two-evidence-branch method.

Method	Loong			MultiHop-RAG		HotpotQA	
	Perfect ↑	F1 ↑	Lat. ↓	Strict EM ↑	Lat. ↓	Strict EM ↑	Lat. ↓
Qwen3-1.7B							
Reference baselines							
Concatenation	11.25	34.52	4.34	57.67	0.21	46.67	0.14
MMA (empty peer)‡	12.75	29.88	2.75	68.67	0.17	42.33	0.20
Two evidence-bearing streams							
Text MAS	10.00	30.25	10.82	57.67	3.68	34.67	2.78
LatentMAS	6.25	19.54	2.61	44.67	0.80	48.00	0.30
KVComm	8.75	17.75	3.06	58.33	0.83	32.33	0.17
C2C	7.50	18.27	4.73	70.67	0.59	40.67	0.14
AC	13.75	29.76	1.47	50.00	0.59	27.33	0.16
Parallel Synthesis†	6.25	23.71	9.17	78.67	2.99	54.00	0.70
MMA	**15.00**	**38.05**	5.47	**80.33**	0.22	**56.67**	0.19
Qwen3-4B							
Reference baselines							
Concatenation	21.25	57.24	5.45	76.00	0.46	55.00	0.14
MMA (empty peer)‡	15.00	37.06	3.28	73.67	0.30	46.00	0.31
Two evidence-bearing streams							
Text MAS	11.25	35.99	8.31	67.00	4.44	46.67	3.32
LatentMAS	7.50	19.21	8.89	62.33	1.18	56.33	0.37
KVComm	8.75	25.84	5.67	74.00	0.63	39.00	0.26
C2C	10.00	20.21	9.04	77.00	1.27	48.67	0.22
AC	12.50	28.34	2.58	64.00	0.41	28.00	0.28
Parallel Synthesis†	15.00	38.67	10.23	**82.33**	4.88	**60.67**	1.35
MMA	**26.25**	**56.31**	8.92	82.00	0.43	58.67	0.24

†Parallel Synthesis trains synthesizer LoRA adapters; other methods freeze their backbones. Results without LoRA appear in Appendix A.2. ‡Same as MMA but with one no-context branch during training and evaluation; the decoder branch retains its original evidence bucket untouched, and communication remains active.

4.2 Main Experiments

Table 1 considers the situation where two homogeneous branches share the same frozen LLM and question, but receive non-overlapping subsets of the context. Among all benchmarks, each subset context document is assigned in its entirety to exactly one branch: documents are neither split nor duplicated. The union of the branch inputs is the evidence supplied to the concatenation baseline.

MMA achieves the best primary accuracy in most experimental settings. Its advantage is most significant on *Loong* with average context length of 23.4K tokens. We hypothesize that MMA's direct hidden-state interaction utilizes frozen LLMs more naturally than methods that serialize communication into text or exchange compressed intermediate representations. On the shorter *MultiHop-RAG* and *HotpotQA* benchmarks, MMA achieves accuracy comparable to Parallel Synthesis while running 3.7–13.6× faster. Both methods substantially outperform the remaining baselines. In overall, MMA is particularly beneficial when the context is long or backbone capacity is limited.

To assess the contribution of peer evidence beyond task-specific adaptation, we retrain MMA (empty peer) with the same architecture and trainable parameter count but remove all documents from the sender branch. Decoding branch retains its original evidence bucket, the mixer and aggregator also remain active. The substantial performance drops on *Loong* and *HotpotQA* support the importance of communicating complementary evidence rather than attributing MMA's gains solely to task-specific adaptation. The smaller gap on *MultiHop-RAG* comes from some questions being answerable with

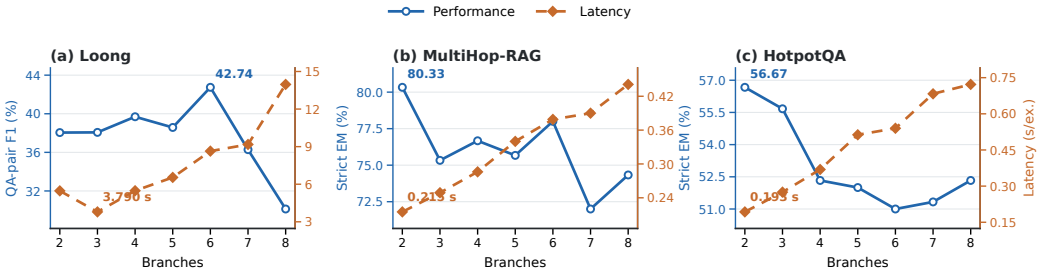


Figure 2: Effects of the number of branches on accuracy and latency (in Qwen3 1.7B MMA). (a) *Loong* performance peaks at six branches before excessive context fragmentation dominates. (b) *MultiHop-RAG* achieves its strongest accuracy–latency operating point with two branches. (c) On *HotpotQA*, additional branches increase communication cost without constribution to performance.

a single branch's evidence, allowing supervised adaptation to remain effective. To further clarify the importance of communication, Appendix A.8 removes one branch's context from the trained *MultiHop-RAG* MMA without further fine-tuning. The resulting substantial performance drop supports MMA's improvement is not from task-specific adaptation alone.

4.3 Extension to More Branches

A key advantage of MMA is its natural scalability to multi-branch settings whereas other communication methods become difficult to coordinate as the number of branches grows. In this section, we vary the number of MMA branches on Qwen3 1.7B from 2 to 8 and examine its effect on performance and latency. On average, *Loong*, *MultiHop-RAG*, and *HotpotQA* contain 7.91, 2.24, and 2.00 documents per example. Since each document is assigned in full to exactly one branch without duplication, with the increase of MMA branch, there will be an increase in number of branches receiving no document. Those empty-document branch are randomly assigned to remove bias.

Figure 2 reveals that the optimal number of branches depends on the structure and length of the input evidence. On *Loong* with average of 7.91 documents per example, increasing the branch count initially distributes the long context into shorter and more manageable streams. Performance peaks at six branches with 42.74 F1. However, after six branches, further context fragmentation increases difficulty of aggregating many latent streams and outweighs the benefit of shorter per-branch contexts: F1 falls to 36.28 with seven branches and 30.13 with eight.

The shorter-context benchmarks exhibit a different pattern. Every *HotpotQA* example contains exactly two supporting documents, so increasing the branch count beyond two does not reduce the evidence burden of the non-empty branches. Instead, it introduces additional evidence-free streams and greater aggregation cost. Consequently, EM generally decreases from 56.67% with two branches to approximately 51–52% with four or more branches. *MultiHop-RAG*, with 2.24 documents per example, shows a similar but less monotonic pattern. These results show that adding branches is most beneficial when there is sufficient long-context evidence. Once the branch count substantially exceeds the available evidence, evidence fragmentation and aggregation overhead dominate. Detailed experiment result in table can be found at Appendix A.3.

4.4 Generalization to Heterogeneous Settings

Heterogeneous branch model. Heterogeneous model implementation on agent communication always introduce challange in model layer/dimension mismatch. In this section, we distribute the whole evidence context passage into one 1.7B model and 4B model and let the decoding branch be the larger 4B model for all methods. In table 2, MMA achieves the best performance among all applicable methods across the three benchmarks. Compared with the symmetric 1.7B–1.7B setting, the heterogeneous 1.7B–4B configuration decreases strict EM from 80.33 to 75.00 on *MultiHop-RAG* and from 56.67 to 52.00 on *HotpotQA*. These relatively easier tasks offer limited opportunity to benefit from the larger model while the model mismatch introduces an additional representation-alignment burden. In contrast, Loong QA-unit F1 increases from 38.05 to 43.77, suggesting that

Table 2: Heterogeneous-model evaluation with Qwen3-1.7B and Qwen3-4B branch implementations. *Loong* reports perfect accuracy and QA-unit F1; *MultiHop-RAG* and *HotpotQA* report strict exact-match accuracy. Performance values are percentages, and latency is measured in seconds/example.

Method	Loong			MultiHop-RAG		HotpotQA	
	Perfect ↑	F1 ↑	Lat. ↓	Strict EM ↑	Lat. ↓	Strict EM ↑	Lat. ↓
Parallel Synthesis				*Not applicable*			
KVComm							
LatentMAS	1.25	12.59	4.43	52.00	0.83	35.33	0.46
C2C	5.00	20.54	17.22	73.67	3.33	48.67	0.25
AC	1.25	19.04	3.56	15.67	0.66	10.67	0.35
MMA	**12.50**	**43.77**	4.50	**75.00**	**0.29**	**52.00**	0.27

Table 3: Heterogeneous-context evluation with all evidence is assigned in the sender branch and only question in decoder branch. Experiment is conducted on 300 *MultiHop-RAG* examples, without further fine-tuning. Values are percentages.

Metric	LatentMAS	KVComm	AC	C2C	MMA
Strict EM ↑	58.33	47.67	35.00	49.00	**65.33**
Canonical Acc. ↑	59.33	48.67	35.33	51.00	**66.33**

MMA can retain the larger model's advantage on challenging long-context tasks even under an heterogeneous configuration.

Heterogeneous context loading. To test extreme-case robustness when the decoding branch has no local evidence, we assign all supporting documents to the communication sender and give the decoder only the question. We evaluate the existing methods on the same *MultiHop-RAG* examples without further fine-tuning, only the evidence assignment changes. Text MAS and Parallel Synthesis are excluded because their separate synthesizers do not provide an analogous decoding branch. Table 3 shows that MMA retains the highest accuracy when the decoder must rely on information from the sender. Its strict EM falls from 80.33% with distributed evidence to 65.33%, indicating that this one-sided assignment remains challenging. C2C falls more sharply, from 70.67% to 49.00%, suggesting a practical limitation of its projected KV-cache transfer when all task evidence resides outside the decoding branch.

Heterogeneous modality. Because MMA communicates through current-token hidden states rather than modality-specific input representations, its communication interface naturally extends across modality boundaries. We evaluate this property on *MultiModalQA* using a Qwen3-VL-2B branch for image evidence and a Qwen3-1.7B branch for the corresponding text or table evidence. Both branches receive the shared question. Table 4 shows that MMA achieves the strongest performance among the applicable methods, 7.33 % higher than C2C and 20.33% higher than LatentMAS. *MultiModalQA* may require multiple answer items, Concatenation often returns a broader set of plausible candidates obtaining F1 credit for correct items but failing EM when additional items fall outside the target answer set. Because raw inputs and KV caches remain within their originating branches, this setting also highlights MMA's potential for sharing task-relevant information while keeping modality-specific private data local.

4.5 Context-Scalable Communication with Distractors

Distributed long-context retrieval makes inter-worker communication a first-order cost: transferring long KV caches can dominate context loading and makes worker placement bandwidth-sensitive (Liu et al., 2024b; Zhong et al., 2024). MMA instead exchanges only the current-token state at sparse selected layers. Therefore, the communication payload is independent of input length. We evaluate all methods on the *MultiHop-RAG* set without retraining, adding answer-filtered whole-document

Table 4: Heterogeneous-modality evaluation on the fixed 300-example *MultiModalQA* set. For two-branch methods, Qwen3-VL-2B receives image evidence, while Qwen3-1.7B receives text or table evidence and generates the answer. Concatenation gives all evidence to a single Qwen3-VL-2B model. Values are percentages.

Method	**Overall** ($n = 300$)		**Image + Table** ($n = 216$)		**Image + Text** ($n = 84$)	
	EM ↑	**F1** ↑	**EM** ↑	**F1** ↑	**EM** ↑	**F1** ↑
Parallel Synthesis	*Not applicable*					
KVComm						
Text MAS	15.00	23.17	15.28	22.00	14.29	26.17
LatentMAS	7.33	15.96	7.41	14.43	7.14	19.89
C2C	20.33	26.03	20.83	23.25	19.05	33.20
MMA	**27.67**	32.00	**27.31**	29.82	**28.57**	**37.61**
Concatenation	19.00	**33.28**	20.37	**31.33**	15.48	36.92

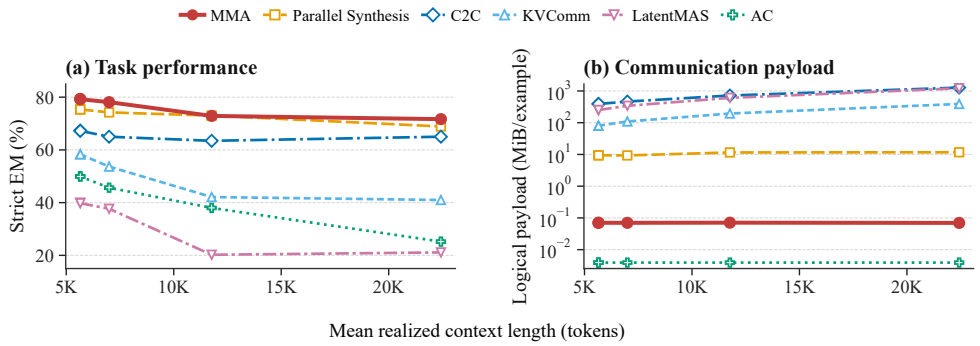


Figure 3: Accuracy and communication payload results when increasing *MultiHop-RAG* context by distractors. MMA preserves task quality while its current-token payload remains nearly independent of input length. Sequence-valued cache transfer grows with the context

distractors to obtain 5.6K–22.4K-token prompts. Every method receives identical evidence under token-balanced assignment. We report strict EM, native logical payload delivered to the receiver.

At 22.4K tokens, MMA achieves the highest strict EM (71.67%) while communicating only 0.070 MiB per example, compared with 11.70 MiB for Parallel Synthesis and 1,278 MiB for C2C. Across 5.6K–22.4K tokens, MMA's payload remains constant (0.99×), while C2C, KVComm, and LatentMAS grow by 3.28–4.87×. Please check Appendix A.9 for detailed result in table.

5 Conclusion and Limitation

In this paper, we introduce MMA, a latent communication framework that enables frozen LLM branches to integrate distributed evidence within a shared decoding process. Sparse cross-branch MMA mixer and latent fusion aggregator provide strong task performance with compact communication while keeping contexts and KV caches local. The framework supports multiple branches, heterogeneous model widths, and different input modalities, making it well suited to collaborative inference over long, distributed contexts. However, MMA's total communication accumulates with output length. Extrapolating from our measured per-token payloads suggests that, for outputs on the order of 10K tokens, repeated synchronization can eliminate its payload advantage over other sequence-valued latent communication. This limitation makes MMA less suitable for long reasoning trajectories generation. Future work could extend MMA to this regime through adaptive communication that synchronizes branches only when needed, rather than at every decoding step.

6 AI USE STATEMENT

In this work, we used generative AI tools for implement methods code of MMA. We have not used generative AI tools for design or provide feedback on research methodology or experiments; and (1) generate synthetic data sets, (2) help develop theoretical models or conceptual frameworks, (3) formulate mathematical claims, (4) provide critical ingredients for proving mathematical claims, (5) assist in the writing of proofs, (6) propose or refine hypotheses, (7) assist with translation, (8) clean and reformat dataset, (9) support qualitative and thematic data analysis and (10) interpret results are not applicable to this work.

Additionally, we used generative AI tools for create or edit some parts of the software code. But the architectural design, system structure, and integration of these components were entirely planned and executed by us. We used AI tools for sourcing/searching for information and related work that we may have missed after our own manual literature search. We also used AI tools to proofread the research paper after we finished writing it, to improve readability and perform grammar checks. We have reviewed all AI-assisted work. The two authors verified and tested LLM-generated code for correctness. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

7 REPRODUCIBILITY STATEMENT

We provide an anonymous repository containing the core MMA implementation, evaluation scripts and configurations, fixed evaluation splits, and the Qwen3-1.7B checkpoints used for the main results: `https://anonymous.4open.science/r/mma-evaluation-artifact-BC09/`. The repository includes instructions for reconstructing and verifying the checkpoint files and reproducing evaluation on *Loong*, *MultiHop-RAG*, and *HotpotQA*. Architectural details are presented in the methodology, while training configurations, baseline implementations, evaluation protocols, and additional ablations are documented in the appendix. The remaining training and experiment-management code will be released upon acceptance.

REFERENCES

Rachit Bansal, Bidisha Samanta, Siddharth Dalmia, Nitish Gupta, Sriram Ganapathy, Abhishek Bapna, Prateek Jain, and Partha Talukdar. LLM augmented LLMs: Expanding capabilities through composition. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=jjA4O1vJRz`.

Tianyu Fu, Zihan Min, Hanling Zhang, Jichao Yan, Guohao Dai, Wanli Ouyang, and Yu Wang. Cache-to-cache: Direct semantic communication between large language models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL `https://openreview.net/forum?id=LeatkxrBCi`.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. MetaGPT: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=VtmBAGCN7o`.

Gautier Izacard and Edouard Grave. Leveraging passage retrieval with generative models for open domain question answering. In Paola Merlo, Jorg Tiedemann, and Reut Tsarfaty (eds.), *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pp. 874–880, Online, April 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.eacl-main.74. URL `https://aclanthology.org/2021.eacl-main.74/`.

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: Communicative agents for "mind" exploration of large language model society. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 51991–52008. Curran Associates, Inc., 2023. doi: 10.52202/

075280-2264. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/a3621ee907def47c1b952ade25c67698-Paper-Conference.pdf.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024a. doi: 10.1162/tacl_a_00638. URL https://aclanthology.org/2024.tacl-1.9/.

Shikun Liu, Mufei Li, Dongqi Fu, Haoyu Wang, Yinglong Xia, Hong Li, Hong Yan, and Pan Li. Towards direct latent-space synthesis for parallel branches in llm-agent workflows, 2026. URL https://arxiv.org/abs/2606.14672.

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*, ACM SIGCOMM '24, pp. 38–56, New York, NY, USA, 2024b. Association for Computing Machinery. ISBN 9798400706141. doi: 10.1145/3651890.3672274. URL https://doi.org/10.1145/3651890.3672274.

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. ChatDev: Communicative agents for software development. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15174–15186, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.810. URL https://aclanthology.org/2024.acl-long.810/.

Vignav Ramesh and Kenneth Li. Communicating activations between language model agents. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=W6RPXUUFic.

Nir Ratner, Yoav Levine, Yonatan Belinkov, Ori Ram, Inbal Magar, Omri Abend, Ehud Karpas, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. Parallel context windows for large language models. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6383–6402, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.352. URL https://aclanthology.org/2023.acl-long.352/.

Xiangyu Shi, Marco Chiesa, Gerald Q. Maguire Jr., and Dejan Kostic. KVComm: Enabling efficient LLM communication through selective KV sharing. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=F7rUng23nw.

Alon Talmor, Ori Yoran, Amnon Catav, Dan Lahav, Yizhong Wang, Akari Asai, Gabriel Ilharco, Hannaneh Hajishirzi, and Jonathan Berant. Multimodal{qa}: complex question answering over text, tables and images. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=ee6W5UgQLa.

Yixuan Tang and Yi Yang. Multihop-RAG: Benchmarking retrieval-augmented generation for multi-hop queries. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=t4eB3zYWBK.

Junlin Wang, Jue WANG, Ben Athiwaratkun, Ce Zhang, and James Zou. Mixture-of-agents enhances large language model capabilities. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=h0ZfDIrj7T.

Minzheng Wang, Longze Chen, Fu Cheng, Shengyi Liao, Xinghua Zhang, Bingli Wu, Haiyang Yu, Nan Xu, Lei Zhang, Run Luo, Yunshui Li, Min Yang, Fei Huang, and Yongbin Li. Leave no

document behind: Benchmarking long-context LLMs with extended multi-doc QA. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 5627–5646, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.322. URL https://aclanthology.org/2024.emnlp-main.322/.

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. Autogen: Enabling next-gen LLM applications via multi-agent conversations. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=BAakY1hNKS.

An Yang et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

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

Yujia Zheng, Zhuokai Zhao, Zijian Li, Yaqi Xie, Mingze Gao, Lizhu Zhang, and Kun Zhang. Thought communication in multiagent collaboration. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=tq9lyV9Cml.

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

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. GPTSwarm: Language agents as optimizable graphs. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=uTC9AFXIhg.

Jiaru Zou, Ruizhong Qiu, Gaotang Li, Xiyuan Yang, Katherine Tieu, Pan Lu, Ke Shen, Hanghang Tong, Yejin Choi, Jingrui He, James Zou, Mengdi Wang, and Ling Yang. Latent collaboration in multi-agent systems. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=syG9I9ofd8.

A Appendix

A.1 Baselines Implementation Details

Unless otherwise stated, all baseline experiments use the same frozen Qwen3-1.7B or Qwen3-4B backbone, evaluation examples, task prompts, evidence assignment, and greedy decoding procedure as MMA. For two-stream methods, evidence is distributed using the same token-balanced assignment, with branch 1 serving as the sender and branch 0 as the receiver and decoder. Loong allows 20K input tokens per stream and 40K for concatenation; MultiHop-RAG and HotpotQA use their matched task-specific context limits. Generation limits are 512 tokens for benchmarks, except where a baseline-specific protocol requires an intermediate message. All latency measurements include communication, aggregation, and final generation. Because evaluation uses one GPU, sequential worker calls are included in Text MAS latency.

Concatenation. We concatenate the evidence assigned to both streams into one prompt and evaluate a single frozen model. Documents retain their deterministic benchmark ordering. The model receives the same task instruction as the communication methods and directly generates the final answer. Concatenation introduces no trainable parameters or intermediate communication.

Text MAS. We implement a hierarchical two-worker text-based multi-agent baseline (Zhuge et al., 2024; Zou et al., 2026). Each worker independently receives the question and one token-balanced evidence subset, then produces a structured evidence-grounded message. A final summarizer receives the original question and both worker messages, but cannot access the original documents. The summarizer merges the communicated findings and produces the benchmark-specific answer format. The workers and summarizer use the same frozen backbone, and no parameters are trained.

LatentMAS. LatentMAS uses directional latent communication from the sender to the receiver (Zou et al., 2026). The sender first processes its private evidence and performs ten latent reasoning steps. In the homogeneous-model experiments, its complete layer-wise KV cache, including the latent reasoning states, is passed to the receiver. The receiver processes its own private prompt while attending to this communicated cache and then generates the answer. The receiver's cache depth is aligned with the sender when necessary. LatentMAS is evaluated without task-specific training. In heterogeneous experiments, sender latent states are mapped through the shared vocabulary into the receiver embedding space and inserted as a soft prefix instead of transmitting dimension-incompatible raw KV tensors.

KVComm. KVComm communicates a sparsified sender KV cache to the receiver (Shi et al., 2026). We use the deterministic Gaussian layer-selection protocol from the reference implementation: 30% of transformer layers are selected according to a Gaussian prior centered on the middle layer with $\sigma = 10$. The complete sender KV tensors are retained at the selected layers and at layer 0. At every unselected layer, only the first attention-sink token is retained. The receiver then decodes using this uneven communicated cache. KVComm is directional, introduces no trainable parameters, and is applicable only when sender and receiver KV geometries are compatible.

Activation Communication (AC). AC performs one-shot communication through an intermediate residual activation (Ramesh & Li, 2025). Both models independently process their evidence until decoder layer 26. The sender's final prompt-token activation then replaces the receiver's corresponding activation, implementing $f(\mathbf{a}, \mathbf{b}) = \mathbf{a}$. The receiver completes the remaining layers and generates the answer alone. Homogeneous AC requires no training or additional parameters. For the heterogeneous Qwen3-1.7B-to-Qwen3-4B setting, we learn a bias-free linear map from the 2048-dimensional sender activation to the 2560-dimensional receiver space. The map is calibrated on 3,072 C4 sentences, with 512 held out, using activation MSE, Adam with learning rate 10^{-3}, and ten epochs; the mapped activation then replaces the receiver activation at layer 26.

Cache-to-Cache (C2C). C2C uses branch 0 as the Receiver and branch 1 as the Sharer (Fu et al., 2026). The two private prompts are left-padded onto a shared physical cache axis and use identical RoPE position indices. At every transformer layer, a learned projector maps the Sharer's key and value tensors into the Receiver's KV space and adds gated projected residuals to the Receiver cache.

Decoding attends to the union of valid Receiver and Sharer cache positions, while padded Sharer positions are masked. We adapt the official C2C implementation using three-layer key and value projectors with hidden and intermediate widths of 1024, dropout 0.1, and Gumbel-sigmoid gates whose temperature is annealed from 1.0 to 0.001. Projectors are applied in 256-token chunks for long-context memory efficiency. Both base models remain frozen and only the C2C projectors are optimized with answer cross-entropy.

Parallel-Synthesis. We adapt Parallel-Synthesis to the same two-stream evidence partitions used by the other communication baselines. Each worker receives the shared question and one disjoint evidence subset, generates a task-conditioned message, and exposes the layer-wise KV states corresponding to its generated tokens (Liu et al., 2026). The worker prompt caches themselves are not transmitted. We re-encode the cached keys to a common RoPE interval and apply a metadata-conditioned affine mapper that predicts per-layer, per-KV-head, and per-channel scale and bias parameters for both keys and values. The calibrated worker caches are concatenated along the sequence dimension and supplied as the past KV cache of a question-only synthesizer, which does not directly receive the original documents or worker messages as text. Following the proposed training scheme, the Qwen3 backbone is frozen, while the cache mapper and rank-16 synthesizer LoRA adapters on the query, key, value, and output projections are optimized using answer cross-entropy.

Training configuration. All backbone parameters are frozen, and only the MMA mixer, latent-fusion aggregator, and heterogeneous-width projectors (when required) are optimized. We train with next-token cross-entropy on the shared output distribution using AdamW, a learning rate of 3×10^{-5}, weight decay of 0.01, and BF16 precision. Training uses a per-device batch size of one, with documents distributed across branches using the same assignment policy as evaluation. Mixer gates are initialized with a bias of -3 and output projections are zero-initialized so that training begins close to the independent frozen backbones.

Trainable parameters. For Qwen3-1.7B and Qwen3-4B, respectively, MMA introduces 146.94M and 229.54M trainable parameters, compared with 529.23M and 680.44M for C2C and 21.23M and 30.84M for Parallel Synthesis. C2C's larger parameter count reflects its layer-wise KV projection and fusion modules. Parallel Synthesis achieves a smaller count through lightweight cache calibration and low-rank adaptation of an additional pretrained LLM synthesizer, rather than learning a standalone aggregation module. In our evaluated implementation, this design relies on compatible worker–synthesizer KV geometry and does not support the heterogeneous configurations considered here. Its small trainable parameter count therefore does not capture the additional synthesizer computation or imply broader applicability. MMA instead allocates trainable capacity to cross-branch mixing and latent fusion, using fewer parameters than C2C while avoiding a separate LLM synthesizer and supporting heterogeneous branches.

A.2 Effect of Synthesizer LoRA in Parallel-Synthesis

The original Parallel-Synthesis formulation jointly trains a cache mapper and a synthesizer-side LoRA adapter. The cache mapper calibrates independently generated branch caches, while LoRA adapts the synthesizer's attention projections to the resulting non-sequential cache interface (Liu et al., 2026). In contrast, MMA freezes every backbone parameter and trains only its communication and aggregation modules. Thus, comparing the complete Parallel-Synthesis model against MMA is not strictly capacity-matched: Parallel-Synthesis can directly adapt the receiver's attention behavior, whereas MMA cannot modify either branch model.

To isolate the contribution of receiver adaptation, we retrain Parallel-Synthesis after disabling synthesizer LoRA. This *mapper-only* variant retains the same data splits, worker outputs, cache mapper, optimization schedule, evaluation protocol, and checkpoint-selection policy. Only the cache mapper remains trainable. Table 5 compares the independently selected best checkpoints of the two variants.

Removing LoRA degrades performance in all six settings. The largest reductions occur on Qwen3-1.7B MultiHop-RAG (-7.67 points) and Loong (-9.31 F1), showing that the reported Parallel-Synthesis performance depends substantially on adapting the synthesizer rather than cache mapping alone.

Table 5: Effect of synthesizer LoRA in the task-specific Parallel-Synthesis baseline. For Loong, performance is reported as Perfect accuracy / QA-pair F1; the other benchmarks report strict exact-match accuracy. All performance values are percentages. Δ is mapper-only minus mapper+LoRA. Latency is measured end-to-end in seconds per example.

Model	Benchmark	Mapper + LoRA		Mapper only		Δ Perf.
		Perform ↑	Latency ↓	Perform ↑	Latency ↓	
Qwen3-1.7B	Loong	6.25 / 23.71	9.165	2.50 / 14.40	9.485	$-3.75/-9.31$
	MultiHopRAG	78.67	2.987	71.00	2.822	-7.67
	HotpotQA	54.00	0.703	50.00	0.611	-4.00
Qwen3-4B	Loong	15.00 / 38.67	10.237	7.50 / 32.70	9.129	$-7.50/-5.97$
	MultiHopRAG	82.33	4.882	78.33	4.026	-4.00
	HotpotQA	60.67	1.350	59.00	0.870	-1.67

For fidelity to the original method, we report mapper+LoRA Parallel-Synthesis in the main comparison. Nevertheless, this ablation clarifies that it benefits from task-specific receiver adaptation unavailable to MMA. Its comparison with a fully frozen-backbone method should therefore be interpreted as an end-to-end method comparison rather than a strictly controlled comparison of communication mechanisms.

A.3 Expansion on Branch Number

Evidence assignment. Each evidence document is kept intact and assigned to exactly one branch, without splitting or duplication. When the number of branches exceeds the number of documents, the remaining branches receive the task prompt but no evidence. We randomize which branch identities are empty to prevent fixed branch positions from becoming systematically associated with missing evidence. The *Empty* columns in Table 6 report the resulting mean number of evidence-free branches per example.

Table 6: Qwen3-1.7B MMA branch-count comparison on Loong, MultiHop-RAG, and HotpotQA. Performance values are percentages, latency is measured in seconds per example, and Empty denotes the mean number of evidence-free branches per example.

Branch	Loong				MultiHop-RAG			HotpotQA		
	Perfect ↑	F1 ↑	Empty	Lat. ↓	EM ↑	Empty	Lat. ↓	EM ↑	Empty	Lat. ↓
2	15.00	38.05	0.0	5.470	**80.33**	0.0	**0.215**	**56.67**	0.0	**0.193**
3	17.50	38.07	0.1	**3.790**	75.33	0.8	0.249	55.67	1.0	0.275
4	16.25	39.69	0.3	5.481	76.67	1.8	0.286	52.33	2.0	0.369
5	16.25	38.58	0.6	6.543	75.67	2.8	0.340	52.00	3.0	0.513
6	**18.75**	**42.74**	1.0	8.642	78.00	3.8	0.379	51.00	4.0	0.540
7	15.00	36.28	1.4	9.175	72.00	4.8	0.390	51.33	5.0	0.683
8	12.50	30.13	1.9	13.968	74.33	5.8	0.441	52.33	6.0	0.722

The complete results reinforce the context-dependent scaling pattern discussed in Section 4.3. Loong benefits from additional branches until six streams, where the reduction in per-branch context outweighs the cost of communicating and aggregating more representations. MultiHop-RAG and HotpotQA contain substantially fewer documents, causing the number of empty branches and latency to increase without a corresponding reduction in the evidence processed by non-empty branches. Branch expansion should therefore be matched to the amount and structure of available evidence rather than treated as uniformly beneficial.

Table 7: Top-1 private-key recovery under different KV-cache probe designs. All values are percentages; random chance is 1.56%.

Method	Oracle-position MLP	Full-sequence MLP	Sliding-window CNN
C2C	100.00	24.00	100.00
LatentMAS	100.00	20.33	100.00
KVComm	99.33	2.00	76.00
Random chance	1.56	1.56	1.56

A.4 Communication-Privacy

We construct a balanced classification problem containing 64 two-character private keys. Probe models are fitted on 1,955 MultiHop-RAG training examples and evaluated on a disjoint 300-example set. The random Top-1 accuracy is $1/64 = 1.56\%$. The victim communication models remain frozen throughout probe training.

A.4.1 Fixed-size communication.

MMA is probed on the current-token receiver embedding transmitted at its four synchronization layers. AC is probed on the layer-26 final-token activation that it transmits to the receiver. For each method, we train both a linear classifier and a two-layer MLP and report the stronger Top-1 result, representing the stronger validated attacker.

A.4.2 Sequence-valued KV communication.

C2C, LatentMAS, and KVComm transmit variable-length cache representations. We therefore use a position-free sliding-window CNN that searches the complete transmitted sequence. C2C is represented by its projected receiver-side KV update, LatentMAS by the sender's prompt-derived KV states together with ten latent communication states, and KVComm by its selected-layer KV payload.

For a matched feature budget, we retain four communication layers and project each layer to 256 dimensions using a fixed Rademacher projection, producing a 1,024-dimensional feature at every token position. The probe first applies token-wise layer normalization and a learned linear bottleneck with a GELU activation. Parallel one-dimensional convolutions with kernel sizes $\{1, 2, 3, 5\}$ then detect local key-bearing patterns. Their outputs are concatenated and classified into 65 classes: 64 key classes and one background class.

During training, each positive window is centered on the private-key span and is accompanied by four randomly sampled background windows from the same sequence. At evaluation time, the CNN slides over every transmitted position without receiving the key location or any absolute positional feature. Evidence for each key is aggregated using Top-2 pooling over all positions, and the highest-scoring key is returned. Key spans are used after prediction only to measure localization accuracy.

We repeat CNN training after randomly permuting the key labels among training examples while keeping the representations, architecture, and evaluation set unchanged. Accuracy near the 1.56% chance level confirms that successful recovery depends on key information encoded in the communicated representation rather than class imbalance, source ordering, or probe memorization.

Comparison with alternative probe designs. Table 7 compares the sliding-window CNN against two alternatives. The oracle-position MLP receives the true key-token positions during both training and evaluation. The full-sequence MLP probe scores tokens independently with an MLP and aggregates scores across the full sequence using length-normalized log-sum-exp pooling. The CNN instead detects local multi-token patterns and receives no key positions at evaluation. It further the risk of privacy leakage persists at no matter what probe algorithm.

Table 8: Communication privacy on MultiHop-RAG with Qwen3-1.7B. Probe leakage reports top-1 key-recovery accuracy; text leakage reports exact key disclosures out of 300 examples.

Method	Probe Leakage		Text Leakage		
	Probed communication	Accuracy (%) ↓	Direct ↓	Indirect ↓	Structured ↓
MMA	Receiver token embedding	2.00	0	0	0
Parallel Synthesis	Mapped worker KV payload	6.33	3	2	2
C2C	Projected KV payload	100.00	0	0	0
AC	Final-token activation	44.33	0	0	0
LatentMAS	Full KV payload	100.00	251	271	239
KVComm	Selected-layer KV payload	76.00	0	0	0
Text MAS	Private-worker message	–	5	5	5
Concatenation	Raw-token input	–	179	63	156

A.4.3 PRIVACY

Privacy protection. MMA is intended to communicate information for the shared task without exposing unrelated private information in individual branches. We evaluate this property on MultiHop-RAG by inserting a private record of two-character key in the sender context and an additional extraction request in the receiver prompt. We use three receiver-side attacks: a *direct* request for the key, an *indirect* request presented as collaborator content, and a *structured* request for the key in JSON. Each attack precedes the original task instruction. We measure two distinct forms of leakage.

- **Text leakage** counts an attack as successful when the exact two-character key appears in the final generated answer.
- **Probe leakage** measures whether the key can be predicted from the communicated representation.

Two-layer MLP classifiers are applied for the fixed-size messages of MMA and AC. And we use a CNN for the sequence-valued KV payloads of C2C, LatentMAS, KVComm and Parallel Synthesis. Text MAS and Concatenation is not probed because its communicated message is already directly observable as text. Full probe specifications are provided in Appendix A.4.

Table 8 shows that final-output inspection alone is insufficient for evaluating communication privacy. C2C and KVComm produce no exact key disclosures, but their communicated KV representations allow near-perfect probe recovery. LatentMAS exposes the key both internally and in its final output, while concatenation frequently reproduces the raw key. In contrast, MMA achieves probe accuracy close to random chance and produces no exact disclosures under any attack. These results suggest that MMA's sparse, gated receiver update communicates substantially less recoverable branch-local information than methods that expose full or projected KV states.

Robustness under attack. Table 9 evaluates task accuracy after inserting direct, indirect, or structured extraction instructions into the receiver prompt. MMA is the only method that combines high clean utility with consistently strong robustness, retaining 70.67–74.00% accuracy across all three attacks. Parallel Synthesis remains competitive under the indirect and structured attacks but drops by 29.67 points under the direct attack. Other baselines either degrade substantially or begin from much lower canary-only accuracy. These results indicate that MMA maintains a stronger and more stable utility–robustness trade-off under adversarial communication.

A.5 COMPONENT ABLATION

We ablate the two learned components of MMA: intermediate cross-branch mixing and latent-fusion output aggregation. We additionally replace directional mixing with non-directional mixing while retaining the remaining architecture. All variants use Qwen3-1.7B with two branches on MultiHop-RAG, the same 1,955-example training set and 300-example evaluation set.

Intermediate mixing provides the largest individual improvement, raising strict EM from 32.67 to 73.00 even without learned output fusion. Latent fusion alone yields only a modest gain, but im-

Table 9: Task accuracy robustness under 3 receiver-side privacy attacks on MultiHop-RAG evaluation with Qwen3-1.7B. *Clean* contains no local key, whereas *canary only* inserts the key without an extraction request. Parentheses report the percentage-point change from the canary-only control.

Method	Before attack		Under attack		
	Clean ↑	Canary only ↑	Direct ↑	Indirect ↑	Structured ↑
MMA	**80.33**	**79.67**	**72.67** (−7.00)	70.67 (−9.00)	**74.00** (−5.67)
Parallel Synthesis	78.67	78.00	48.33 (−29.67)	**73.00** (−5.00)	72.33 (−5.67)
C2C	70.67	70.67	39.67 (−31.00)	59.00 (−11.67)	66.00 (−4.67)
AC	50.00	47.33	47.00 (−0.33)	47.33 (+0.00)	48.00 (+0.67)
LatentMAS	44.67	52.33	2.67 (−49.67)	3.00 (−49.33)	2.33 (−50.00)
KVComm	58.33	48.33	3.00 (−45.33)	4.00 (−44.33)	6.00 (−42.33)
Text MAS	57.67	57.00	53.00 (−4.00)	46.33 (−10.67)	51.67 (−5.33)

Table 10: Component ablation on MultiHop-RAG. Strict EM is reported as a percentage, latency is measured in seconds per example, and parameter counts include only trainable communication and aggregation modules.

Configuration	Intermediate Mixing	Aggregation	Strict EM ↑	Lat. ↓	Params
Neither	None	Logit sum	32.67	0.583	0.0M
Fusion only	None	Latent fusion	35.00	0.617	63.0M
Mixer only	Directional	Logit sum	73.00	0.257	84.0M
Non-directional MMA	Non-directional	Latent fusion	77.67	**0.210**	130.1M
Full MMA	Directional	Latent fusion	**80.33**	0.215	146.9M

proves the directional mixer by a further 7.33 points when the two components are combined. Their positive factorial interaction is 5.00 points, suggesting that latent fusion is most effective after intermediate communication has produced informative cross-branch states. Directional mixing provides an additional 2.67-point gain over the otherwise matched non-directional variant.

A.6 Synchronization Schedule Ablation

We next vary the number and placement of synchronization layers while retaining the full directional mixer and latent-fusion aggregator. The zero-sync configuration retains latent-fusion output aggregation but removes all intermediate communication. The four-layer early and late schedules have identical parameter counts, isolating the effect of communication depth.

Table 11: Synchronization-schedule ablation on MultiHop-RAG. Layers are zero-indexed. The zero-sync configuration uses latent-fusion aggregation without intermediate mixing.

Syncs	Placement	Layers	Strict EM ↑	Lat. ↓	Params
0	–	–	35.00	0.617	63.0M
1	Late	27	52.00	0.286	84.0M
2	Late	19, 27	64.00	0.230	104.9M
4	Early	3, 7, 11, 15	78.67	**0.207**	146.9M
4	Late	15, 19, 23, 27	**80.33**	0.215	146.9M
7	Even	3, 7, 11, 15, 19, 23, 27	79.00	0.208	209.9M

Performance increases substantially as the communication budget grows from zero to four synchronization points. Denser communication does not provide a further benefit: the seven-layer schedule uses 42.9% more trainable parameters than the four-layer schedule while reducing strict EM by 1.33 points. At a fixed four-layer budget, late synchronization outperforms early synchronization by 1.67 points. These results support the sparse late-layer schedule used in the main experiments as an effective balance between communication capacity and model complexity.

A.7 Robustness to Branch Identity

MMA uses directional gates, learned branch embeddings, and an anchored latent-fusion head. These components could introduce a preference for a particular branch index, allowing performance to depend on which evidence subset is assigned to that branch. We test this possibility using the retained Qwen3-1.7B two-branch MultiHop-RAG checkpoint. Evidence is first divided into the same token-balanced buckets used in the main evaluation. We then keep each bucket intact while changing only its destination branch. We compare the original assignment, a complete reversal of both branch identities, and five deterministic per-example permutations. No parameters are updated during this study.

Table 12: Branch-identity robustness on the fixed 300-example MultiHop-RAG evaluation set. Match denotes exact output agreement with the original branch assignment. The half-swap condition reports mean and population standard deviation over five seeds.

Context assignment	**Canonical ↑**	**Strict EM ↑**	**Match ↑**	**Lat. ↓**
Original assignment	81.33	80.33	100.00	0.213
Reverse branch assignment	81.00	80.00	90.67	0.210
Swap branches for half of context	81.67 ± 0.56	80.67 ± 0.56	95.53 ± 0.96	0.209 ± 0.001

The complete reversal changes 28 of 300 generated outputs, showing that the architecture is not strictly permutation-invariant. However, these changes are nearly neutral with respect to correctness: 11 originally correct predictions become incorrect, while 10 originally incorrect predictions become correct. Consequently, strict EM decreases by only 0.33 points. Across the five seeded permutations, strict EM ranges from 80.00 to 81.67 and averages slightly above the identity result. MMA's performance therefore does not rely on a privileged evidence-to-branch assignment, despite its branch-specific communication and fusion parameters.

A.8 Causal Use of Distributed Evidence

To show whether the prediction depends on both streams, we conduct a paired evidence-removal intervention on the fixed 300-example MultiHop-RAG evaluation set. Starting from the reported Qwen3-1.7B two-branch checkpoint, we first reproduce the intact-evidence result and then remove all documents from one branch after the original token-balanced assignment. This process keeps the surviving branch's evidence byte-identical and leaves the question, checkpoint, prompt template, and decoding parameters unchanged. We evaluate removal of each fixed branch.

Table 13: Causal evidence-removal intervention on the fixed 300-example MultiHop-RAG evaluation set with Qwen3-1.7B MMA. Removal is applied after token-balanced assignment, so the surviving branch, question, and decoding configuration are unchanged. Δ is measured relative to the intact-evidence control.

Condition	**Canonical ↑**	**Strict EM ↑**	**Δ Strict**	**Removed docs**
Intact evidence	81.33	80.33	0.00	0.00
Drop branch 0	46.00	45.67	−34.67	1.06
Drop branch 1	47.67	47.00	−33.33	1.18

Without retraining, removing the context of branch 0 or branch 1 reduces performance by 34.67 or 33.33 percentage points respectively. These large drops show that trained MMA relies strongly on evidence from both branches. In contrast, the smaller gap for the retrained empty-peer control suggests that supervision can recover substantial performance from partial evidence on MultiHop-RAG. Therefore, strong empty-peer performance on MultiHop-RAG only does not imply that MMA's gains arise from adaptation alone or that cross-branch communication is unnecessary.

A.9 CONTEXT-LENGTH STRESS TEST

We evaluate robustness to increasing irrelevant context on the fixed 300-example MultiHop-RAG set. All supporting documents are retained, and whole-document distractors are added using five fill budgets. A fill budget controls distractor packing rather than truncation: required evidence is never removed and may already exceed the budget. Consequently, the 2K and 4K budgets both remain near the support-only floor.

Contexts are nested within each of three distractor seeds, and every method receives the same evidence under the main token-balanced assignment without retraining. We report strict EM and answer-bearing accuracy, which additionally credits responses from which an accepted answer can be recovered.

Table 14: Matched context-length stress test on MultiHop-RAG. Fill Budget denotes the total-prompt budget used when adding whole-document distractors; mandatory supporting evidence is always retained. Values are mean ± population standard deviation over three distractor seeds. Latency is measured in seconds per example.

Method	Fill Budget	Strict EM ↑	Answer-bearing ↑	Latency ↓
Concatenation	2K	55.33 ± 0.00	55.33 ± 0.00	0.379 ± 0.001
Text MAS	2K	57.67 ± 0.00	57.67 ± 0.00	3.207 ± 0.000
LatentMAS	2K	40.00 ± 0.00	42.33 ± 0.00	0.717 ± 0.000
KVComm	2K	58.33 ± 0.00	59.00 ± 0.00	0.331 ± 0.000
C2C	2K	67.33 ± 0.00	67.67 ± 0.00	0.475 ± 0.000
AC	2K	50.00 ± 0.00	54.33 ± 0.00	0.327 ± 0.000
Parallel Synthesis	2K	75.33 ± 0.00	77.00 ± 0.00	1.605 ± 0.000
MMA	2K	$\mathbf{79.33 \pm 0.00}$	$\mathbf{79.33 \pm 0.00}$	$\mathbf{0.207 \pm 0.002}$
Concatenation	4K	49.44 ± 0.16	55.44 ± 0.16	0.379 ± 0.001
Text MAS	4K	57.67 ± 0.00	57.67 ± 0.00	3.203 ± 0.002
LatentMAS	4K	39.78 ± 0.16	42.11 ± 0.16	0.718 ± 0.001
KVComm	4K	58.33 ± 0.00	59.00 ± 0.00	0.331 ± 0.000
C2C	4K	67.22 ± 0.16	67.44 ± 0.16	0.476 ± 0.000
AC	4K	49.89 ± 0.16	54.22 ± 0.16	0.328 ± 0.001
Parallel Synthesis	4K	75.33 ± 0.00	77.00 ± 0.00	1.606 ± 0.001
MMA	4K	$\mathbf{79.22 \pm 0.16}$	$\mathbf{79.22 \pm 0.16}$	$\mathbf{0.206 \pm 0.001}$
Concatenation	8K	45.67 ± 1.25	52.22 ± 0.68	0.438 ± 0.004
Text MAS	8K	55.67 ± 0.98	55.67 ± 0.98	3.216 ± 0.007
LatentMAS	8K	37.67 ± 1.44	40.44 ± 1.29	0.785 ± 0.011
KVComm	8K	53.67 ± 1.36	54.44 ± 1.37	0.395 ± 0.006
C2C	8K	65.00 ± 0.72	65.11 ± 0.87	0.579 ± 0.001
AC	8K	45.67 ± 1.70	49.67 ± 1.66	0.403 ± 0.009
Parallel Synthesis	8K	74.22 ± 0.42	75.22 ± 0.16	1.694 ± 0.015
MMA	8K	$\mathbf{78.11 \pm 0.63}$	$\mathbf{78.11 \pm 0.63}$	$\mathbf{0.238 \pm 0.001}$
Concatenation	16K	30.00 ± 0.54	38.56 ± 0.68	0.700 ± 0.011
Text MAS	16K	47.22 ± 1.66	47.33 ± 1.70	3.437 ± 0.026
LatentMAS	16K	20.22 ± 0.16	22.78 ± 0.83	1.560 ± 0.022
KVComm	16K	42.11 ± 1.10	43.33 ± 1.19	0.759 ± 0.026
C2C	16K	63.44 ± 0.16	63.56 ± 0.16	0.961 ± 0.001
AC	16K	38.00 ± 3.31	43.78 ± 2.53	0.590 ± 0.021
Parallel Synthesis	16K	$\mathbf{73.00 \pm 1.25}$	$\mathbf{73.89 \pm 0.83}$	2.665 ± 0.038
MMA	16K	72.89 ± 1.10	73.00 ± 1.25	$\mathbf{0.367 \pm 0.002}$
Concatenation	32K	22.44 ± 0.16	32.78 ± 0.96	1.257 ± 0.008
Text MAS	32K	50.67 ± 0.27	50.67 ± 0.27	4.009 ± 0.122
LatentMAS	32K	21.11 ± 1.13	23.11 ± 1.37	2.526 ± 0.020
KVComm	32K	41.00 ± 0.98	42.89 ± 1.40	1.861 ± 0.021
C2C	32K	65.00 ± 1.19	65.11 ± 1.13	2.050 ± 0.010
AC	32K	25.22 ± 0.57	32.56 ± 1.66	1.025 ± 0.010
Parallel Synthesis	32K	68.89 ± 0.31	69.33 ± 0.27	4.794 ± 0.122
MMA	32K	$\mathbf{71.67 \pm 1.19}$	$\mathbf{71.67 \pm 1.19}$	$\mathbf{0.743 \pm 0.002}$

Table 14 shows that MMA is strongest at four of five fill budgets and trails Parallel Synthesis by only 0.11 points at 16K. From the support-only floor to the largest condition, MMA's strict EM decreases from 79.33 to 71.67, whereas concatenation falls from 55.33 to 22.44. At 22.43K realized tokens, MMA also remains ahead of C2C (65.00) and Parallel Synthesis (68.89), while retaining the lowest latency at 0.743 seconds per example.