STATES AS MESSAGES: LATENT COMMUNICATION FOR STATE SPACE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

State space models (SSMs) are increasingly used in language modeling, and when model instances hold different contexts, one may need information that another has already processed. Transferring this information through natural-language summaries requires autoregressive generation at the source and re-encoding at the receiver. Latent communication can reduce this overhead by directly exchanging internal representations such as the fixed-size recurrent states of SSMs. Using these states for communication requires the receiver to combine the source's state with its own while retaining information from both contexts. We introduce a dynamics-grounded fuser for pretrained multilayer SSMs. The fuser learns a lightweight per-channel correction to a single-layer state decomposition while keeping the base model weights frozen. Across five scales each of Mamba and Mamba-2, our method closes up to 75% and 84% of the receiver-only to full-context perplexity gap on WikiText-103, respectively. The method generalizes across sequence lengths, partition ratios, and domains without retraining. On separate-document QA with HotpotQA and MuSiQue, task-specific state communication achieves exact-match scores comparable to natural-language summaries and closes up to 74% of the output-distribution gap to the full-context teacher. On HotpotQA, state communication runs approximately $1.6\times$ faster end to end than the summary baselines on a single GPU, with no generated message tokens.

1 INTRODUCTION

State space models (SSMs) have shown strong language modeling performance in models such as Mamba and Mamba-2 (Gu & Dao, 2024; Dao & Gu, 2024; Zuo et al., 2024; Anthony et al., 2024; Lahoti et al., 2026). SSM layers are also used in hybrid models such as Nemotron, which combine them with attention layers (Blakeman et al., 2025). When multiple instances of these models are used together, they may process separate contexts but need information from one another to complete a task.

For example, in a multi-agent system, a source may have processed one document while a receiver holds another document and needs to answer a question that requires information from both. Natural language provides a flexible, model-agnostic interface for transferring this information (Hong et al., 2024; Wu et al., 2024; Li et al., 2023). However, sending the original text requires the receiver to process the document again, while sending a summary requires autoregressive generation at the source followed by re-encoding at the receiver. Latent communication can reduce this overhead by directly transferring the internal representations the source has already computed.

Existing latent communication methods exchange KV caches (Yao et al., 2025; Yang et al., 2025; Fu et al., 2026), which SSMs do not maintain. An SSM instead compresses its processed context into a fixed-size recurrent state, making it a communication medium whose size does not grow with context length. To communicate through these states, the receiver must incorporate the source's information while retaining information from its own context. Simply adding or averaging two independently computed states does not account for how the source contribution would evolve as the receiver processes its context.

We propose a latent communication method for instances of the same frozen pretrained SSM. Our method uses a state decomposition obtained directly from the SSM recurrence as an analytic fusion prior. This decomposition applies the receiver's accumulated decay to the source state before adding

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

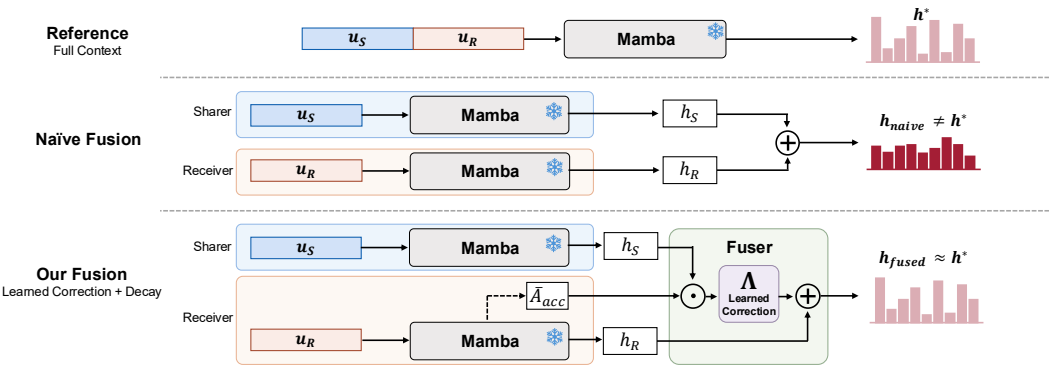


Figure 1: Full-context processing, naive fusion, and our dynamics-grounded state fusion. The full-context reference processes both contexts in sequence, while fusion combines states computed independently by the source and receiver. Naive fusion does not account for the SSM dynamics. Our method applies the receiver's accumulated decay and a learned per-channel correction to the source state before adding the receiver state, while keeping all base model weights frozen.

the receiver state. The rule is exact for a single SSM recurrence with unchanged inputs and transition coefficients. In a multilayer model, processing the contexts independently changes the inputs received by later layers, making its layerwise application approximate. We adapt the rule by learning a lightweight per-channel correction while keeping all base model weights frozen (Figure 1).

Across five scales each of Mamba and Mamba-2, our method closes up to 75% and 84% of the receiver-only to full-context perplexity gap on WikiText-103, respectively, and improves ARC-Easy and HellaSwag accuracy at every scale. The learned correction generalizes across sequence lengths, partition ratios, and domains without retraining. On separate-document QA with HotpotQA and MuSiQue, task-specific state communication achieves exact-match scores comparable to natural-language summaries and closes up to 74% of the output-distribution gap to the full-context teacher. On HotpotQA, state communication runs approximately $1.6\times$ faster end to end on a single GPU, with no generated message tokens. Linear probes also show that source words absent from the receiver's context become detectable in the fused state.

Our contributions are the following.

1. We formulate fixed-size hidden-state exchange as a directed latent communication channel between instances of the same pretrained SSM.
2. We introduce a dynamics-grounded fuser that learns a lightweight correction to the single-layer state decomposition for pretrained multilayer models.
3. We demonstrate perplexity and zero-shot accuracy improvements across five scales each of Mamba and Mamba-2, with further experiments showing generalization across sequence lengths, partition ratios, and domains without retraining.
4. We show that state communication achieves exact-match scores comparable to natural-language summaries on HotpotQA and MuSiQue, while running approximately $1.6\times$ faster end to end on a single GPU.

2 Background and Related Work

State Space Models. State space models map an input sequence $u_k \in \mathbb{R}$ to an output $y_k \in \mathbb{R}$ through a latent state $h_k \in \mathbb{R}^N$. A discretized linear SSM follows the recurrence

$$h_k = \bar{A}h_{k-1} + \bar{B}u_k, \qquad y_k = Ch_k, \tag{1}$$

where $\bar{A}$ and $\bar{B}$ are the discretized state transition and input matrices, and C maps the state to the output. Mamba (Gu & Dao, 2024) uses a diagonal state matrix A, following S4D (Gu et al., 2022),

and introduces selectivity by making B_k, C_k, and the discretization step Δ_k functions of the current input. The resulting recurrence is

$$h_k = \bar{A}_k h_{k-1} + \bar{B}_k u_k, \qquad y_k = C_k h_k, \tag{2}$$

where $\bar{A}_k = \exp(\Delta_k A)$ varies across timesteps even though A remains fixed. Mamba-2 (Dao & Gu, 2024) retains the selective recurrent formulation while using a scalar-times-identity state transition within each head.

SSM States and Composition. An SSM state retains information from the processed context for use in subsequent generation. A Mamba block maintains an SSM state $h \in \mathbb{R}^{D \times N}$, where D is the number of channels and N is the state dimension per channel. Both Mamba and Mamba-2 maintain SSM states whose size is independent of sequence length. Their diagonal transition structure allows accumulated decays to be applied elementwise, providing the basis for the state decomposition used in our fuser.

State Soup (Pióro et al., 2024) studies recurrence-based state composition for in-context skill learning, retrieval, and mixing. PICASO (Liu et al., 2025) studies permutation-invariant context aggregation in retrieval-augmented generation, and Document Souping (Jafari et al., 2025) applies state averaging to document aggregation. We study context transfer between instances of the same frozen pretrained SSM, where the receiver incorporates the source's independently computed states and continues generation.

Context Transfer Between Language Models. Natural language is a common interface for transferring information between language model instances (Hong et al., 2024; Wu et al., 2024; Li et al., 2023), including settings where models hold different information (Liu et al., 2024; Peng et al., 2025). The source generates a message that the receiver incorporates through its input sequence. This interface is flexible, but requires autoregressive message generation and receiver-side re-encoding even when the source has already processed its context. Latent communication instead transfers internal representations directly, allowing the receiver to reuse information already encoded by the source.

KV Cache Reuse and Fusion. Several works reuse or combine transformer KV caches. PromptCache (Gim et al., 2024) reuses cached representations of prompt modules, CacheBlend (Yao et al., 2025) combines caches with selective token recomputation, KVLink (Yang et al., 2025) uses learned link tokens, and Cache-to-Cache (Fu et al., 2026) learns projections for cross-model cache transfer. These methods operate on KV caches that retain separate entries for individual tokens. SSMs instead compress the processed context into a fixed-size recurrent state, so the same cache-integration operations do not directly apply. Incorporating the source's state into the receiver's state requires accounting for the SSM dynamics.

3 METHOD

3.1 PROBLEM FORMULATION

Consider a source and a receiver running the same frozen pretrained SSM with L layers. From zero initial SSM states, the source processes $u_S = [u_1, \ldots, u_{T_S}]$, while the receiver independently processes $u_R = [u_{T_S+1}, \ldots, u_T]$. This produces terminal states $h_S^{(\ell)}$ and $h_R^{(\ell)}$ at each layer, together with the receiver's accumulated decay $\bar{A}_{\text{accum}}^{(\ell)}$.

The source transmits its terminal states across all layers. After both contexts have been processed, the receiver combines these with its own states and accumulated decays, then continues generation from the fused states $h_R'^{(\ell)}$. Our objective is to approximate the continuation distribution of the same model processing $[u_S; u_R]$ jointly. Figure 2 shows the fusion operation and its multilayer application. We omit the layer superscript in the following decomposition.

3.2 SINGLE-LAYER STATE DECOMPOSITION

We use the single-layer state decomposition (Pióro et al., 2024) as an analytic prior. Starting from $h_0 = 0$ and unrolling Eq. 2 gives

$$h^* = \sum_{k=1}^{T} \left(\prod_{j=k+1}^{T} \bar{A}_j \right) \odot (\bar{B}_k u_k). \tag{3}$$

Splitting at T_S gives

$$h^* = \underbrace{\sum_{k=1}^{T_S} \left(\prod_{j=k+1}^{T} \bar{A}_j \right) \odot (\bar{B}_k u_k)}_{\text{Source contribution}} + \underbrace{\sum_{k=T_S+1}^{T} \left(\prod_{j=k+1}^{T} \bar{A}_j \right) \odot (\bar{B}_k u_k)}_{\text{Receiver contribution}}. \tag{4}$$

The second term is h_R. In the first term, the receiver-side decay is shared across all source contributions. Defining $\bar{A}_{\text{accum}} = \prod_{j=T_S+1}^{T} \bar{A}_j$, we obtain

$$\sum_{k=1}^{T_S} \left(\prod_{j=k+1}^{T} \bar{A}_j \right) \odot (\bar{B}_k u_k) = \bar{A}_{\text{accum}} \odot \underbrace{\sum_{k=1}^{T_S} \left(\prod_{j=k+1}^{T_S} \bar{A}_j \right) \odot (\bar{B}_k u_k)}_{h_S}. \tag{5}$$

Substituting into Eq. 4 yields

$$\boxed{h^* = \bar{A}_{\text{accum}} \odot h_S + h_R.} \tag{6}$$

The full derivation appears in Appendix B.

Fixed-parameter SSMs. For a fixed diagonal transition, as in S4D (Gu et al., 2022),

$$\bar{A}_{\text{accum}}^{(\text{S4D})} = \bar{A}^{T-T_S}, \tag{7}$$

where the power is elementwise. This factor depends only on receiver length, while the final states retain input-dependent information.

Selective SSMs. In Mamba, $\bar{A}_j = \exp(\Delta_j A)$ depends on the input through Δ_j, giving

$$\bar{A}_{\text{accum}}^{(\text{Mamba})} = \prod_{j=T_S+1}^{T} \bar{A}_j \tag{8}$$

The accumulated decay therefore depends on receiver length and content. Mamba-2 shares a scalar decay within each head, which is broadcast over the corresponding state entries.

Computational cost. The receiver's accumulated decay accounts for how the source contribution is transformed over its processing window. Accumulating decay requires $\mathcal{O}(T_R DN)$ operations and $\mathcal{O}(DN)$ additional memory per layer. Once both terminal states are available, fusion requires $\mathcal{O}(DN)$ operations per layer, independent of context length.

3.3 DYNAMICS-GROUNDED STATE FUSION

The decomposition in Eq. 6 is exact for the single SSM recurrence above. In a multilayer model, however, each layer receives the output of the previous layer. When the receiver processes its context alone, these outputs differ from those produced with access to both contexts. Applying the rule at each layer therefore does not exactly recover the full-context states. So we learn a lightweight per-channel correction to account for this difference (Figure 2).

$$h_R'^{(\ell)} = \Lambda^{(\ell)} \odot \bar{A}_{\text{accum}}^{(\ell)} \odot h_S^{(\ell)} + h_R^{(\ell)}. \tag{9}$$

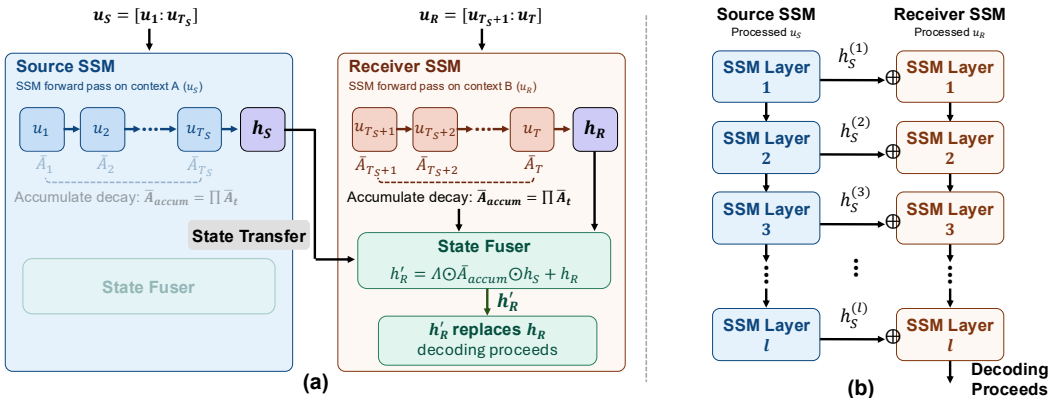


Figure 2: (a) State fusion using independently computed states and receiver-side decay, $h'_R = \Lambda \odot \bar{A}_{\text{accum}} \odot h_S + h_R$. (b) Fusion across layers before continuation. Only the correction parameters are trained, with the base model frozen.

The correction $\Lambda^{(\ell)}$ has the same shape as the SSM state at layer ℓ. It rescales the decayed source contribution before that contribution is added to the receiver's state. We initialize every entry of $\Lambda^{(\ell)}$ to one, so fusion begins from the analytic prior. Only the correction parameters are trained, while the pretrained model weights remain frozen.

For Mamba, $\Lambda^{(\ell)} \in \mathbb{R}^{D \times N}$, giving $D \times N$ parameters per layer and LDN parameters for a model with identical state dimensions across layers. This adds less than 1% of the base model parameters across the evaluated Mamba scales from 130M to 2.8B. For Mamba-2, the same elementwise rule applies in its head-based state layout, with the accumulated decay broadcast over the corresponding state dimensions. In both architectures, the correction contains one learned parameter per SSM state entry.

The elementwise correction follows the diagonal transition structure and preserves the accumulated decay as an input-dependent factor. Empirically, most learned correction values remain close to one in the models analyzed in Section 4.7.

3.4 Training

All base model parameters are frozen, and only the correction parameters $\Lambda^{(\ell)}$ are trained. We train the receiver to approximate the continuation distribution of a full-context model with the same pretrained weights.

Each training example contains source context u_S, receiver context u_R, and a subsequent prediction window $v = [v_1, \ldots, v_M]$. The full-context model processes the concatenation of u_S and u_R. For state communication, the source and receiver process their contexts independently, after which their final states are fused at every layer. The receiver then continues from the fused states over the prediction window.

We minimize the KL divergence between the full-context and fused output distributions,

$$\mathcal{L} = \sum_{t=1}^{M} D_{\text{KL}} \left(p_{\text{full}}^{(t)} \parallel p_{\text{fused}}^{(t)} \right), \tag{10}$$

where $p_{\text{full}}^{(t)}$ and $p_{\text{fused}}^{(t)}$ are the next-token distributions for predicting v_t. Both models receive the same preceding continuation tokens $v_{<t}$, and the loss is evaluated only over the prediction window. Which is a self-distillation setup: the full-context model serves as the teacher, and the fused model is trained to match its continuation distribution. The loss supervises output distributions rather than the fused states directly.

Table 1: State communication across Mamba model sizes. Changes are measured relative to receiver-only. Negative perplexity changes and positive accuracy changes indicate improvement. Accuracy changes are in percentage points. Full-context reference scores appear in Appendix D.1.

		130M	370M	790M	1.4B	2.8B
WikiText	Receiver only	21.42	14.70	12.16	11.00	9.62
	State communication	**20.44**	**14.03**	**11.63**	**10.51**	**9.21**
	Δ PPL	-0.98	-0.67	-0.53	-0.49	-0.41
PG-19	Receiver only	19.03	14.42	12.44	11.50	10.36
	State communication	**18.78**	**14.23**	**12.28**	**11.42**	**10.27**
	Δ PPL	-0.25	-0.19	-0.16	-0.08	-0.09
ARC-Easy	Receiver only	37.58%	40.11%	43.48%	46.93%	50.00%
	State communication	**39.27%**	**43.22%**	**45.29%**	**49.24%**	**52.86%**
	Δ accuracy	+1.69%	+3.11%	+1.81%	+2.31%	+2.86%
HellaSwag	Receiver only	32.66%	42.75%	50.58%	54.14%	60.35%
	State communication	**33.55%**	**43.96%**	**51.98%**	**55.41%**	**61.92%**
	Δ accuracy	+0.89%	+1.21%	+1.40%	+1.27%	+1.57%

4 Experiments & Results

4.1 Experimental Setup

We evaluate five scales each of Mamba and Mamba-2, using Mamba for the detailed ablations and analyses. For each checkpoint, we train separate fusion parameters on WikiText-103 (Merity et al., 2016) with the base model frozen. Each 1024-token sequence contains 384 source tokens, 384 receiver tokens, and a 256-token prediction window. QA uses task-specific fusion parameters (Section 4.3). Training details appear in Appendix C.

We evaluate perplexity on WikiText-103 and PG-19 (Rae et al., 2020), and zero-shot accuracy on ARC-Easy (Clark et al., 2018) and HellaSwag (Zellers et al., 2019) with length-normalized accuracy.

Receiver-only measures performance without communication, while natural-language summaries provide a communication baseline for QA. Full-context provides the joint-processing reference. We also report gap closure,

$$\text{Gap closure} = \frac{\text{Metric}_{\text{receiver}} - \text{Metric}_{\text{ours}}}{\text{Metric}_{\text{receiver}} - \text{Metric}_{\text{full}}}. \tag{11}$$

A value of 0% gap closure indicates no improvement over receiver-only, and 100% indicates full information recovery compared to matching the full-context reference on the reported metric.

4.2 State Communication Across Scales and Architectures

Evaluation across Mamba scales. Table 1 reports results across five Mamba sizes at partition ratio 0.5 with 1024-token sequences. The same fusion design and training procedure are used for each checkpoint, with dimensions adjusted to the model's SSM states.

State communication reduces perplexity relative to receiver-only at all five scales on both WikiText and PG-19. On downstream tasks, ARC-Easy improves by 1.69–3.11% and HellaSwag by 0.89–1.57%. At 2.8B, the gains are 2.86% on ARC-Easy and 1.57% on HellaSwag. Relative to full context processing, the method recovers 67–75% of the WikiText perplexity gap.

Evaluation on Mamba-2. We apply the same communication rule to five Mamba-2 scales. Table 2 reports receiver-only and state-communication scores. State communication reduces perplexity at every scale on both WikiText-103 and PG-19, recovering 76.7–83.7% and 30.4–64.2% of the respective gaps to full context. The learned correction also improves over the analytic prior on both datasets at all five scales (Appendix D.2).

Table 2: State communication across Mamba-2 scales. We report perplexity on WikiText-103 and PG-19 and length-normalized accuracy percentages on the zero-shot tasks. Changes are calculated from the displayed scores relative to receiver-only. Negative perplexity changes and positive accuracy changes indicate improvement.

Dataset	Condition	130M	370M	780M	1.3B	2.7B
WikiText	Receiver only	20.96	14.62	12.13	10.69	9.27
	State communication	**19.80**	**13.82**	**11.47**	**10.14**	**8.81**
	Δ PPL	-1.16	-0.80	-0.66	-0.55	-0.46
PG-19	Receiver only	23.82	18.13	15.79	14.33	12.93
	State communication	**23.66**	**17.87**	**15.54**	**14.09**	**12.72**
	Δ PPL	-0.16	-0.26	-0.25	-0.24	-0.21
ARC-Easy	Receiver only	35.48%	39.44%	43.86%	47.43%	50.34%
	State communication	**39.35%**	**43.77%**	**47.18%**	**52.19%**	**54.17%**
	Δ accuracy	+3.87%	+4.33%	+3.32%	+4.76%	+3.83%
HellaSwag	Receiver only	32.77%	42.65%	50.22%	55.09%	60.58%
	State communication	**33.41%**	**44.56%**	**52.62%**	**57.75%**	**63.35%**
	Δ accuracy	+0.64%	+1.91%	+2.40%	+2.66%	+2.77%

Table 3: Separate-document QA with Mamba-2.8B-instruct. EM and KL closure use 7,402 HotpotQA and 2,417 MuSiQue examples. Overhead includes all communication compute beyond the receiver's forward pass, HotpotQA latency is averaged over 200 examples on one H100, excluding inter-device transmission. NL-short and NL-medium allow up to 32 and 128 summary tokens.

	HotpotQA		MuSiQue		HotpotQA cost	
Method	EM	KL closure	EM	KL closure	Overhead (GFLOPs)	Latency (ms)
Receiver only	0.489	0%	0.163	0%	–	–
NL-short	0.501	28%	0.208	23%	439	278
NL-medium	0.500	28%	0.221	31%	1507	292
State communication	**0.507**	**74%**	**0.221**	**68%**	**0.02**	**180**
Full context (reference)	0.541	–	0.296	–	–	–

On zero-shot tasks, ARC-Easy improves by 3.32–4.76% and HellaSwag by 0.64–2.77%. ARC-Easy gains are larger than those of the similarly sized Mamba models at all five scales, while HellaSwag gains are larger at four of the five scales.

4.3 Separate-Document QA

We evaluate on HotpotQA (Yang et al., 2018) and MuSiQue (Trivedi et al., 2022) in a gold-only two-document setting. The source receives one supporting paragraph, while the receiver receives the other paragraph and the question. We train correction parameters on the community instruction-tuned Mamba-2.8B model `clibrain/mamba-2.8b-instruct-openhermes`. The answer-formatting adapter is shared across all conditions and remains frozen during fuser training. We compare state communication against natural-language summaries with generation limits of 32 and 128 tokens. The source does not receive the question in either the state or summary conditions.

Table 3 reports exact match, KL closure relative to the full-context teacher, and HotpotQA communication overhead and latency. On HotpotQA, state communication reaches 0.507 EM, compared with 0.489 for receiver-only and 0.501 for the strongest summary baseline. On MuSiQue, it reaches 0.221 EM, compared with 0.163 for receiver-only and 0.221 for the strongest summary baseline. KL closure is 74% versus 28% on HotpotQA and 68% versus 31% on MuSiQue. The higher KL closure is consistent with our training objective, which directly matches the full-context output distribution. Exact match provides a separate measure of answer quality, where state communication performs comparably to natural-language summaries at lower computational cost.

Table 4: Baseline comparison on Mamba-130M (partition ratio 0.5, 1024 tokens). Raw perplexity (↓) and accuracy (↑) are reported alongside gap closure. Receiver-only is the no-communication baseline. Full context is shown separately as the joint-processing reference. Sum and Average use the state-pooling operations studied in Document Souping (Jafari et al., 2025), applied here without base-model finetuning. Bold marks the best result among learned fusion methods. MLP-CW is a parameter-matched channel-wise MLP.

		Language Modeling				*Downstream Zero-Shot*					
		WikiText-103		PG-19		ARC-Easy		PIQA		HellaSwag	
Method	Params/layer	PPL	Gap↑	PPL	Gap↑	Acc	Gap↑	Acc	Gap↑	Acc	Gap↑
Receiver only	—	21.42	0%	19.03	0%	37.58%	0%	62.46%	0%	32.66%	0%
Sum	0	23.46	−143%	23.19	−867%	38.09%	11%	61.75%	−34%	32.78%	9%
Average	0	21.32	7%	19.57	−113%	37.46%	−3%	63.22%	37%	33.00%	26%
Analytic prior (Λ=**1**)	0	20.77	46%	19.05	−4%	38.97%	31%	63.28%	39%	33.40%	56%
Linear (no decay)	24,576	20.65	54%	18.95	17%	37.37%	−5%	63.17%	34%	32.96%	23%
MLP-CW (matched)	25,088	**20.33**	**75%**	**18.75**	**58%**	37.79%	5%	62.95%	24%	32.99%	25%
Ours	24,576	20.44	69%	18.78	52%	**39.27%**	**37%**	**63.49%**	**50%**	**33.55%**	**67%**
Full context (reference)	—	19.99	—	18.55	—	42.09%	—	64.53%	—	33.98%	—

4.4 Communication Cost and Latency

We compare the computational cost of state communication with natural-language summary-based communication. Table 3 reports end-to-end latency averaged over 200 HotpotQA examples on one H100 using the instruction-tuned Mamba-2.8B model. State communication takes 180 ms, compared with 278 ms for NL-short and 292 ms for NL-medium. This gives approximately a $1.6\times$ speedup while achieving comparable exact-match scores. These measurements exclude inter-device transmission.

State communication avoids summary generation and receiver-side re-encoding. The receiver accumulates its decay during its forward pass, then fuses the states using two elementwise multiplications and one addition per state entry (Section 3.2). The communication-related overhead is approximately 0.02 GFLOPs, compared with 439 GFLOPs for NL-short and 1507 GFLOPs for NL-medium. These savings apply to communication-related computation. Receiver processing and answer generation remain necessary, so the reduction in communication overhead does not translate proportionally into an end-to-end speedup.

Communication payload. Mamba-2.8B transfers 320 KB of FP32 SSM state per layer, totaling 20 MB across 64 layers, independent of the source context length. A short natural-language summary has a much smaller payload but requires generation and receiver-side re-encoding. State communication exchanges additional bytes to avoid these computations.

4.5 Fusion Design Ablations

We compare five alternative state fusion methods on Mamba-130M to examine the contribution of the accumulated decay and learned correction (Table 4). All methods use the same source and receiver states. Sum and average use the state-pooling operations studied by Jafari et al. (2025), applied here without base-model finetuning. Both methods, as well as the analytic prior with $\Lambda = \mathbf{1}$, require no learned parameters. The learned linear baseline, $h'_R = \Lambda \odot h_S + h_R$, matches our parameter count but omits the accumulated decay. We also evaluate MLP-CW, a parameter-matched channel-wise MLP.

Summation substantially worsens language-modeling performance, while averaging gives inconsistent results. Without learned parameters, the analytic prior closes 46% of the WikiText perplexity gap and improves ARC-Easy by approximately 1.4 percentage points. The learned correction increases WikiText gap closure to 69%, compared with 54% for the learned linear baseline without accumulated decay.

MLP-CW achieves lower WikiText perplexity than our method but lower accuracy on ARC-Easy, PIQA, and HellaSwag. A channel-mixing MLP with approximately $96\times$ more parameters further reduces WikiText perplexity to 20.11, but also trails our method on these downstream tasks

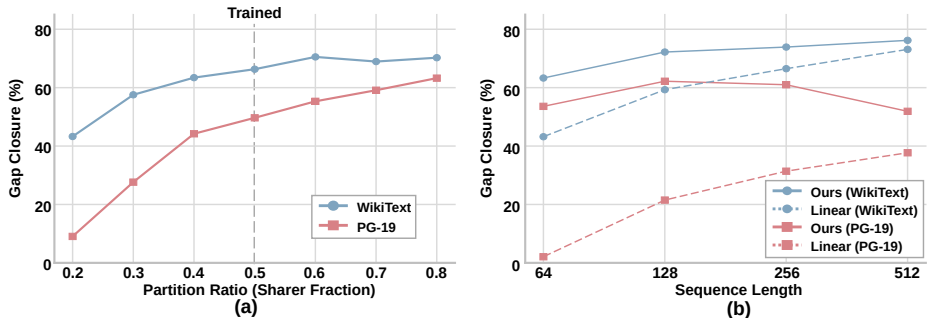


Figure 3: Generalization on Mamba-370M trained at partition ratio 0.5 and length 1024. Perplexity gap closure on WikiText-103 and PG-19 across (a) partition ratios and (b) sequence lengths, comparing accumulated-decay fusion with the no-decay baseline.

(Appendix D). The accumulated-decay design gives the strongest downstream results among the compared fusion methods.

4.6 Generalization and Limits

We evaluate generalization across information partition ratios and sequence lengths. On Mamba-370M trained at partition ratio 0.5, state communication improves WikiText perplexity over receiver-only across ratios 0.2–0.8 (Figure 3a). PG-19 gains are weaker and can be negative when the source holds little context, but gap closure reaches approximately 63% at the largest source fraction. The correction also transfers from 1024-token training sequences to lengths 64–512, with accumulated decay improving robustness relative to the no-decay baseline (Figure 3b).

At 8192 tokens, both full-context processing and state transfer degrade, consistent with the base models' limited extrapolation beyond their 2048-token pretraining length. State transfer reduces the degradation relative to full-context processing but remains worse than receiver-only (Appendix D.4).

4.7 Analysis of Transferred Information

We use linear probes to detect source words in the receiver's state, retaining only examples where the target word is absent from the receiver's context. Detection improves after communication on both Mamba-130M and Mamba-370M. At each target's best layer, all 100 targets improve, with 95% bootstrap intervals excluding zero for 88 and 94 targets, respectively (Appendix G). Because the target word is absent from the receiver's context, the improvement indicates that source information becomes accessible in the receiver's state after communication. This provides evidence of information transfer beyond the aggregate task scores, although probe accuracy alone does not establish whether the receiver uses that information during generation. On Mamba-130M, 85.5% of the learned correction entries lie within $[0.95, 1.05]$, remaining close to the analytic prior.

Disabling fusion at one layer at a time shows that layers 12, 15–17, and 19 contribute most to the perplexity improvement, while early layers contribute little (Appendix E.1, Figure 4).

5 Conclusion and Limitations

We show that a single-layer SSM decomposition can be adapted into a fixed-size latent communication method between instances of the same pretrained SSM. A lightweight per-channel correction enables state exchange across five scales each of Mamba and Mamba-2. The learned correction generalizes across partition ratios, sequence lengths, and out-of-domain text without retraining. On multi-document QA, state communication provides a competitive alternative to natural-language summaries while transmitting no generated message tokens and reducing latency. Key limitations are that the analytic prior is exact only for a single SSM recurrence, most controlled experiments use artificial sequence splits, and downstream task gains remain modest relative to output-distribution recovery. We discuss detailed limitations and future directions in Appendix A.

REFERENCES

Quentin Anthony, Yury Tokpanov, Paolo Glorioso, and Beren Millidge. Blackmamba: Mixture of experts for state-space models. *arXiv preprint arXiv:2402.01771*, 2024.

Aaron Blakeman, Aaron Grattafiori, Aarti Basant, Abhibha Gupta, Abhinav Khattar, Adi Renduchintala, Aditya Vavre, Akanksha Shukla, Akhiad Bercovich, Aleksander Ficek, et al. Nvidia nemotron 3: Efficient and open intelligence. *arXiv preprint arXiv:2512.20856*, 2025.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In *Forty-first International Conference on Machine Learning*, 2024.

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.

In Gim, Guojun Chen, Seung-seob Lee, Nikhil Sarda, Anurag Khandelwal, and Lin Zhong. Prompt cache: Modular attention reuse for low-latency inference. *Proceedings of Machine Learning and Systems*, 6:325–338, 2024.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling*, 2024.

Albert Gu, Karan Goel, Ankit Gupta, and Christopher Ré. On the parameterization and initialization of diagonal state space models. *Advances in neural information processing systems*, 35:35971–35983, 2022.

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.

Yasaman Jafari, Zixian Wang, Leon Bergen, and Taylor Berg-Kirkpatrick. The surprising soupability of documents in state space models. *arXiv preprint arXiv:2505.24033*, 2025.

Aakash Sunil Lahoti, Kevin Li, Berlin Chen, Caitlin Wang, Aviv Bick, Zico Kolter, Tri Dao, and Albert Gu. Mamba-3: Improved sequence modeling using state space principles. In *International Conference on Learning Representations*, volume 2026, pp. 86173–86202, 2026.

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: Communicative agents for" mind" exploration of large language model society. *Advances in neural information processing systems*, 36:51991–52008, 2023.

Tian Yu Liu, Alessandro Achille, Matthew Trager, Aditya Golatkar, Luca Zancato, and Stefano Soatto. Picaso: Permutation-invariant context composition with state space models. In *The Thirteenth International Conference on Learning Representations*, 2025.

Wei Liu, Chenxi Wang, Yifei Wang, Zihao Xie, Rennai Qiu, Yufan Dang, Zhuoyun Du, Weize Chen, Cheng Yang, and Chen Qian. Autonomous agents for collaborative task under information asymmetry. *Advances in Neural Information Processing Systems*, 37:2734–2765, 2024.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

Run Peng, Ziqiao Ma, Amy Pang, Sikai Li, Zhang Xi-Jia, Yingzhuo Yu, Cristian-Paul Bara, and Joyce Chai. Communication and verification in llm agents towards collaboration under information asymmetry. *arXiv preprint arXiv:2510.25595*, 2025.

Maciej Pióro, Maciej Wołczyk, Razvan Pascanu, Johannes von Oswald, and João Sacramento. State soup: In-context skill learning, retrieval and mixing. *arXiv preprint arXiv:2406.08423*, 2024.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. Compressive transformers for long-range sequence modelling. In *International Conference on Learning Representations*, 2020.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Musique: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. Autogen: Enabling next-gen llm applications via multi-agent conversations. In *First conference on language modeling*, 2024.

Jingbo Yang, Bairu Hou, Wei Wei, Yujia Bao, and Shiyu Chang. KVLink: Accelerating large language models via efficient KV cache reuse. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 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 *Proceedings of the 2018 conference on empirical methods in natural language processing*, pp. 2369–2380, 2018.

Jiayi Yao, Hanchen Li, Yuhan Liu, Siddhant Ray, Yihua Cheng, Qizheng Zhang, Kuntai Du, Shan Lu, and Junchen Jiang. Cacheblend: Fast large language model serving for rag with cached knowledge fusion. In *Proceedings of the twentieth European conference on computer systems*, pp. 94–109, 2025.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 4791–4800, 2019.

Jingwei Zuo, Maksim Velikanov, Dhia Eddine Rhaiem, Ilyas Chahed, Younes Belkada, Guillaume Kunsch, and Hakim Hacid. Falcon mamba: The first competitive attention-free 7b language model. *arXiv preprint arXiv:2410.05355*, 2024.

AI USE STATEMENT

In this work, we used generative AI tools for editing and polishing the text and implementing methods. We have not used generative AI tools for, help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, design or provide feedback on research methodology or experiments, support qualitative and thematic data analysis, and interpret results. Generating synthetic datasets, assist with translation, clean and reformat dataset, are not applicable to this work. Additionally, we used generative AI tools for create or edit software code, summarize or analyse existing literature, and edit a research paper to improve readability. We have reviewed all AI-assisted work. LLM-generated code was verified and tested for correctness, and all LLM assisted paper refinements were verified for correctness and accuracy by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

A LIMITATIONS AND FUTURE WORK

The decomposition is exact for an isolated SSM recurrence when independent and joint processing use the same inputs and recurrence coefficients. In pretrained multilayer models, independent context processing changes the intermediate representations and can change these coefficients. The learned correction reduces the resulting mismatch while keeping the base model frozen, but a gap to joint processing remains.

Convolution state not fused. Our method transfers SSM states without fusing the short convolution buffers in Mamba blocks. Independently processing the receiver's context changes the convolution inputs near the partition boundary. Terminal SSM-state fusion does not reconstruct these earlier computations, contributing to the difference from full-context processing.

Evaluation scope. All transfers use instances of the same pretrained checkpoint. The controlled experiments partition a single sequence, while HotpotQA and MuSiQue assign separate supporting documents to the source and receiver. Transfer between different checkpoints and broader communication settings remain untested.

Downstream task transfer. ARC-Easy and HellaSwag improve across the evaluated scales, while PIQA gains are mixed. The MLP comparisons in Appendix D.3 also show that lower perplexity does not necessarily yield higher downstream accuracy. Task-aligned fusion objectives may improve this transfer.

Long contexts and multiple sources. At 8192 tokens, state communication worsens PG-19 perplexity relative to receiver-only at all five Mamba scales (Appendix D.4). Dividing a fixed source paragraph into additional independently processed segments also reduces QA accuracy when using a correction trained for one source (Appendix H.5). These results limit claims of generalization across context lengths and numbers of sources.

Communication cost. The measured QA speedup is relative to natural-language summary pipelines on one GPU. The fixed-size state payload is larger than a short text summary, and inter-device transfer is excluded from the latency measurement. Distributed performance therefore depends on transmission and synchronization costs as well as local computation.

B SINGLE-LAYER STATE DECOMPOSITION

Consider the recurrence $h_k = \bar{A}_k \odot h_{k-1} + \bar{B}_k u_k$ with $h_0 = 0$. We identify diagonal transitions with their entries and use elementwise products, with an empty product equal to the all-ones tensor. The decomposition assumes that independent and joint processing use the same recurrence inputs and coefficients within each segment.

Unrolling the recurrence gives

$$h_T = \sum_{k=1}^{T} \left(\prod_{j=k+1}^{T} \bar{A}_j \right) \odot (\bar{B}_k u_k). \tag{12}$$

Splitting at T_S separates the source and receiver contributions,

$$h_T = \sum_{k=1}^{T_S} \left(\prod_{j=k+1}^{T} \bar{A}_j \right) \odot (\bar{B}_k u_k) + \sum_{k=T_S+1}^{T} \left(\prod_{j=k+1}^{T} \bar{A}_j \right) \odot (\bar{B}_k u_k). \tag{13}$$

The second term is the receiver's independently computed state h_R. For $k \leq T_S$, the source term factors as

$$\prod_{j=k+1}^{T} \bar{A}_j = \underbrace{\prod_{j=T_S+1}^{T} \bar{A}_j}_{\bar{A}_{\text{accum}}} \odot \prod_{j=k+1}^{T_S} \bar{A}_j. \tag{14}$$

The accumulated decay is independent of k, and the remaining source sum is h_S. Therefore,

$$\boxed{h^* = h_T = \bar{A}_{\text{accum}} \odot h_S + h_R.} \tag{15}$$

This identity requires no learned parameters under the stated assumptions.

Accumulated decay. For a fixed transition, $\bar{A}_{\text{accum}} = \bar{A}^{T_R}$ depends only on receiver length. For selective transitions, it also depends on the input through Δ_k. Extending a fixed trajectory with decay entries in $(0, 1)$ further attenuates the source contribution. A running product over the receiver's transitions requires $O(T_R DN)$ operations and $O(DN)$ storage per layer. Terminal fusion requires $O(DN)$ operations.

Multilayer application. Independent context processing changes convolutional boundary features and intermediate representations, so recurrence inputs and coefficients can differ from a joint pass. The learned correction improves continuation predictions after terminal states are combined at every layer, without reconstructing the receiver's earlier computations. Mamba-2 applies the same rule by broadcasting each head's scalar decay over its state entries.

C Training Details

C.1 Hyperparameters

Table 5 summarizes the WikiText-103 fusion-training configuration. The same training recipe is used across the evaluated Mamba and Mamba-2 checkpoints, with separate learned correction parameters for each checkpoint. Task-specific QA training is described in Appendix H.

C.2 Fuser Parameters per Model Scale

Table 6 reports the fuser architecture details for the five Mamba checkpoints. The fuser consists of a single learnable matrix $\Lambda^{(\ell)} \in \mathbb{R}^{D \times N}$ per fused layer.

C.3 Training Procedure

All layerwise correction parameters are trained jointly. For each 1024-token training sequence, the source processes tokens 1–384 and the receiver independently processes tokens 385–768 from zero initial SSM states. We combine their terminal states at every layer using Eq. 9. The receiver then continues over tokens 769–1024. The teacher processes the full sequence, and the KL objective compares teacher and fused output distributions on the continuation positions using the same preceding continuation tokens. All base model weights remain frozen.

We initialize all Λ entries to one, corresponding to the analytic prior. Training the Mamba-130M correction takes under four hours on one NVIDIA A100. Frozen-model forward passes generate the training states and teacher outputs on the fly.

Table 5: Training hyperparameters. All settings are shared across model scales unless noted.

Hyperparameter	Value
Dataset	WikiText-103 (train split)
Number of training samples	50,000
Sequence length	1,024 tokens
Partition ratio	0.5 (equal split)
Context ratio	0.75 (loss on last 25%)
Loss	KL divergence
Max continuation length	256 tokens
Optimizer	AdamW
Learning rate	10^{-3}
LR schedule	Linear decay with warmup
Warmup ratio	0.1 (first 10% of total steps)
Weight decay	0.01
Max gradient norm	1.0
Batch size	128
Number of epochs	10

Table 6: Fuser parameters for the five Mamba checkpoints. Each remains below 1% of the corresponding base model size.

	130M	370M	790M	1.4B	2.8B
D (d_inner)	1,536	2,048	3,072	4,096	5,120
N (d_state)	16	16	16	16	16
Total layers	24	48	48	48	64
Fused layers	24	48	48	48	64
Params per layer	24,576	32,768	49,152	65,536	81,920
Total fuser params	589,824	1,572,864	2,359,296	3,145,728	5,242,880
% of base model	0.45%	0.42%	0.30%	0.22%	0.19%

D Extended Results

D.1 Complete Mamba Results

Table 7 reports the complete scores underlying the receiver-relative comparisons in Table 1. Full context processes the source and receiver contexts jointly. These scores quantify the remaining difference from joint processing and provide the reference for gap closure. A negative gap closure indicates degradation below receiver-only. When full context does not outperform receiver-only, we report raw metrics because normalized gap closure no longer measures recovery toward a stronger reference.

PIQA improves at three Mamba scales, is unchanged at 790M, and decreases at 1.4B. WinoGrande is excluded because full-context performance is near chance across the evaluated scales.

D.2 Complete Mamba-2 Results

We report the full-context, receiver-only, and state-communication scores for all five Mamba-2 scales. The language-modeling comparison additionally includes the analytic prior with $\Lambda = \mathbf{1}$. The learned correction reduces perplexity relative to both receiver-only and the analytic prior on WikiText-103 and PG-19 at every scale. Gap closure is calculated from the displayed scores using Equation 11.

Table 7: Complete scaling results for Mamba, including the full-context joint-processing reference. Perplexity (↓), length-normalized accuracy on ARC-Easy and HellaSwag, and accuracy on PIQA (↑) are shown.

		130M	370M	790M	1.4B	2.8B
WikiText	Full context	19.99	13.79	11.40	10.32	9.07
	Receiver	21.42	14.70	12.16	11.00	9.62
	Ours	**20.44**	**14.03**	**11.63**	**10.51**	**9.21**
PG-19	Full context	18.55	14.04	12.11	11.19	10.07
	Receiver	19.03	14.42	12.44	11.50	10.36
	Ours	**18.78**	**14.23**	**12.28**	**11.42**	**10.27**
ARC-Easy	Full context	42.09%	47.60%	53.45%	59.76%	63.26%
	Receiver	37.58%	40.11%	43.48%	46.93%	50.00%
	Ours	**39.27%**	**43.22%**	**45.29%**	**49.24%**	**52.86%**
HellaSwag	Full context	33.98%	45.77%	54.71%	58.90%	65.82%
	Receiver	32.66%	42.75%	50.58%	54.14%	60.35%
	Ours	**33.55%**	**43.96%**	**51.98%**	**55.41%**	**61.92%**
PIQA	Full context	64.53%	69.48%	71.98%	74.16%	75.24%
	Receiver only	62.46%	66.76%	68.82%	71.27%	72.20%
	Ours	63.49%	66.97%	68.82%	71.00%	72.31%

Table 8: Mamba-2 perplexity on 500 WikiText-103 sequences and 200 PG-19 sequences. Lower values are better. The analytic prior uses no learned correction parameters. Full context provides the joint-processing reference.

Dataset	Condition	130M	370M	780M	1.3B	2.7B
WikiText-103	Receiver only	20.961	14.620	12.133	10.690	9.266
	Analytic prior ($\Lambda = \mathbf{1}$)	20.001	13.986	11.569	10.230	8.899
	State communication	19.798	13.823	11.473	10.144	8.814
	Full context (reference)	19.445	13.620	11.344	10.014	8.696
	Gap closed (%)	76.7	79.7	83.7	80.8	79.3
	Prior gap closed (%)	63.3	63.4	71.5	68.0	64.4
PG-19	Receiver only	23.823	18.132	15.788	14.329	12.930
	Analytic prior ($\Lambda = \mathbf{1}$)	23.701	17.948	15.575	14.144	12.756
	State communication	23.661	17.871	15.535	14.094	12.715
	Full context (reference)	23.290	17.707	15.385	13.930	12.595
	Gap closed (%)	30.4	61.4	62.8	58.9	64.2
	Prior gap closed (%)	22.9	43.3	52.9	46.4	51.9

D.3 Additional MLP Fuser Baselines

Architecture details. Both MLP fusers operate on the SSM hidden state shape $\mathbb{R}^{D\times N}$, where D is the channel dimension (e.g., $D = 1536$ for Mamba-130M) and $N = 16$ is the state dimension per channel. They differ in whether channels interact.

MLP-CW (channel-wise). A two-layer MLP with hidden size $H = 512$ is applied independently to each of the D channels with weights shared across channels:

$$[h_{S,d}; h_{R,d}] \in \mathbb{R}^{2N} \;\to\; \mathrm{Linear}(2N, H) \;\to\; \mathrm{SiLU} \;\to\; \mathrm{Linear}(H, N) \;\to\; \delta_d,$$

with the fused state computed as $h'_R = h_R + \delta$. The output projection is zero-initialized so the fuser starts as identity. We set $H = 512$ to match our fuser's per-layer parameter count. Channels do not interact; D is treated as a batch dimension.

MLP-XC (channel-mixing). A small per-channel preprocessor (4-layer MLP with hidden size $H = 32$ and intermediate LayerNorm+SiLU blocks) produces a per-channel delta $\delta \in \mathbb{R}^{D\times N}$, followed by a cross-channel mixer $\mathrm{Linear}(D, D)$ applied along the channel dimension. The cross-channel mixer dominates the parameter count: at Mamba-130M's $D = 1536$, it accounts for $\sim$2.36M of the $\sim$2.36M parameters per layer ($> 99\%$), yielding a total fuser size of 56.7M. MLP-XC's larger size relative to MLP-CW thus reflects the cost of explicit cross-channel mixing rather than added per-channel depth.

Table 9: Mamba-2 zero-shot length-normalized accuracy percentages. Evaluation uses 2,376 ARC-Easy, 10,042 HellaSwag, and 1,838 PIQA examples. Higher values are better. Full context provides the joint-processing reference.

Dataset	Condition	130M	370M	780M	1.3B	2.7B
ARC-Easy	Receiver only	35.48%	39.44%	43.86%	47.43%	50.34%
	State communication	39.35%	43.77%	47.18%	52.19%	54.17%
	Full context (reference)	41.50%	47.22%	53.20%	58.00%	62.92%
	Gap closed (%)	64.3	55.7	35.5	45.0	30.4
HellaSwag	Receiver only	32.77%	42.65%	50.22%	55.09%	60.58%
	State communication	33.41%	44.56%	52.62%	57.75%	63.35%
	Full context (reference)	34.12%	45.99%	54.90%	60.10%	66.06%
	Gap closed (%)	47.4	57.2	51.3	53.1	50.5
PIQA	Receiver only	63.11%	67.46%	71.00%	72.36%	73.50%
	State communication	63.38%	67.90%	70.57%	72.74%	75.35%
	Full context (reference)	65.07%	69.31%	72.20%	74.86%	76.50%
	Gap closed (%)	13.8	23.8	-35.8	15.2	61.7

Table 10: Additional MLP fuser baselines on Mamba-130M (partition ratio 0.5, 1024 tokens). MLP-CW is a parameter-matched channel-wise MLP that does not mix channels. MLP-XC is a dense channel-mixing MLP with 56.7M parameters, about 96× larger than our 0.59M-parameter fuser.

		WikiText-103		PG-19		ARC-Easy		PIQA		HellaSwag	
Method	Fuser Params	PPL	Gap↑	PPL	Gap↑	Acc	Gap↑	Acc	Gap↑	Acc	Gap↑
Full context	—	19.99	—	18.55	—	42.09%	—	64.53%	—	33.98%	—
Receiver only	—	21.42	0%	19.03	0%	37.58%	0%	62.46%	0%	32.66%	0%
Sum	0	23.46	−143%	23.19	−867%	38.09%	11%	61.75%	−34%	32.78%	9%
Average	0	21.32	7%	19.57	−113%	37.46%	−3%	63.22%	37%	33.00%	26%
Analytic prior (Λ=1)	0	20.77	46%	19.05	−4%	38.97%	31%	63.28%	39%	33.40%	56%
Linear (no decay)	0.59M	20.65	54%	18.95	17%	37.37%	−5%	63.17%	34%	32.96%	23%
MLP-CW (matched)	0.60M	20.33	75%	18.75	58%	37.79%	5%	62.95%	24%	32.99%	25%
MLP-XC (channel-mixing)	56.7M	20.11	92%	18.72	65%	37.88%	7%	63.33%	42%	33.18%	39%
Ours	0.59M	20.44	69%	18.78	52%	39.27%	37%	63.49%	50%	33.55%	67%

Both fusers are trained with the same KL objective, hyperparameters, and data as our method (Appendix C, Table 5).

MLP fuser baselines. Table 10 evaluates two generic learned fusers. MLP-CW is parameter-matched to our fuser (0.60M vs. 0.59M parameters) but operates channel-wise and therefore does not mix the D channels. MLP-XC allows dense channel mixing, but this changes the parameter regime: for Mamba-130M, where $D = 1536$, dense channel mixing scales roughly as D^2 per layer, yielding a 56.7M-parameter fuser, about 96× larger than ours.

The MLP baselines improve teacher/perplexity matching: MLP-CW improves WikiText/PG-19 PPL to 20.33/18.75, and the much larger MLP-XC improves further to 20.11/18.72. However, these distributional improvements do not translate into stronger downstream accuracy. Both MLP fusers trail the recurrence-grounded fuser on ARC-Easy, PIQA, and HellaSwag, despite MLP-XC having dense channel mixing and substantially more parameters. This suggests that the remaining challenge is not simply adding nonlinear capacity or cross-channel mixing, but learning fusion mechanisms whose teacher-matching gains transfer to task-level decisions. These results suggest that recurrence-grounded fusion provides a useful lightweight inductive bias for downstream transfer, while generic MLP fusion is more effective at imitating the full-context teacher distribution.

D.4 Long-Context Evaluation

We evaluate PG-19 continuation perplexity at a total sequence length of 8192 tokens on all five Mamba scales. The context ratio is 0.75 and the source partition ratio is 0.5. The full-context model therefore prefills 6144 tokens, while the source and receiver each process 3072 tokens independently

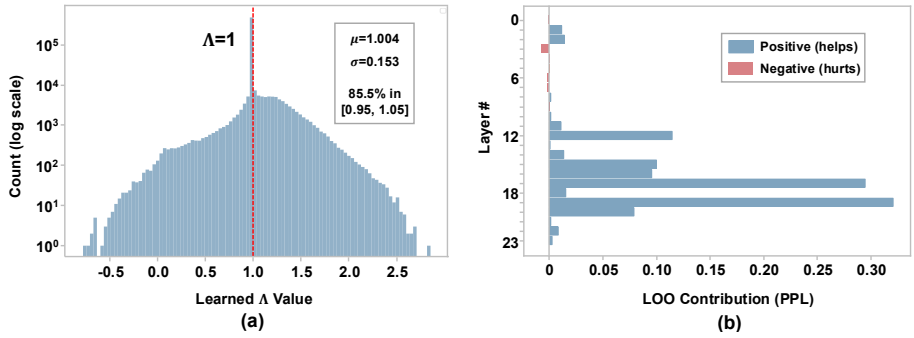


Figure 4: Analysis on Mamba-130M. (a) Distribution of learned Λ values (log scale). Values concentrate around $\Lambda = 1$, with 85.5% in [0.95, 1.05], suggesting the zero-parameter analytic prior is a strong initialization. (b) Per-layer leave-one-out contribution. Fusion benefit is concentrated in mid-to-late layers (12, 15–17, 19), while early layers contribute negligibly.

before fusion. The remaining 2048 tokens form the prediction window. The correction parameters were trained at a total sequence length of 1024 tokens and are applied without retraining.

Table 11 reports raw perplexities. At every scale, receiver-only performs better than both state communication and the full-context reference. At 2.8B, fusion raises perplexity from 16.25 to 28.72, while the full-context value is 118.44. These results show a limit of the evaluated method at long contexts. The experiment does not isolate the contribution of the base model's length extrapolation from the error introduced by state fusion.

Table 11: PG-19 continuation perplexity at 8192 tokens. All checkpoints are from the Mamba family. Lower values are better. The correction is applied without retraining.

Method	130M	370M	790M	1.4B	2.8B
Full context	22.31	21.98	38.20	15.62	118.44
Receiver only	21.44	16.38	14.62	13.08	16.25
State communication	21.79	16.84	15.68	13.43	28.72

D.5 Split Ratio Sweep

Tables 12 and 13 report the full split ratio sweep for Mamba-130M and Mamba-370M on WikiText-103, PG-19, and ARC-Easy. The fuser is trained at ratio 0.5 and evaluated at all ratios without retraining. These results correspond to Figure 3a in the main paper.

E Layer-wise Analysis Details

The learned correction remains near identity. After training, most entries of Λ remain close to one. On Mamba-130M, 85.5% fall within $[0.95, 1.05]$ (Figure 4a), and the same concentration near one is observed across the five Mamba scales. The largest deviations occur in layers where fusion contributes most to performance (Appendix E.1, Table 14).

E.1 Per-Layer Analysis

Table 14 reports three per-layer measurements on Mamba-130M jointly: the learned Λ statistics, the leave-one-out (LOO) and single-layer fusion deltas, and the pre-fusion state divergence between sharer and receiver. For LOO, we disable fusion at that layer only and measure the resulting perplexity; for single-layer, we apply fusion at only that layer and disable it elsewhere. Cosine similarity and

Table 12: Split ratio generalization on Mamba-130M. Fuser trained at ratio 0.5. Perplexity (↓) for WikiText and PG-19, acc_norm (↑) for ARC-Easy.

		0.2	0.3	0.4	0.5*	0.6	0.7	0.8
WikiText	Full context	19.99	19.99	19.99	19.99	19.99	19.99	19.99
	Sharer only	27.74	26.69	25.82	24.81	24.08	23.28	22.56
	Receiver only	20.34	20.61	20.98	21.42	21.94	22.74	23.81
	Fused	20.14	20.24	20.34	20.44	20.56	20.78	20.98
PG-19	Full context	18.55	18.55	18.55	18.55	18.55	18.55	18.55
	Sharer only	23.26	22.93	22.65	22.37	22.07	21.79	21.40
	Receiver only	18.69	18.78	18.88	19.03	19.20	19.44	19.83
	Fused	18.69	18.72	18.75	18.78	18.83	18.88	18.97
ARC-Easy	Full context	42.09%	42.09%	42.09%	42.09%	42.09%	42.09%	42.09%
	Sharer only	29.97%	31.61%	32.49%	34.21%	36.83%	37.92%	39.02%
	Receiver only	40.82%	39.81%	38.68%	37.58%	36.20%	33.38%	30.22%
	Fused	39.48%	39.18%	38.93%	39.27%	38.64%	37.96%	38.96%

Table 13: Split ratio generalization on Mamba-370M. Fuser trained at ratio 0.5. Perplexity (↓) for WikiText and PG-19, acc_norm (↑) for ARC-Easy.

		0.2	0.3	0.4	0.5*	0.6	0.7	0.8
WikiText	Full context	13.79	13.79	13.79	13.79	13.79	13.79	13.79
	Sharer only	19.35	18.62	18.00	17.38	16.87	16.32	15.82
	Receiver only	14.02	14.17	14.40	14.70	15.05	15.57	16.25
	Fused	13.91	13.93	13.98	14.03	14.08	14.22	14.32
PG-19	Full context	14.04	14.04	14.04	14.04	14.04	14.04	14.04
	Sharer only	18.04	17.78	17.56	17.35	17.11	16.86	16.53
	Receiver only	14.15	14.22	14.31	14.42	14.56	14.76	15.07
	Fused	14.14	14.17	14.19	14.23	14.27	14.33	14.41
ARC-Easy	Full context	47.60%	47.60%	47.60%	47.60%	47.60%	47.60%	47.60%
	Sharer only	30.72%	33.38%	35.86%	37.46%	39.27%	42.34%	43.94%
	Receiver only	45.75%	44.28%	42.30%	40.11%	37.92%	35.56%	31.94%
	Fused	45.29%	44.95%	43.73%	43.22%	42.85%	41.50%	40.82%

relative L2 difference are computed between the sharer's and receiver's states immediately before fusion.

Layers 12, 15–17, and 19 make the largest leave-one-out contributions. Removing fusion at layer 19 raises perplexity by 0.320, while early layers contribute little. Layers 16 and 17 also have the largest mean corrections, 1.137 and 1.140. Pre-fusion state divergence alone does not explain these contributions, with some layers showing substantial divergence but little fusion benefit.

F Computational Cost Analysis

F.1 Communication Overhead in Separate-Document QA

Table 3 reports communication-related compute beyond the receiver's forward pass. NL-short and NL-medium incur 439 and 1507 GFLOPs, respectively. State communication incurs approximately 0.02 GFLOPs, including accumulation of the receiver's decay and terminal fusion across layers. The natural-language pipelines generate an intermediate message and require the receiver to encode it. State communication uses the source's terminal SSM states and generates no message tokens.

These operation counts describe communication overhead. The end-to-end latency measurement also includes the other computation in the evaluated QA pipeline, so the ratio of overhead FLOPs does not imply the same ratio of end-to-end runtimes. On one H100, the measured latencies are 278 ms for

Table 14: Per-layer analysis on Mamba-130M (WikiText-103, split 0.5, 1024 tokens). Full fusion PPL = 19.987; base (no fusion) PPL = 20.861. Bolded layers are the top fusion contributors by LOO Δ; they show greater Λ deviation from 1 but no systematic relationship with pre-fusion state divergence.

	Λ statistics		Fusion contribution		State divergence	
Layer	Mean	Std	LOO Δ	1-Layer Δ	Cosine Sim	Rel. Diff
0	0.982	0.063	−0.001	−0.002	0.954	0.299
1	0.985	0.117	+0.011	+0.003	0.459	1.075
2	0.988	0.119	+0.014	−0.004	0.538	0.969
3	0.982	0.090	−0.008	−0.007	0.617	0.871
4	0.980	0.048	+0.000	+0.001	0.598	0.879
5	0.981	0.053	−0.000	+0.001	0.513	0.981
6	0.979	0.056	−0.001	−0.000	0.539	0.958
7	0.983	0.098	−0.001	−0.008	0.429	1.071
8	0.982	0.071	+0.001	+0.002	0.434	1.066
9	0.982	0.068	−0.000	+0.001	0.377	1.123
10	0.980	0.183	+0.001	+0.000	0.390	1.106
11	0.995	0.187	+0.010	+0.022	−0.526	0.979
12	**1.039**	**0.187**	**+0.114**	**+0.032**	**0.594**	**0.896**
13	0.982	0.148	−0.000	−0.012	0.625	0.866
14	1.007	0.173	+0.013	−0.001	0.657	0.823
15	**1.027**	**0.177**	**+0.100**	**+0.028**	**0.350**	**1.149**
16	**1.137**	**0.284**	**+0.095**	**+0.125**	**0.753**	**0.702**
17	**1.140**	**0.233**	**+0.295**	**+0.140**	**0.540**	**0.959**
18	0.985	0.136	+0.015	+0.015	0.640	0.849
19	**1.023**	**0.252**	**+0.320**	**+0.236**	**0.837**	**0.569**
20	**1.017**	**0.250**	**+0.078**	**−0.015**	**0.684**	**0.795**
21	0.981	0.058	+0.001	+0.005	0.756	0.728
22	0.982	0.053	+0.008	+0.011	0.688	0.780
23	0.981	0.028	+0.000	+0.000	0.825	0.627

NL-short, 292 ms for NL-medium, and 180 ms for state communication. Appendix H.3 describes the measurement and implementation.

State payload. For Mamba-2.8B, the transmitted SSM states contain 81,920 FP32 entries per layer across 64 layers, totaling 20 MB. The per-layer payload is 320 KB and is independent of source context length. This is larger than a short natural-language summary. Inter-device transmission is excluded from the reported latency.

F.2 Isolated Fusion Latency

Table 15 reports terminal fusion time on one NVIDIA A100-80GB GPU. Fusion processes a fixed number of state entries per checkpoint, and its measured time changes little across the tested sequence lengths. These measurements are separate from the end-to-end QA latency in Table 3.

Table 15: Terminal state-fusion latency on one A100-80GB. Receiver-side decay accumulation is included in the separate QA overhead accounting.

Sequence length	Mamba-130M (ms)	Mamba-2.8B (ms)
2,048	0.88	2.37
4,096	0.88	2.30
8,192	0.89	2.33

Terminal fusion uses two elementwise multiplications and one addition per state entry. The per-layer latency is approximately 0.037 ms for Mamba-130M and 0.036 ms for Mamba-2.8B, across 24 and 64 layers, respectively.

G Probing Experiment Details

We test whether source words become more detectable in the receiver's state after communication. Each of 20,000 WikiText-103 sequences contains 1024 tokens split equally between source and receiver. We select 100 targets with token rank at least 200 and source-half presence rates in $[0.05, 0.20]$. For each target w_t, the label records its presence in the source, and we retain only examples where it is absent from the receiver,

$$y_{i,t} = \mathbf{1}[w_t \in \text{source}_i], \qquad \mathcal{M}_t = \{i \mid w_t \notin \text{receiver}_i\}. \tag{16}$$

Probe training. We extract the receiver's state before and after fusion at every layer and mean-pool over its $N = 16$ state dimensions,

$$\tilde{h}^{(\ell)} = \frac{1}{N} \sum_{k=1}^{N} h^{(\ell)}_{:,k} \in \mathbb{R}^D. \tag{17}$$

For each target and layer, we train separate linear classifiers on the before- and after-fusion states. Each classifier maps D features to two logits. We use an 80/20 train/test split and AdamW with learning rate 10^{-3} and weight decay 10^{-2} for up to 30 epochs with early stopping. We report the accuracy difference $\Delta_{t,\ell}$ with 95% confidence intervals from 1,000 bootstrap resamples. For each target, we report the layer with the largest improvement.

Results. All 100 targets improve at their selected layers on both Mamba-130M and Mamba-370M. The reported confidence intervals exclude zero for 88 and 94 targets, respectively. Before-fusion accuracy clusters near 0.50, while after-fusion accuracy ranges from 0.54 to 0.74. Best-layer improvements have Pearson correlation $r = 0.717$ across models. Mamba-370M has 82 targets with $\Delta \geq 0.05$ and 25 with $\Delta \geq 0.10$, compared with 65 and 15 on Mamba-130M.

H Multi-Document QA Experimental Details

H.1 Fuser Training

To evaluate state fusion in a natural multi-agent setting and compare against natural language communication, we test on HotpotQA (Yang et al., 2018) and MuSiQue (Trivedi et al., 2022), multi-hop question answering benchmarks where each question is associated with two supporting paragraphs. The sharer receives one supporting paragraph, while the receiver receives the other along with the question (gold-only setting). We use the community instruction-tuned model `clibrain/mamba-2.8b-instruct-openhermes`. Because this model's answer formatting is inconsistent across conditions, we apply a rank-16 LoRA adapter trained on 5,000 examples for 1 epoch to align the output format. The adapter is part of the base model rather than part of the communication mechanism, and is shared identically across all four evaluation conditions (full context, receiver only, NL summaries, and state fusion), ensuring that comparisons isolate the effect of the communication mechanism. Our frozen-weights claim refers to the fuser's training setup, where the LoRA-adapted base model is held fixed and only Λ is updated. The five-scale experiments in Section 4 use unmodified Mamba weights with no LoRA. Because the adapter modifies the base model, the WikiText-trained fuser does not directly apply, so we train a separate fuser on 80,000 randomly sampled training examples for each benchmark using the same hyperparameters as the WikiText fuser (Appendix C, Table 5): $\Lambda^{(\ell)}$ initialized to $\mathbf{1}$, KL divergence loss against the full-context teacher, and AdamW with learning rate 10^{-3}. Fuser training runs for 1 epoch, after which it is evaluated on the full development set (HotpotQA: n=7402, MuSiQue: n=2417). We do not perform early stopping or any hyperparameter tuning against development-set performance. For natural language baselines, the sharer generates a 32- or 128-token summary of its paragraph that is prepended to the receiver's input.

H.2 Prompt Templates and Decoding Settings

We use the checkpoint's Zephyr chat format with a shared system message. The common wrapper is

```
<|system|>
You are reading documents to help answer a multi-hop
question.</s>
<|user|>
[user content]</s>
<|assistant|>
[generated output]</s>
```

For source-state extraction, processing stops at the end of the user turn, before the assistant marker. All generation is greedy, with limits of 32 tokens for NL-short summaries, 128 for NL-medium summaries, and 32 for answers.

Source state. The source user content is `Document 1 (Title A): [A's paragraph text]`. We extract its terminal SSM states without generating a message or providing the question.

Receiver and state communication. The receiver user content is

```
Document 2 (Title B): [B's paragraph text]

Question:  [question]

Answer in one short phrase:
```

Receiver-only processes this prompt from zero initial states. State communication fuses the source states at the end of `Document 2`, then continues through the question and answer prompt. Full context prepends `Document 1 (Title A): [A's paragraph]` on the line before `Document 2` in the same user turn.

Natural-language summaries. The source user content contains `Document 1 (Title A): [A's paragraph]`, a blank line, and `Summarize the above passage in one sentence.` For NL-medium, the instruction uses `2-3 sentences` in place of `one sentence`. The source does not receive the question. The receiver prepends `[Summary from Agent A]` and the generated summary on successive lines, followed by a blank line, to its user content above. The answer prompt is shared across all conditions.

H.3 HotpotQA Latency Measurement and Implementation

The latencies in Table 3 are averaged over 200 HotpotQA development examples on one H100. NL-short takes 278 ms, NL-medium takes 292 ms, and state communication takes 180 ms. The QA accuracy results use the larger development sets specified in the table caption. These timings exclude inter-device transmission and are distinct from the isolated A100 fusion measurements in Appendix F.2.

The community instruction-tuned Mamba-2.8B checkpoint uses the original `mamba-ssm` implementation. We use a custom Triton kernel that accepts a supplied initial SSM state. After independently processing the source and receiver contexts, the fused terminal states initialize the receiver's continuation. The kernel is released with our code.

The reported overhead includes all communication-related compute beyond the receiver's forward pass, including decay accumulation and terminal state fusion. The GFLOP values are operation counts, while the latency values are measured wall-clock times.

H.4 Additional HotpotQA Evaluation

On an additional 2,819-example HotpotQA evaluation, state communication improves EM over receiver-only by 1.99 percentage points, with a 95% confidence interval of $[0.64, 3.33]$ points. We report this evaluation separately from Table 3, since it uses a different evaluation sample.

H.5 TRANSFER FROM MULTIPLE SOURCE SEGMENTS

We evaluate transfer from multiple source segments on HotpotQA using Mamba-2.8B. The source paragraph is divided into k ordered segments, each processed independently, while the receiver's paragraph is unchanged. The total source information is fixed. At each layer, all source states are combined with the receiver's state in a single expression. The correction was trained for a single source and is applied unchanged for $k > 1$.

Table 16 reports exact match and confidence intervals for the change relative to receiver-only and to the single-source condition. The full-context model scores 0.526 EM and receiver-only scores 0.486 in this evaluation. Point estimates decline as the number of source segments increases. At $k = 4$, the difference from the single-source condition is -1.49 percentage points, with a 95% confidence interval of $[-2.51, -0.46]$ points. For $k = 2, 3, 4$, the confidence intervals for improvement over receiver-only include zero. Additional segment boundaries and the mismatch with the single-source training configuration are both possible contributors, and this experiment does not distinguish them.

Table 16: HotpotQA results with a fixed source paragraph divided into k segments. The receiver is additional to the k source segments. Changes and 95% confidence intervals are in percentage points. EM scores are rounded, while changes retain the precision reported by the evaluation.

k	EM	Δ vs. receiver	95% CI	Δ vs. $k = 1$	95% CI
1	0.506	+1.99	$[+0.64, +3.33]$	–	–
2	0.499	+1.31	$[-0.09, +2.72]$	-0.67	$[-1.52, +0.17]$
3	0.498	+1.21	$[-0.21, +2.62]$	-0.78	$[-1.74, +0.18]$
4	0.491	+0.50	$[-0.94, +1.93]$	-1.49	$[-2.51, -0.46]$

H.6 TRANSFORMER LATENT COMMUNICATION ON MULTI-DOCUMENT QA

We evaluate context transfer on Llama-3.2-3B-Instruct using 7,402 HotpotQA development examples in the same gold-only two-document setting as the SSM evaluation. Table 16 compares naive KV concatenation with a CacheBlend variant (Yao et al., 2025) in FP32. Both methods transfer the full source KV cache. Our CacheBlend implementation selects the recomputation set once at layer 1 and does not implement gradual filtering. At a nominal recomputation setting of $r = 0.15$, the reported receiver recomputation fraction is 18.0%.

Both methods improve over receiver-only. Selective recomputation raises EM from 0.444 to 0.476, F1 from 0.607 to 0.662, and KL closure from 75.1% to 90.2%. Recomputing all tokens recovers the full-context scores of 0.508 EM and 0.712 F1. These results provide a reference for cache-based communication with token recomputation, while the SSM method incorporates the source state without replaying its tokens. The transformer and SSM evaluations use different base models and full-context references, so their scores do not isolate the effect of the communication method across architectures.

Table 17: Transformer results on 7,402 HotpotQA development examples using Llama-3.2-3B-Instruct in FP32. The CacheBlend variant selects tokens at layer 1 without gradual filtering. Cache-transfer methods transmit the full source KV cache. Recompute reports the CacheBlend recomputation fraction.

Method	EM	F1	KL closure	Recompute
Receiver only	0.372	0.518	0%	–
Naive KV concatenation	0.444	0.607	75.1%	0%
CacheBlend variant, $r = 0.15$	0.476	0.662	90.2%	18.0%
CacheBlend variant, $r = 1.0$	0.508	0.712	100%	100%
Full context (reference)	0.508	0.712	–	–