M^{2}RNN: Non-Linear RNNs with Matrix-Valued States for Scalable Language Modeling

Mayank Mishra
UC Berkeley, Learning Machine
mayank_mishra@berkeley.edu

Shawn Tan[†]
Google DeepMind

Ion Stoica
UC Berkeley

Joseph E. Gonzalez[*]
UC Berkeley

Tri Dao[*]
Princeton University, Together AI

Abstract

Transformers are limited to computations in the TC^0 complexity class, excluding tasks such as entity tracking and code execution that provably require greater expressivity. Motivated by this, we revisit non-linear Recurrent Neural Networks (RNNs) and introduce Matrix-to-Matrix RNN (M^{2}RNN): an architecture with matrix-valued hidden states and non-linear state transitions. We show that (i) the language modeling performance of non-linear RNNs is limited by their state size, and (ii) outer-product state expansion enables efficient tensor-core utilization. Empirically, M^{2}RNN achieves perfect state-tracking generalization at sequence lengths beyond training. These gains transfer to large-scale language modeling: at 7B MoE, Hybrid M^{2}RNN outperforms Hybrid Gated DeltaNet by 0.4–0.5 perplexity points using $3\times$ smaller recurrent states, and replacing even a single recurrent layer with M^{2}RNN yields accuracy gains comparable to Hybrid M^{2}RNN with minimal throughput cost. Hybrid Gated DeltaNet with a single M^{2}RNN layer also outperforms state-of-the-art hybrid linear-attention architectures by up to 8 points on LongBench. Together, these results establish non-linear RNN layers as a compelling building block for efficient and scalable language models.

1 Introduction

Foundation models trained on massive datasets and adapted to downstream tasks have become the dominant paradigm in modern machine learning [9, 12, 27]. At their core, these models rely on sequence-based architectures capable of processing diverse modalities including text, code, images, video, and speech [9, 12, 21, 27, 48, 60]. The prevailing architecture is the decoder-only Transformer, composed of alternating multi-layer perceptron (MLP) and causal attention layers [88]. Much of attention's success stems from its amenability to parallelization across the sequence length, the availability of hardware-efficient implementations [17, 19, 76, 96], and its effective in-context retrieval capabilities. Attention-based Transformers have also demonstrated remarkable scalability, powering models ranging from hundreds of billions [9, 12, 46, 81, 90, 92] to trillions of parameters [56, 85].

However, attention's quadratic time complexity during training and linearly growing memory requirements during inference have motivated the development of more efficient alternatives [84]. State Space Models (SSMs) and linear attention [18, 29–31, 68, 93–95] have emerged as promising

[*]Equal advising.
[†]Work done at MIT-IBM Watson Lab.
Training code: https://github.com/open-lm-engine/lm-engine
Models: https://huggingface.co/collections/open-lm-engine/m2rnn

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

replacements. Katharopoulos et al. [40] propose computing linear attention via a dot product of kernel features, Peng et al. [69] approximate attention using random feature methods, Schlag et al. [75] introduce the delta update rule, showing significant improvements in associative recall, and Dao and Gu [18], Gu and Dao [29] propose Mamba with an efficient training algorithm that scales linearly with sequence length.

Linear attention admits both a recurrent and a parallel formulation. The constant-state size for linear attention enables memory-efficient and fast autoregressive inference (using the recurrent form) while the standard parallel form retains quadratic complexity during training. To bridge this gap, a chunkwise parallel formulation can be employed [37, 93], preserving linear time complexity while batching computations within chunks to leverage matrix multiplication units (tensor cores on NVIDIA GPUs [64]). SSMs like Mamba-1 [29] can alternatively be accelerated via the parallel scan algorithm [8, 34].

Although linear RNNs have attracted considerable attention due to their efficient inference and training, they present several notable limitations:

1. **Limited state tracking**: Linear RNNs with input-independent or diagonal transition matrices, such as Mamba-2 and Gated DeltaNet, are provably less expressive than non-linear RNNs, particularly for hard state-tracking tasks such as code evaluation, entity tracking, and permutation composition (S_5 group permutation task); they are no more expressive than Transformers [54]. Linear RNNs with input-dependent, non-diagonal transitions can be more expressive [28, 80, 86], but training them stably requires constraining the spectral norm of the transition at every step.
2. **Poor in-context retrieval performance**: Linear RNNs exhibit weak performance in in-context retrieval tasks [93–95]. Because their recurrent state is updated via a fixed-rank outer product, the state can be overwritten when the number of key-value associations exceeds its capacity, leading to degraded in-context retrieval performance. This manifests as poor performance on needle-in-a-haystack benchmarks and long-context recall tasks that require retrieving specific information from distant context.

A common strategy is to interleave quadratic attention layers with sub-quadratic recurrent layers in order to achieve a balance between inference efficiency and downstream task performance in state-of-the-art production models. This alleviates the poor language modeling and long-context performance of linear RNNs while retaining their efficiency advantages for long-context training and inference [7, 44, 72].

While non-linear RNNs have been shown to excel at hard state tracking problems [54], they face several practical challenges:

1. **Poor language modeling performance**: Current non-linear RNNs significantly underperform linear RNNs in language modeling. We show that large state sizes are critical for strong language modeling performance, and that the inferior results of non-linear RNNs are primarily due to their smaller state sizes rather than their non-linearity.
2. **Poor long-context retrieval performance**: We further show that the smaller state sizes significantly hurt the retrieval and long-context performance. The vector-valued non-linear RNNs in our experiments consistently underperform Mamba-2 and Gated DeltaNet by ≈ 20 points in real-world retrieval tasks (Tables 8, 2).
3. **Training inefficiency**: Unlike linear RNNs, non-linear RNNs cannot be parallelized across the sequence length, making training significantly more expensive. This is compounded by poor hardware utilization: naïve recurrent GEMM execution prevents reuse of on-chip tiles across time steps, forcing costly HBM I/O and synchronization. Block-wise variants such as FlashRNN [70] keep the recurrence on-chip but map it onto tensor cores by batching sequences, which requires padding when the per-GPU micro-batch is small.

We propose Matrix-to-Matrix RNN (M^2RNN), a non-linear RNN architecture that, when combined with hybrid linear RNNs, results in significant performance improvements in both language modeling and long-context performance. Specifically, M^2RNN layers address the following key limitations of linear RNNs and non-linear RNNs:

1. **Improved state tracking**: Matrix-to-Matrix RNN (M^2RNN) layer achieves perfect state tracking on S_3, significantly improving state tracking performance over diagonal linear RNNs similar to

non-linear RNNs and GRUs [10]. We further demonstrate that M^{2}RNN layers can express all computations that can be performed by a non-linear RNN.

2. **Improved language modeling, in-context retrieval and long-context performance**: We demonstrate the importance of large state sizes for recurrent models and that it is a key driver behind the success of linear RNNs such as Mamba-2 [18] and Gated DeltaNet [94]. We show that the outer product state expansion mechanism is directly applicable to non-linear RNNs. Similar to LSTMs [35] and GRUs [10], we use a forget gate to prevent gradient degradation across time steps [13], but unlike LSTMs and GRUs, our forget gate is independent of the recurrent state, enabling parallel computation. Hybrid M^{2}RNN models outperform Hybrid Mamba-2 and Hybrid Gated DeltaNet on language modeling benchmarks. The outer product state expansion mechanism also improves in-context retrieval and long-context performance which are sensitive to the state size. Further, we demonstrate that combining M^{2}RNN layers with Hybrid Mamba-2 and Hybrid Gated DeltaNet enhances long-context performance by up to 8 points at both 410M dense and 7B MoE scales (Tables 9, 3).
3. **Improved hardware utilization**: The outer product based state expansion mechanism maps each recurrent step to a dense tensor-core GEMM whose shape is independent of the batch size, avoiding the batch padding that FlashRNN [70] needs at small micro-batch sizes, at the cost of more FLOPs per step than vector-valued RNNs (Section 3.4). Since M^{2}RNN layers are computationally expensive and non-parallelizable across time steps, we further explore using M^{2}RNN layers sparingly. We demonstrate with experiments that using M^{2}RNN layers sparingly (even just one layer) results in substantial improvements in all benchmarks.

Beyond efficient single-device execution, training models at scale requires careful attention to systems design. We describe the forward and backward kernels for M^{2}RNN layers implemented in Triton [87], and present two strategies for applying tensor parallelism (TP) to M^{2}RNN layers: a topology-aware approach that requires no additional communication, and a topology-independent approach that preserves parameter count across TP configurations at the cost of extra synchronization (Section G.3). We discuss related work, including xLSTM [3], FlashRNN [70], Griffin [20] and Titans [4] in Appendix A.

2 Background

We use bold letters ($\mathbf{A}$) to represent matrices and small letters (a) to represent column vectors in the following sections.

2.1 State Tracking

State tracking refers to determining the state of a system by observing a sequence of updates. Its complexity is characterized by circuit complexity classes. TC^0, the class of languages decidable by constant-depth threshold circuits of polynomial size, contains simple problems such as parity (the permutation group S_2); Merrill et al. [53] show that Transformers [88] without chain-of-thought lie in TC^0, making them capable of solving only these simpler tasks (Appendix A). NC^1, decidable by logarithmic-depth circuits of polynomial size[2], contains many practical tasks such as tracking chess moves or evaluating programs, which reduce to problems at least as hard as the S_5 word problem.[3] Since S_5 is NC^1-complete, these tasks lie strictly outside both Transformers [53] and linear SSMs with input-independent or diagonal transitions [54], motivating architectures with greater expressivity.

2.2 Linear RNNs

Linear attention and SSMs admit both recurrent and parallel formulations, and are collectively referred to as *linear* RNNs. Merrill et al. [54] demonstrate a significant expressivity gap between these and *non-linear* RNNs [23], which apply a non-linear activation at each recurrence step and have traditionally been employed for sequence modeling, particularly for tasks requiring state tracking. The non-linearity yields a richer function class but precludes parallel scan [8, 34] for efficient computation [54]. Nevertheless, the state-tracking capabilities enabled by non-linear RNNs can make

[2] Depth is measured in the input size.

[3] S_k denotes the symmetric group on k elements.

their incurred cost a reasonable tradeoff. Many linear attention and SSM models take the general form

$$\mathbf{H}_t = \mathbf{H}_{t-1}\mathbf{A}(x_t) + \mathbf{B}(x_t), \quad y_t = \mathbf{H}_t^\top c(x_t) + d(x_t) \tag{1}$$

where $\mathbf{H}_t \in \mathbb{R}^{K\times V}$ is the matrix-valued recurrent state, $\mathbf{A}(x_t) \in \mathbb{R}^{V\times V}$ is the transition matrix, $\mathbf{B}(x_t) \in \mathbb{R}^{K\times V}$, $c(x_t) \in \mathbb{R}^{K\times 1}$, and $d(x_t) \in \mathbb{R}^{V\times 1}$ are functions of the input $x_t \in \mathbb{R}^d$, and K, V are the key and value head dimensions; $d(x_t)$ is the residual stream. Katharopoulos et al. [40] replace the similarity score in attention ($\exp(q_i^\top k_j)$) [88] with a kernel $\kappa(q_i, k_j) = \langle \phi(q_i), \phi(k_j)\rangle$, where ϕ is the feature map for κ, yielding the recurrence

$$\mathbf{H}_t = \mathbf{H}_{t-1} + \phi(k_t){v_t}^\top, \quad z_t = z_{t-1} + \phi(k_t), \quad y_t = \frac{\mathbf{H}_t^\top \phi(q_t)}{z_t^\top \phi(q_t)} \tag{2}$$

Prior works found that removing the normalization term $z_t^\top \phi(q_t)$ in Equation 2 improves training stability [83], giving $\mathbf{H}_t = \mathbf{H}_{t-1} + \phi(k_t){v_t}^\top$, $y_t = \mathbf{H}_t^\top \phi(q_t)$. Yang et al. [95] (Table 2) provides a comprehensive summary of recurrent forms for various linear attention models.

2.3 Limitations of Linear RNNs

Limited State Tracking. Merrill et al. [54] show that hard state tracking tasks like S_5 cannot be solved when the transition matrix is input-independent ($\mathbf{A}(x_t) = \mathbf{A}$) or diagonal ($\mathbf{A}(x_t) = \text{diag}(a(x_t))$), but input-dependent, non-diagonal transitions can simulate deterministic finite-state automata (DFAs). Grazzi et al. [28] extend this, showing that SSMs whose transitions have only positive eigenvalues (as in Mamba and Gated DeltaNet) cannot even solve parity, but Mamba-1 [29] and Gated DeltaNet [94] can solve S_3 and S_5 once eigenvalues are allowed in $[-1, 1]$. Siems et al. [80] further increase expressivity via products of $k-1$ Householder matrices, enabling tasks like S_k. Terzić et al. [86] instead use structured sparse matrices with straight-through gradient estimators, retaining parallel-scan compatibility [8, 34] while improving expressivity.

Poor in-context retrieval performance. Linear RNNs and SSMs significantly underperform parameter-matched Transformers [88] on in-context retrieval [93–95]. Recent models such as DeltaNet [95] and Gated DeltaNet [94] narrow but do not close this gap, as we further demonstrate in Section 4.4 (Tables 8, 2).

2.4 Non-Linear RNNs

The basic non-linear RNN [23] takes the form $h_t = \tanh(\mathbf{W}h_{t-1} + x_t)$, $y_t = h_t$, where $h_t, x_t \in \mathbb{R}^{d\times 1}$ and $\mathbf{W} \in \mathbb{R}^{d\times d}$ is a dense, input-independent transition matrix. Vanishing and exploding gradients make learning long-range dependencies difficult [5]. LSTMs [35] and GRUs [11] address this via multiplicative gating.

Expressivity. Under finite precision, non-linear RNNs are equivalent to finite-state automata and recognize all regular languages [50]. Under infinite precision, saturated non-linear RNNs simulate arbitrary DFAs and thus solve hard state-tracking tasks such as permutation group composition that lie outside TC^0 [54]. This advantage over linear RNNs (confined to TC^0 under diagonal or input-independent transitions) motivates their use in architectures targeting stateful computation beyond what Transformers can express.

2.5 Limitations of Non-Linear RNNs

2.5.1 Poor language modeling performance

Despite their expressivity advantages, LSTMs [35] and GRUs [10] significantly underperform both Transformers [88] and modern linear RNNs on language modeling. We argue the gap is largely attributable to state size: vector-valued non-linear RNNs maintain $h_t \in \mathbb{R}^d$, far smaller than the matrix-valued $\mathbf{H}_t \in \mathbb{R}^{K\times V}$ of linear attention (Appendix B) at matched parameter count. Naively increasing d is impractical: $\mathbf{W} \in \mathbb{R}^{d\times d}$ grows quadratically and for large d no longer fits in SM shared memory, eliminating on-chip reuse.

2.5.2 Poor in-context retrieval performance

Non-linear RNNs also struggle at in-context retrieval, again due to state size: a vector-valued $h_t \in \mathbb{R}^d$ must compress the entire context, becoming overloaded as sequence length grows. Transformers retain all tokens in an explicit KV cache; linear RNNs with matrix-valued states fare better than non-linear RNNs but still lag Transformers on in-context retrieval and recall-intensive benchmarks [93–95] (Section 4.4). Non-linear RNNs, constrained to vector-valued states, exhibit an even larger retrieval deficit.

2.5.3 Training Inefficiency

Non-Parallelizability on Sequence Length. The element-wise non-linearity breaks the associativity required by parallel scan [8, 34], forcing sequential computation. DEER [45], ELK [26] and ParaRNN [16] parallelize the recurrence by solving a system of L non-linear equations via Newton's method [65], but storing the Jacobian is memory-intensive and computationally expensive. To make it feasible, ParaRNN imposes diagonal structure on the transition matrix, eliminating cross-neuron mixing and reducing expressivity. Moreover, each forward pass requires multiple Newton iterations, and the number of iterations needed for convergence is not known a priori, so the sequential recurrence may be more efficient in practice.

Expensive Computation due to Back-to-back GEMM. Computing $\mathbf{W}h_{t-1}$ on a GPU tiles the output across SMs. This works for a single GEMM but at the next step the output tiles from $t-1$ become input tiles for t and reside on different SMs than where they are needed. Standard tiling prevents direct reuse, so intermediates must be written to and reloaded from HBM, introducing costly synchronization similar to fusing back-to-back GEMMs for MLPs. FlashRNN [70] (also adopted by xLSTM [3]) addresses this with a block-wise recurrence, analogous to multi-head attention, keeping per-block state in SM shared memory to avoid HBM-level synchronization and achieve significant speedups.

Poor Hardware Utilization due to Wasted FLOPs. FlashRNN [70] parallelizes the recurrence over batch (B) and heads (N). To use tensor cores on NVIDIA GPUs, B must be padded to $B_{\text{padded}} = 16$ for the WMMA instruction [64], yielding a GEMM of shape $(B_{\text{padded}}, d, d)$ per step. In practice, per-GPU batch sizes of 16 or more are rarely feasible at large model scales (80GB H100s), so padding wastes substantial FLOPs (75% at $B = 4$). Section 3.4 describes how the M^{2}RNN recurrence avoids this padding.

3 Matrix-to-Matrix RNN

We describe Matrix-to-Matrix RNN (M^{2}RNN), our non-linear RNN layer which addresses the poor language modeling, retrieval and hardware utilization of vector-valued non-linear RNNs while retaining the state tracking ability that linear RNNs lack.

3.1 M^{2}RNN Layer

M^{2}RNN uses matrix-valued hidden states with an outer product state expansion similar to linear attention [40] and SSMs [18, 29], increasing the state size without significantly increasing the parameter count. We add a forget gate that is independent of the recurrent state (and thus parallelizable), and a residual connection around the recurrence to improve gradient flow and mitigate the compounding of $\tanh$ across layers. The M^{2}RNN recurrence is:

$$\mathbf{Z}_t = \tanh(\mathbf{H}_{t-1}\mathbf{W} + k_t v_t^\top) \tag{3}$$

$$\mathbf{H}_t = f_t \mathbf{H}_{t-1} + (1 - f_t)\mathbf{Z}_t \tag{4}$$

$$y_t = \mathbf{H}_t^\top q_t + w_r \odot v_t \tag{5}$$

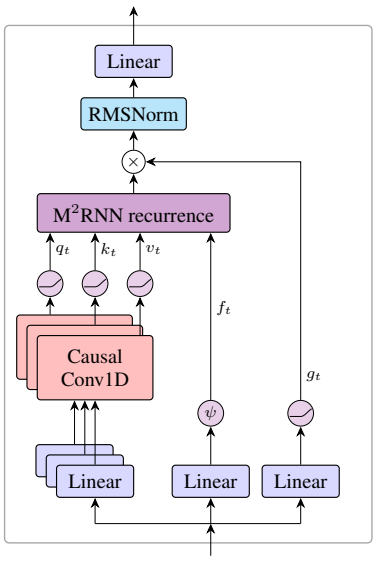

Figure 1: Visualization of the Matrix-to-Matrix RNN layer. This block replaces attention and is combined with MLP and RMSNorm modules as in the Transformer.

where $\mathbf{H}_{t-1} \in \mathbb{R}^{K\times V}$ is the previous state (initialized to $\mathbf{H}_0 = 0_{K\times V}$), $\mathbf{Z}_t \in \mathbb{R}^{K\times V}$ is the state update, and $\mathbf{W} \in \mathbb{R}^{V\times V}$ is the input-independent transition matrix. The query, key, value and forget gate ($q_t, k_t \in \mathbb{R}^{K\times 1}, v_t \in \mathbb{R}^{V\times 1}, f_t \in [0, 1]$) are functions of the input $x_t \in \mathbb{R}^{d\times 1}$. $y_t \in \mathbb{R}^{V\times 1}$ is the output after query readout and residual addition, with residual weight w_r. Following LSTMs [35] and GRUs [10], we use a scalar-valued (per-head) forget gate to mitigate vanishing gradients: $f_t = 1$ preserves the state and $f_t = 0$ resets it. Unlike LSTMs and GRUs, f_t depends only on x_t and not on $\mathbf{H}_{t-1}$. More details are provided in Appendix D.

The queries, keys, and values are computed via linear projections followed by a depthwise causal short-convolution (kernel size 4) and SiLU [22, 33]. The output gate g_t uses SiLU without a convolution, and f_t uses the parameterization from Appendix D. All pre-recurrence projections depend only on x_t and are computed in parallel across the sequence. Following Mamba-1 [29], Mamba-2 [18], and Gated DeltaNet [94], we combine the H3 block [24] with a gated MLP, a design that has proven effective in production LLMs (Figure 1); the full layer equations are given in Appendix F. Initialization for the transition matrix and training stability improvements are in Appendix E.

3.2 Improved State Tracking

Proposition 1. *The M²RNN recurrence can represent all tasks representable by non-linear vector-valued RNNs and hence can represent regular languages.*

The proof is provided in Appendix C.

Siems et al. [80] show that Gated DeltaProduct $[-1, 1]^4$ parameterized as a product of $k - 1$ Householder matrices can in principle represent S_k, so S_3 should be representable with two Householder matrices. We empirically compare M²RNN, GRU [10], Gated DeltaNet $[-1, 1]$ [94], and Gated DeltaProduct $[-1, 1]$ [80] (two Householder matrices) on the S_3 task, training on length 128 and evaluating up to 512.

As shown in Figure 2, Gated DeltaNet $[-1, 1]$ fails to learn the task. Gated DeltaProduct $[-1, 1]$ achieves near-perfect accuracy up to length 256 but fails to generalize beyond the training length, indicating that theoretical expressivity alone does not guarantee robust state tracking. In contrast, both GRU and M²RNN generalize perfectly to unseen lengths, achieving $\geq 99.5\%$ accuracy up to 512. Motivated by this, we investigate whether non-linear RNNs retain this length generalization advantage on language modeling when equipped with an expanded state.

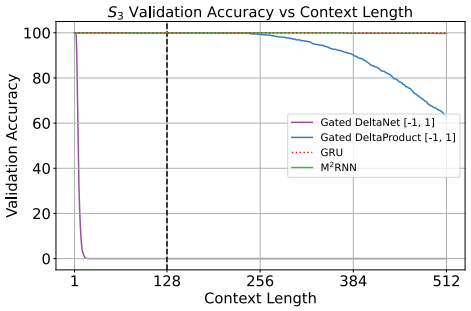

Figure 2: Accuracy on the permutation group S_3 task for M²RNN, Gated DeltaNet $[-1, 1]$, Gated DeltaProduct $[-1, 1]$ and GRU. The vertical line at 128 denotes the training context length for all models. For the Gated DeltaProduct, we use product of 2 Householder matrices since it can solve the S_3 task in theory [28]. However, as the evaluation context length is increased beyond the training context length, we observe accuracy degradation.

3.3 Improved Language Modeling, In-Context Retrieval and Long-Context Performance

Outer-product state expansion is standard in SSMs and linear RNNs [18, 29, 93–95], and Qin et al. [71] show that larger states improve language modeling even with low-rank parameterizations (we corroborate this in Appendix B). Following Mamba-2 [18], we combine the outer product with multi-value attention to further scale the state under a fixed parameter budget (see Table 5 in Appendix B). The matrix-valued state directly addresses the two weaknesses of vector-valued non-linear RNNs identified in Section 2.5.

Language modeling. The poor language modeling of LSTMs [35] and GRUs [10] relative to Transformers [88] and linear RNNs is traditionally attributed to non-linearity. We argue the primary bottleneck is state capacity: expanding $h_t \in \mathbb{R}^d$ to $\mathbf{H}_t \in \mathbb{R}^{K\times V}$ greatly increases the information

[4] $[-1, 1]$ denotes the variant with negative eigenvalues.

stored per step without a proportional parameter increase. Appendix B shows that larger states yield consistent gains under a fixed parameter budget.

In-context retrieval and long-context performance. The outer product write $k_t v_t^\top$ enables efficient storage of key-value associations and structured retrieval via the readout $\mathbf{H}_t^\top q_t$; capacity absent in vector-valued non-linear RNNs. We show the resulting gains in Tables 8, 2 and on long-context tasks in Tables 9, 3.

3.4 Improved Hardware Utilization

Improved Tiling. The outer product expansion lets M^{2}RNN parallelize across the batch dimension B and the number of heads N: each SM runs an independent recurrence over a $K \times V$ state, with each step a single GEMM that contracts along V to produce a $K \times V$ output. The recurrence requires no padding as long as K and V are multiples of 16, enabling tensor core use; e.g., with $K = 64, V = 16$, $\mathbf{H}_{t-1}\mathbf{W}$ uses the WGMMA instruction on Hopper GPUs [64]. Crucially, the GEMM shape is independent of B, unlike vector-valued non-linear RNNs whose GEMM dimensions depend on B and often require batch padding for tensor core compatibility, avoiding the overhead in Section 2.5.3 at the cost of more FLOPs per step than vector-valued RNNs.

Efficient Integration of M^{2}RNN Layers in Hybrid Architectures. Since M^{2}RNN layers are expensive, we use them sparingly. Replacing only a single Mamba-2 or Gated DeltaNet layer with M^{2}RNN in a hybrid architecture matches the accuracy of the full hybrid M^{2}RNN model while keeping training throughput within 6% of the Hybrid Gated DeltaNet at both 4k and 16k context lengths. This provides a scalable strategy for incorporating M^{2}RNN layers into large language models.

Kernels and Distributed Training. We provide custom Triton [87] forward and backward kernels that parallelize across batch and heads, assigning each (b, n) pair to a separate SM; we use the multi-value formulation ($N_q = N_k = 1$, $N_v = N_f = N_w = N$). The forward pass does not store intermediate $\mathbf{H}_t$; the backward pass recomputes the recurrence and writes $\mathbf{H}_t$ for every step to HBM, as Triton cannot keep the recomputed states on-chip (Appendix G). For multi-GPU training, we present two tensor parallelism strategies in Appendix G: a *topology-aware* variant with no extra communication beyond standard TP, and a *topology-independent* variant that preserves parameter count across TP configurations at the cost of additional synchronization.

4 Experiments

We train dense 410M-parameter (24 layers) and Mixture-of-Experts (MoE) [78] 7B-total/1.1B-active (40 layers) models on 100B tokens from the high quality subset of Nemotron-CC-v2 [63].

4.1 Models

We compare M^{2}RNN against Mamba-2 [18], Gated DeltaNet [94], RNN, and GRU [10]. Mamba-2 and Gated DeltaNet are our main baselines since they (and their variants) underpin recent state-of-the-art production models [38, 44, 63, 85]. Following common practice in production LLMs [38, 44], we use 1 attention layer per 7 recurrent layers. We compare Hybrid M^{2}RNN against Transformer++ [88], Hybrid Mamba-2, and Hybrid Gated DeltaNet. We further augment Hybrid Gated DeltaNet with M^{2}RNN layers, yielding "Hybrid GDN + M^{2}RNN-n" with n M^{2}RNN layers. For $n = 1$, we replace the first recurrent layer; for $n > 1$, we replace the layer immediately before each attention layer, producing Hybrid GDN + M^{2}RNN-3 and Hybrid GDN + M^{2}RNN-5 for the 410M and 7B models respectively.

4.2 Model Training

All models use RMSNorm [98] with pre-normalization [91]; Transformer++ uses RoPE [82] while linear and hybrid architectures use NoPE [41]. Model width, intermediate MLP width, layer count, and remaining hyperparameters are held constant across architectures; only the sequence-mixing block parameters vary. MLP and MoE layers use the SwiGLU activation [77]. We train with AdamW

[47] on a cosine schedule, linearly warming up over the first 10B tokens to a peak of 3×10^{-4} then decaying to 10% of the peak (3×10^{-5}), with gradient clipping 1 and weight decay 0.1 (not applied to normalization parameters). We use batch size of 1024 with context length 4096, and all models see identical data in identical order. The 410M dense and 7B MoE models train on 32 and 64 H100s respectively, with each experiment taking between 14–20 hours. Training infrastructure details are provided in Appendix G.4.

4.3 Language Modeling

We evaluate on EleutherAI's LM Eval Harness [25] (LAMBADA [66], HellaSwag [97], PIQA [6], ARC [15], Winogrande [74], BoolQ [14], OpenBookQA [58], CoPA [73], SciQ [89]) and report Wikitext [49] and LAMBADA perplexity. We report the 7B (1.1B active) MoE model here and 410M dense results are in Appendix I.

Homogeneous Architectures. At 7B MoE, Mamba-2 has the best Wikitext perplexity, followed by M^{2}RNN, which beats Gated DeltaNet by 0.09 points (Table 1). M^{2}RNN matches Mamba-2 to within 0.06 points on average downstream accuracy while consistently outperforming Gated DeltaNet. Vanilla RNN and GRU significantly underperform M^{2}RNN and other linear RNNs.

2-way Hybrid Models. Combining M^{2}RNN with attention [88] (Hybrid M^{2}RNN) yields gains on Wikitext [49]: at 7B MoE, Hybrid M^{2}RNN beats Hybrid Mamba-2 [18] by 0.1 and Hybrid Gated DeltaNet [94] by 0.5 perplexity points, and average downstream accuracy also improves over all other hybrids (Table 1). We attribute the gain to M^{2}RNN's non-linear state transition: in the homogeneous setting, M^{2}RNN's smaller state size relative to Mamba-2 and Gated DeltaNet limits it, but in hybrids, attention handles in-context retrieval, freeing M^{2}RNN's non-linear recurrence to add expressivity beyond linear RNNs. Overall, 2-way hybrids improve upon both Transformer++ and homogeneous M^{2}RNN models.

3-way Hybrid Models. 3-way hybrids mix attention, Gated DeltaNet or Mamba-2, and M^{2}RNN layers. Replacing a single linear RNN layer with M^{2}RNN (Hybrid Gated DeltaNet + M^{2}RNN-1, Hybrid Mamba-2 + M^{2}RNN-1) matches full Hybrid M^{2}RNN accuracy at comparable throughput (Section 4.6), suggesting M^{2}RNN layers capture information distinct from Gated DeltaNet.

At 7B MoE, Hybrid Gated DeltaNet + M^{2}RNN-1 cuts Wikitext perplexity by 0.44 over Hybrid Gated DeltaNet, and Hybrid Mamba-2 + M^{2}RNN-1 by 0.1 over Hybrid Mamba-2. Increasing the number of M^{2}RNN layers to 5 yields an additional gain of 0.22 perplexity points. These results highlight the value of non-linear recurrences in sequence mixing.

Model	Wiki PPL (↓)	LMB PPL (↓)	LAMBADA	HellaSwag	PIQA	ARC-E	ARC-C	WinoGrande	BoolQ	OBQA	COPA	SciQ	Avg Acc
Mamba-2	<u>13.73</u>	<u>11.03</u>	50.09	67.43	76.82	74.33	43.52	61.64	64.01	41.00	80.00	93.40	<u>65.22</u>
Gated DeltaNet	13.89	11.10	48.92	66.94	77.26	73.82	40.53	60.14	65.41	39.60	78.00	92.20	64.28
RNN	17.65	43.68	30.25	64.49	76.61	72.64	41.04	55.64	58.35	40.20	79.00	87.60	60.58
GRU	14.80	15.33	43.94	67.36	76.82	75.67	43.52	58.96	63.79	41.00	82.00	91.30	64.44
M^{2}RNN	13.80	11.48	49.18	67.57	76.71	75.51	44.97	58.17	64.16	41.40	82.00	92.00	65.17
Transformer++	14.94	17.26	43.90	62.99	75.63	71.97	38.91	57.54	63.12	37.20	75.00	91.40	61.76
Hybrid Mamba-2	13.10	11.05	50.18	68.62	76.55	72.64	43.09	60.62	63.61	43.00	79.00	93.00	65.03
Hybrid Gated DeltaNet	13.51	11.06	50.20	67.23	77.09	73.06	41.64	59.12	62.39	40.80	82.00	92.90	64.64
Hybrid M^{2}RNN	<u>13.00</u>	<u>10.58</u>	51.74	68.44	76.55	76.81	43.09	61.09	63.21	40.40	81.00	93.30	<u>65.56</u>
Hybrid Mamba-2	13.10	11.05	50.18	68.62	76.55	72.64	43.09	60.62	63.61	43.00	79.00	93.00	65.03
Hybrid Mamba-2 + M^{2}RNN-1	<u>13.01</u>	<u>10.84</u>	50.92	68.35	77.86	74.66	42.24	60.54	62.57	40.40	83.00	92.80	65.33
Hybrid Mamba-2 + M^{2}RNN-5	13.02	11.59	49.84	68.19	77.04	75.17	42.58	59.04	65.26	42.40	83.00	93.90	<u>65.64</u>
Hybrid Gated DeltaNet	13.51	11.06	50.20	67.23	77.09	73.06	41.64	59.12	62.39	40.80	82.00	92.90	64.64
Hybrid Gated DeltaNet + M^{2}RNN-1	13.07	10.33	51.54	68.69	78.18	74.45	41.81	60.30	65.08	42.00	82.00	93.90	**65.80**
Hybrid Gated DeltaNet + M^{2}RNN-5	**12.85**	**10.29**	51.80	68.99	76.82	76.73	43.60	60.69	63.30	40.20	81.00	93.50	65.66

Table 1: Language Modeling on the 7B (1B active) MoE model across commonsense reasoning benchmarks. All models are evaluated in the 0-shot setting. Accuracies (↑) are higher-is-better.

4.4 In-Context Retrieval

In-Context Retrieval on RULER. We evaluate in-context retrieval on the Single Needle In A Haystack (S-NIAH-1/2/3) [62] and Multi-Query (MQ), Multi-Key (MK), and Multi-Value (MV) variants of RULER [36]; see Appendix J for variant descriptions. Figure 3 reports 7B MoE accuracy for Gated DeltaNet-based models; Mamba-2-based results are in Appendix K (Figure 9). Models are trained at 4,096 context length and evaluated up to 16,384.

At 7B MoE, the 3-way Hybrid Gated DeltaNet + M^{2}RNN consistently beats the 2-way hybrids except on S-NIAH-1 (Figure 3), and also improves MQ/MK/MV-NIAH at training-context lengths ($\leq 4{,}096$). On S-NIAH-1, both Hybrid M^{2}RNN and Hybrid Gated DeltaNet hit 100% accuracy at

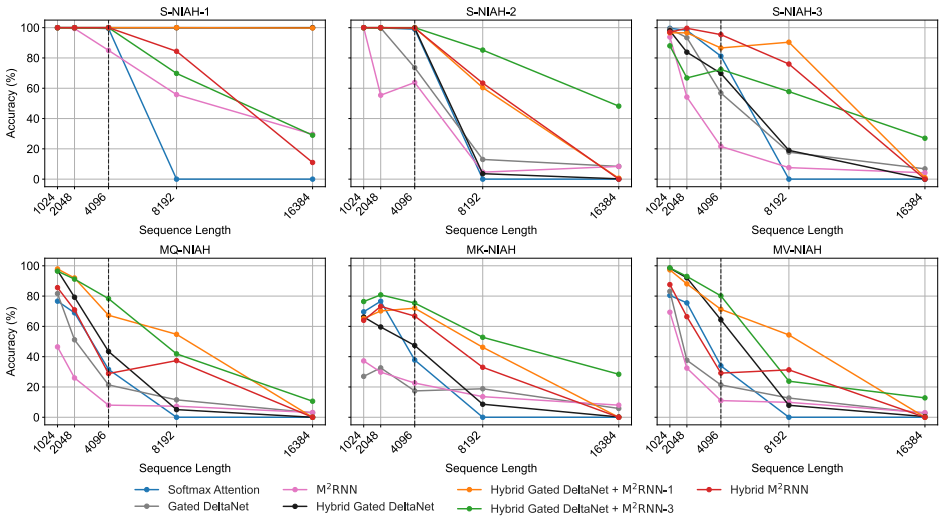

Figure 3: Zero-shot in-context retrieval performance on the RULER benchmark for M^{2}RNN layers and Gated DeltaNet models for the 7B MoE model. The vertical lines at 4096 indicate the training context length.

all lengths; adding M^{2}RNN layers degrades accuracy, likely due to the extremely synthetic passages in S-NIAH-1.

Model	SQuAD	NQ	DROP	TriviaQA	FDA	SWDE	Avg
Mamba-2	41.7	26.5	30.0	65.0	33.0	54.4	41.8
Gated DeltaNet	40.3	24.2	29.9	64.3	32.4	61.8	<u>42.2</u>
RNN	20.0	12.2	20.4	53.1	0.8	9.8	19.4
GRU	27.0	16.7	24.6	59.3	5.2	18.4	25.2
M^{2}RNN	37.9	21.1	28.2	62.5	27.6	49.9	37.9
Transformer++	22.1	24.0	27.6	62.2	67.0	70.1	45.5
Hybrid Mamba-2	48.2	31.4	32.7	67.9	73.1	77.1	55.1
Hybrid Gated DeltaNet	46.4	31.2	28.3	63.6	67.1	71.9	51.4
Hybrid M^{2}RNN	48.4	31.4	31.4	64.6	78.0	79.8	**55.6**
Hybrid Mamba-2	48.2	31.4	32.7	67.9	73.1	77.1	55.1
Hybrid Mamba-2 + M^{2}RNN-1	46.6	30.6	33.6	67.1	78.4	75.0	<u>55.2</u>
Hybrid Mamba-2 + M^{2}RNN-5	46.8	31.5	28.7	66.0	78.5	79.7	<u>55.2</u>
Hybrid Gated DeltaNet	46.4	31.2	28.3	63.6	67.1	71.9	51.4
Hybrid Gated DeltaNet + M^{2}RNN-1	46.4	29.6	32.1	66.4	73.1	77.4	54.2
Hybrid Gated DeltaNet + M^{2}RNN-5	47.1	32.0	31.2	65.5	77.2	77.9	<u>55.1</u>

Table 2: In-context retrieval performance (↑) on real-world data at the 7B (1B active) MoE scale.

In-Context Retrieval on Real-World Data. Yang et al. [95] show DeltaNet underperforms Mamba-2 [18] on real-world retrieval despite strong synthetic-task results. We test whether this holds for M^{2}RNN at 7B MoE scale (Table 2). Transformer++ [88] substantially outperforms recurrent models thanks to its linearly growing KV cache. Among purely recurrent architectures, Gated DeltaNet [94] beats Mamba-2 [18] and M^{2}RNN, which we attribute to Gated DeltaNet's $3\times$ larger state. In hybrids, all models improve over Transformer++, with Hybrid M^{2}RNN gaining the most ($+10.1$ points) and beating Hybrid Mamba-2 and Hybrid Gated DeltaNet. Adding M^{2}RNN layers to Hybrid Gated DeltaNet (or Hybrid Mamba-2) further improves both (Table 2).

4.5 Long-Context Performance

On LongBench [2] at 7B MoE (Table 3), Hybrid M^{2}RNN beats Hybrid Gated DeltaNet by 1.5 points in the 2-way setting. Combining M^{2}RNN with Hybrid Gated DeltaNet adds up to 8 points across long-context summarization, coding, and few-shot tasks; Hybrid Mamba-2 + M^{2}RNN-5 shows similar gains. We also report results on the BabiLong benchmark [42] in Appendix L.

Model	GovReport	QMSum	MultiNews	TREC	TriviaQA	SAMSum	LCC	RepoBench-P	Avg
Mamba-2	5.1	16.0	13.4	42.0	59.7	22.0	13.4	8.7	22.5
Gated DeltaNet	4.8	20.3	12.2	47.0	61.6	36.2	9.7	7.9	<u>25.0</u>
RNN	9.3	16.7	10.5	4.0	27.5	13.0	17.1	15.8	14.2
GRU	5.7	18.0	10.8	9.0	52.4	24.6	20.9	13.9	19.4
M^{2}RNN	8.1	19.1	11.8	26.5	51.3	30.7	20.4	19.3	23.4
Transformer++	6.1	6.1	17.0	16.0	11.4	10.0	13.1	10.2	11.2
Hybrid Mamba-2	9.1	7.7	20.3	62.5	23.3	12.1	5.6	9.8	18.8
Hybrid Gated DeltaNet	10.4	15.8	16.7	57.5	35.1	23.6	9.7	16.3	23.1
Hybrid M^{2}RNN	10.9	13.9	21.1	52.0	44.7	18.5	19.8	15.7	<u>24.6</u>
Hybrid Mamba-2	9.1	7.7	20.3	62.5	23.3	12.1	5.6	9.8	18.8
Hybrid Mamba-2 + M^{2}RNN-1	8.6	3.9	21.2	25.5	15.0	11.5	7.5	11.6	13.1
Hybrid Mamba-2 + M^{2}RNN-5	18.6	10.3	7.1	60.5	45.7	21.4	9.7	8.0	<u>22.7</u>
Hybrid Gated DeltaNet	10.4	15.8	16.7	57.5	35.1	23.6	9.7	16.3	23.1
Hybrid Gated DeltaNet + M^{2}RNN-1	17.2	16.5	6.0	69.5	64.9	34.1	20.1	15.1	30.4
Hybrid Gated DeltaNet + M^{2}RNN-5	12.2	16.0	16.1	59.0	73.5	36.0	18.0	18.9	**31.2**

Table 3: LongBench results (↑) on the 7B (1B active) MoE model.

4.6 Training Throughput

We measure training throughput on $8\times$ NVIDIA HGX H100 GPUs (NVLink) at our 7B (1B active) MoE configuration for M^{2}RNN, Gated DeltaNet [94], Mamba-2 [18], and Transformer++. Mamba-2 achieves the highest throughput at all context lengths, while Transformer++ degrades sharply at long contexts due to its quadratic cost (Figure 10 in Appendix). M^{2}RNN has linear complexity but higher constant factors than the linear baselines in homogeneous settings. Replacing a single Gated DeltaNet layer with M^{2}RNN (Hybrid Gated DeltaNet + M^{2}RNN-1) stays within 6% of Hybrid Gated DeltaNet throughput at 16k context length while consistently improving quality across benchmarks; a favorable tradeoff that makes M^{2}RNN practical when used sparingly. Our current Triton [87] implementation is relatively unoptimized, leaving significant room to further narrow the remaining throughput gap.

5 Limitations and Future Work

M^{2}RNN's sequential non-linear recurrence yields higher constants than linear alternatives, a hand-tuned CUTLASS implementation or scan-style approximations such as ParaRNN [16] could narrow this gap. M^{2}RNN layers are significantly more expensive than linear RNNs limiting their practical usage to only a few layers in the entire model (we demonstrate that this setting works well and even outperforms 2-way hybrid models). Our experiments span 410M dense and 7B (1B active) MoE models trained on 100B tokens at 4k context; validating M^{2}RNN at larger parameter scales requires significantly more compute. We hypothesize that M^{2}RNN layers are also applicable to other modalities like audio, code, math etc but we haven't validated this empirically.

6 Conclusion

We introduce Matrix-to-Matrix RNN (M^{2}RNN), a non-linear RNN with matrix-valued hidden states that addresses key limitations of linear recurrent models via expressive non-linear state transitions, a forget gate, and outer-product state expansion, yielding strong state-tracking. M^{2}RNN is competitive with Mamba-2 [18] and Gated DeltaNet [94] at both 410M dense and 7B (1B active) MoE scales. More importantly, hybrids of M^{2}RNN with attention significantly outperform the equivalent Mamba-2 and Gated DeltaNet hybrids on language modeling, in-context retrieval, and long-context generalization; even a single M^{2}RNN layer adds substantial accuracy at only 6% throughput degradation, making it a practical enhancement for existing architectures.

Matrix-valued recurrence avoids the batch padding that vector-valued recurrences need under FlashRNN [70], enabling tensor-core utilization independent of batch size; combined with our custom kernels, this makes M^{2}RNN viable for training production language models. We also present two complementary tensor parallelism strategies: topology-aware (grouped-value, no extra communication beyond standard TP) and topology-independent (preserves parameter count via a few extra AllReduces), making M^{2}RNN deployable across diverse multi-GPU configurations.

Acknowledgement. We would like to thank Songlin Yang, Bharat Runwal and Kevin Li for providing valuable feedback and discussion throughout the duration of this project. We gratefully acknowledge the support of the Schmidt Sciences AI2050 fellowship, the Google ML and Systems Junior Faculty Awards, and the Google Research Scholar program.

References

[1] Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burovski, Geeta Chauhan, Anjali Chourdia, Will Constable, Alban Desmaison, Zachary DeVito, Elias Ellison, Will Feng, Jiong Gong, Michael Gschwind, Brian Hirsh, Sherlock Huang, Kshiteej Kalambarkar, Laurent Kirsch, Michael Lazos, Mario Lezcano, Yanbo Liang, Jason Liang, Yinghai Lu, C. K. Luk, Bert Maher, Yunjie Pan, Christian Puhrsch, Matthias Reso, Mark Saroufim, Marcos Yukio Siraichi, Helen Suk, Shunting Zhang, Michael Suo, Phil Tillet, Xu Zhao, Eikan Wang, Keren Zhou, Richard Zou, Xiaodong Wang, Ajit Mathews, William Wen, Gregory Chanan, Peng Wu, and Soumith Chintala. Pytorch 2: Faster machine learning through dynamic python bytecode transformation and graph compilation. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, ASPLOS '24, page 929–947, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400703850. doi: 10.1145/3620665.3640366. URL `https://doi.org/10.1145/3620665.3640366`.

[2] Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, et al. Longbench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pages 3119–3137, 2024.

[3] Maximilian Beck, Korbinian Pöppel, Markus Spanring, Andreas Auer, Oleksandra Prudnikova, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. xlstm: Extended long short-term memory. *Advances in Neural Information Processing Systems*, 37:107547–107603, 2024.

[4] Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time, 2024. URL `https://arxiv.org/abs/2501.00663`.

[5] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994. doi: 10.1109/72.279181.

[6] Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language, 2019. URL `https://arxiv.org/abs/1911.11641`.

[7] 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.

[8] Guy E. Blelloch. Prefix sums and their applications. Technical Report CMU-CS-90-190, School of Computer Science, Carnegie Mellon University, November 1990.

[9] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[10] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[11] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[12] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240): 1–113, 2023.

[13] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[14] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions, 2019. URL https://arxiv.org/abs/1905.10044.

[15] 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, 2018. URL https://arxiv.org/abs/1803.05457.

[16] Federico Danieli, Pau Rodriguez, Miguel Sarabia, Xavier Suau, and Luca Zappella. Pararnn: Unlocking parallel training of nonlinear rnns for large language models. *arXiv preprint arXiv:2510.21450*, 2025.

[17] Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023.

[18] Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. *arXiv preprint arXiv:2405.21060*, 2024.

[19] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35:16344–16359, 2022.

[20] Soham De, Samuel L. Smith, Anushan Fernando, Aleksandar Botev, George Cristian-Muraru, Albert Gu, Ruba Haroun, Leonard Berrada, Yutian Chen, Srivatsan Srinivasan, Guillaume Desjardins, Arnaud Doucet, David Budden, Yee Whye Teh, Razvan Pascanu, Nando De Freitas, and Caglar Gulcehre. Griffin: Mixing gated linear recurrences with local attention for efficient language models, 2024. URL https://arxiv.org/abs/2402.19427.

[21] Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[22] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural networks*, 107:3–11, 2018.

[23] Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[24] Daniel Y Fu, Tri Dao, Khaled K Saab, Armin W Thomas, Atri Rudra, and Christopher Ré. Hungry hungry hippos: Towards language modeling with state space models. *arXiv preprint arXiv:2212.14052*, 2022.

[25] Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024. URL https://zenodo.org/records/12608602.

[26] Xavier Gonzalez, Andrew Warrington, Jimmy T. H. Smith, and Scott W. Linderman. Towards scalable and stable parallelization of nonlinear rnns, 2025. URL https://arxiv.org/abs/2407.19115.

[27] Aaron Grattafiori et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

[28] Riccardo Grazzi, Julien Siems, Arber Zela, Jörg KH Franke, Frank Hutter, and Massimiliano Pontil. Unlocking state-tracking in linear rnns through negative eigenvalues. *arXiv preprint arXiv:2411.12537*, 2024.

[29] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First conference on language modeling*, 2024.

[30] Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021.

[31] Albert Gu, Isys Johnson, Karan Goel, Khaled Saab, Tri Dao, Atri Rudra, and Christopher Ré. Combining recurrent, convolutional, and continuous-time models with linear state space layers. *Advances in neural information processing systems*, 34:572–585, 2021.

[32] Wentao Guo, Mayank Mishra, Xinle Cheng, Ion Stoica, and Tri Dao. Sonicmoe: Accelerating moe with io and tile-aware optimizations. *arXiv preprint arXiv:2512.14080*, 2025.

[33] D Hendrycks. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[34] W. Daniel Hillis and Guy L. Steele. Data parallel algorithms. *Commun. ACM*, 29(12):1170–1183, December 1986. ISSN 0001-0782. doi: 10.1145/7902.7903. URL https://doi.org/10.1145/7902.7903.

[35] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9 (8):1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735.

[36] Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. Ruler: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*, 2024.

[37] Weizhe Hua, Zihang Dai, Hanxiao Liu, and Quoc Le. Transformer quality in linear time. In *International conference on machine learning*, pages 9099–9117. PMLR, 2022.

[38] IBM Granite Team. Granite 4.0 language models. https://huggingface.co/collections/ibm-granite/granite-40-language-models, 2025. Hugging Face collection, accessed January 28, 2026.

[39] Dhiraj Kalamkar, Dheevatsa Mudigere, Naveen Mellempudi, Dipankar Das, Kunal Banerjee, Sasikanth Avancha, Dharma Teja Vooturi, Nataraj Jammalamadaka, Jianyu Huang, Hector Yuen, et al. A study of bfloat16 for deep learning training. *arXiv preprint arXiv:1905.12322*, 2019.

[40] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pages 5156–5165. PMLR, 2020.

[41] Amirhossein Kazemnejad, Inkit Padhi, Karthikeyan Natesan Ramamurthy, Payel Das, and Siva Reddy. The impact of positional encoding on length generalization in transformers. *Advances in Neural Information Processing Systems*, 36:24892–24928, 2023.

[42] Yuri Kuratov, Aydar Bulatov, Petr Anokhin, Ivan Rodkin, Dmitry Sorokin, Artyom Sorokin, and Mikhail Burtsev. Babilong: Testing the limits of llms with long context reasoning-in-a-haystack, 2024. URL https://arxiv.org/abs/2406.10149.

[43] Quoc V. Le, Navdeep Jaitly, and Geoffrey E. Hinton. A simple way to initialize recurrent networks of rectified linear units, 2015. URL https://arxiv.org/abs/1504.00941.

[44] Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, Itay Dalmedigos, Erez Safahi, Shaked Meirom, Yonatan Belinkov, Shai Shalev-Shwartz, et al. Jamba: A hybrid transformer-mamba language model. *arXiv preprint arXiv:2403.19887*, 2024.

[45] Yi Heng Lim, Qi Zhu, Joshua Selfridge, and Muhammad Firmansyah Kasim. Parallelizing non-linear sequential models over the sequence length, 2024. URL https://arxiv.org/abs/2309.12252.

[46] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

[47] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. URL https://arxiv.org/abs/1711.05101.

[48] Loren Lugosch, Mirco Ravanelli, Patrick Ignoto, Vikrant Singh Tomar, and Yoshua Bengio. Speech model pre-training for end-to-end spoken language understanding. *arXiv preprint arXiv:1904.03670*, 2019.

[49] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Byj72udxe.

[50] William Merrill. Sequential neural networks as automata. *arXiv preprint arXiv:1906.01615*, 2019.

[51] William Merrill and Ashish Sabharwal. The expressive power of transformers with chain of thought, 2024. URL https://arxiv.org/abs/2310.07923.

[52] William Merrill and Ashish Sabharwal. A little depth goes a long way: The expressive power of log-depth transformers, 2025. URL https://arxiv.org/abs/2503.03961.

[53] William Merrill, Ashish Sabharwal, and Noah A. Smith. Saturated transformers are constant-depth threshold circuits. *Transactions of the Association for Computational Linguistics*, 10: 843–856, 2022. doi: 10.1162/tacl_a_00493. URL https://aclanthology.org/2022.tacl-1.49/.

[54] William Merrill, Jackson Petty, and Ashish Sabharwal. The illusion of state in state-space models. *arXiv preprint arXiv:2404.08819*, 2024.

[55] William Merrill, Yanhong Li, Tyler Romero, Anej Svete, Caia Costello, Pradeep Dasigi, Dirk Groeneveld, David Heineman, Bailey Kuehl, Nathan Lambert, Chuan Li, Kyle Lo, Saumya Malik, DJ Matusz, Benjamin Minixhofer, Jacob Morrison, Luca Soldaini, Finbarr Timbers, Pete Walsh, Noah A. Smith, Hannaneh Hajishirzi, and Ashish Sabharwal. Olmo hybrid: From theory to practice and back, 2026. URL https://arxiv.org/abs/2604.03444.

[56] Meta AI. The llama 4 herd: The beginning of a new era of natively multimodal ai innovation, 2025. URL https://ai.meta.com/blog/llama-4-multimodal-intelligence/.

[57] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed precision training. *arXiv preprint arXiv:1710.03740*, 2017.

[58] Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering, 2018. URL https://arxiv.org/abs/1809.02789.

[59] Mayank Mishra. Lm engine: A hyper-optimized library for pretraining and finetuning, 2024. URL https://github.com/open-lm-engine/lm-engine.

[60] Mayank Mishra, Matt Stallone, Gaoyuan Zhang, Yikang Shen, Aditya Prasad, Adriana Meza Soria, Michele Merler, Parameswaran Selvam, Saptha Surendran, Shivdeep Singh, et al. Granite code models: A family of open foundation models for code intelligence. *arXiv preprint arXiv:2405.04324*, 2024.

[61] Sajad Movahedi, Vera Milovanović, Shlomo Libo Feigin, Alexander Theus, Thomas Hofmann, Valentina Boeva, T. Konstantin Rusch, and Antonio Orvieto. Fixed-point reasoners: Stable and adaptive deep looped transformers, 2026. URL https://arxiv.org/abs/2606.18206.

[62] Elliot Nelson, Georgios Kollias, Payel Das, Subhajit Chaudhury, and Soham Dan. Needle in the haystack for memory based large language models. *arXiv preprint arXiv:2407.01437*, 2024.

[63] NVIDIA. Nvidia nemotron nano 2: An accurate and efficient hybrid mamba-transformer reasoning model, 2025. URL https://arxiv.org/abs/2508.14444.

[64] NVIDIA Corporation. *Parallel Thread Execution (PTX ISA)*. NVIDIA. URL https://docs.nvidia.com/cuda/parallel-thread-execution/.

[65] J. M. Ortega and W. C. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables*. Society for Industrial and Applied Mathematics, 2000. doi: 10.1137/1.9780898719468. URL https://epubs.siam.org/doi/abs/10.1137/1.9780898719468.

[66] Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc-Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The lambada dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th annual meeting of the association for computational linguistics (volume 1: Long papers)*, pages 1525–1534, 2016.

[67] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[68] Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, et al. Rwkv: Reinventing rnns for the transformer era. *arXiv preprint arXiv:2305.13048*, 2023.

[69] Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah A Smith, and Lingpeng Kong. Random feature attention. *arXiv preprint arXiv:2103.02143*, 2021.

[70] Korbinian Pöppel, Maximilian Beck, and Sepp Hochreiter. Flashrnn: I/o-aware optimization of traditional rnns on modern hardware, 2025. URL `https://arxiv.org/abs/2412.07752`.

[71] Zhen Qin, Songlin Yang, Weixuan Sun, Xuyang Shen, Dong Li, Weigao Sun, and Yiran Zhong. Hgrn2: Gated linear rnns with state expansion. *arXiv preprint arXiv:2404.07904*, 2024.

[72] Liliang Ren, Yang Liu, Yadong Lu, Yelong Shen, Chen Liang, and Weizhu Chen. Samba: Simple hybrid state space models for efficient unlimited context language modeling. *arXiv preprint arXiv:2406.07522*, 2024.

[73] Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S. Gordon. Choice of Plausible Alternatives: An Evaluation of Commonsense Causal Reasoning. In *AAAI Spring Symposium on Logical Formalizations of Commonsense Reasoning*, Stanford University, March 2011. URL `http://ict.usc.edu/pubs/Choice%20of%20Plausible%20Alternatives-%20An%20Evaluation%20of%20Commonsense%20Causal%20Reasoning.pdf`.

[74] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale, 2019. URL `https://arxiv.org/abs/1907.10641`.

[75] Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *International conference on machine learning*, pages 9355–9366. PMLR, 2021.

[76] Jay Shah, Ganesh Bikshandi, Ying Zhang, Vijay Thakkar, Pradeep Ramani, and Tri Dao. Flashattention-3: Fast and accurate attention with asynchrony and low-precision. *Advances in Neural Information Processing Systems*, 37:68658–68685, 2024.

[77] Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

[78] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

[79] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

[80] Julien Siems, Timur Carstensen, Arber Zela, Frank Hutter, Massimiliano Pontil, and Riccardo Grazzi. Deltaproduct: Improving state-tracking in linear rnns via householder products. *arXiv preprint arXiv:2502.10297*, 2025.

[81] Shaden Smith, Mostofa Patwary, Brandon Norick, Patrick LeGresley, Samyam Rajbhandari, Jared Casper, Zhun Liu, Shrimai Prabhumoye, George Zerveas, Vijay Korthikanti, et al. Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model. *arXiv preprint arXiv:2201.11990*, 2022.

[82] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

[83] Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. Retentive network: A successor to transformer for large language models. *arXiv preprint arXiv:2307.08621*, 2023.

[84] Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey, 2022. URL https://arxiv.org/abs/2009.06732.

[85] Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, et al. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025.

[86] Aleksandar Terzić, Nicolas Menet, Michael Hersche, Thomas Hofmann, and Abbas Rahimi. Structured sparse transition matrices to enable state tracking in state-space models. *arXiv preprint arXiv:2509.22284*, 2025.

[87] Philippe Tillet, H. T. Kung, and David Cox. Triton: an intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, MAPL 2019, page 10–19, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450367196. doi: 10.1145/3315508.3329973. URL https://doi.org/10.1145/3315508.3329973.

[88] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[89] Johannes Welbl, Nelson F. Liu, and Matt Gardner. Crowdsourcing multiple choice science questions, 2017. URL https://arxiv.org/abs/1707.06209.

[90] BigScience Workshop, Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, et al. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*, 2022.

[91] Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. In *International conference on machine learning*, pages 10524–10533. PMLR, 2020.

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

[93] Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. *arXiv preprint arXiv:2312.06635*, 2023.

[94] Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. *arXiv preprint arXiv:2412.06464*, 2024.

[95] Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. *Advances in neural information processing systems*, 37:115491–115522, 2024.

[96] Ted Zadouri, Markus Hoehnerbach, Jay Shah, Timmy Liu, Vijay Thakkar, and Tri Dao. Flashattention-4: Algorithm and kernel pipelining co-design for asymmetric hardware scaling, 2026. URL https://arxiv.org/abs/2603.05451.

[97] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence?, 2019. URL https://arxiv.org/abs/1905.07830.

[98] Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in neural information processing systems*, 32, 2019.

[99] Muru Zhang, Mayank Mishra, Zhongzhu Zhou, William Brandon, Jue Wang, Yoon Kim, Jonathan Ragan-Kelley, Shuaiwen Leon Song, Ben Athiwaratkun, and Tri Dao. Ladder-residual: parallelism-aware architecture for accelerating large model inference with communication overlapping, 2025. URL https://arxiv.org/abs/2501.06589v2.

A Related Work

xLSTM and FlashRNN. xLSTM [3] introduces two recurrent cells: mLSTM has a matrix-valued state but replaces the non-linear recurrence with a linear (associative) update, which enables chunkwise-parallel training while sLSTM keeps a non-linear recurrence but restricts the state to a vector. Neither combines a non-linear recurrence with a large matrix-valued state, which is the combination M^{2}RNN targets. M^{2}RNN can be viewed as the simplest parametrization that fills this gap: we take a vector-valued non-linear RNN, expand its state into a matrix via an outer-product write as in linear attention, and add a scalar forget gate that is independent of the recurrent state so that it can be computed in parallel and keeps the per-step computation a single dense GEMM. The same idea can be applied to sLSTM or GRUs. FlashRNN [70] provides I/O-aware kernels for vector-valued non-linear RNNs (LSTM, GRU, sLSTM). Its kernels map the recurrence onto tensor cores by batching sequences (16 per WMMA call), whereas M^{2}RNN's read, write and gate operations reduce directly to dense 16×16 or 64×16 matrix multiplies, independent of the batch size. We compare training throughput against FlashRNN's kernels in Appendix M.

Titans. Titans [4] is also a non-linear recurrent architecture, but its non-linearity comes from a generalized MLP memory module rather than the matrix-valued memory used by most linear attention mechanisms and by M^{2}RNN. An MLP-valued memory is possible in principle within M^{2}RNN, but it is significantly more expensive and, in practice, not trainable without the chunkwise update that Titans relies on. The chunkwise update of Titans could in principle be applied to M^{2}RNN as well, which could significantly improve training throughput at the cost of potentially reduced accuracy.

Parallelizing non-linear RNNs. DEER [45], ELK [26] and ParaRNN [16] parallelize non-linear recurrences over the sequence length by casting them as a system of non-linear equations solved with (quasi-)Newton iterations. These approaches are complementary to M^{2}RNN: they could be combined with the matrix-valued state, although the memory cost of the Jacobian grows with the state size.

Expressivity of Transformers. The TC^0 upper bound applies to fixed-depth Transformers without chain-of-thought. Chain-of-thought [51], logarithmic depth [52] or looping [61] let Transformers go beyond TC^0, at the cost of additional sequential computation.

Expressive linear RNNs. Linear RNNs with input-dependent, non-diagonal transitions can exceed the expressivity of diagonal ones such as Mamba-2 and Gated DeltaNet [28, 54, 80, 86], since iterated products of general unstructured matrices cannot be computed in TC^0. Training such models with unstructured transitions requires constraining the spectral norm of the transition at every step to keep the recurrence stable (equivalent to $|\gamma_t| < 1$ for Mamba-2 and $|1 - \beta_t| < 1$ for Gated DeltaNet because of their structured transitions), whereas the $\tanh$ non-linearity bounds the state in non-linear RNNs.

Hybrid architectures. In our hybrids, attention layers appear to handle exact in-context retrieval while M^{2}RNN layers contribute non-linear state tracking. Olmo Hybrid [55] proposes a similar specialization between linear recurrent layers and global attention layers. M^{2}RNN can further improve the state-tracking ability of such hybrids, allowing them to perform state tracking in the NC^1 complexity class.

B Ablations: Effect of State Size and Forget Gate

We ablate the contributions of state size and the forget gate on the recurrent model performance at the 410M scale, comparing a vanilla RNN, an RNN with a forget gate, M^{2}RNN without a forget gate, and M^{2}RNN (Table 4). The matrix-valued M^{2}RNN model (410M parameters) outperforms the vector-valued RNN (406M parameters) by more than 10 perplexity points on WikiText [49] and 260

points on LAMBADA [66], underscoring the critical role of state size. Both components help, and the matrix-valued state accounts for the larger share of the improvement. These results confirm that expanding the state size is the primary driver of performance improvements in recurrent architectures, with gating mechanisms potentially providing further gains.

Forget gate	Matrix-valued state	Wikitext PPL (↓)	LAMBADA PPL (↓)	State size
✗	✗	33.45	302.34	1,360
✓	✗	26.95	89.44	1,360
✗	✓	26.41	48.03	86,016
✓	✓	**22.97**	**33.79**	86,016

Table 4: Ablation of the forget gate and the matrix-valued state for 410M models.

Attention Pattern	$\mathbf{W}_q$	$\mathbf{W}_k$	$\mathbf{W}_v$	$\mathbf{W}_g$	$\mathbf{W}_f$	$\mathbf{W}_o$	Total (P)	State Size (NKV)
Multi-Head	NKd_{m}	NKd_{m}	NVd_{m}	NVd_{m}	Nd_{m}	NVd_{m}	$\approx N(2K+3V)d_{\text{m}}$	$\approx \frac{PKV}{(2K+3V)d_{\text{m}}}$
Multi-Query	NKd_{m}	Kd_{m}	Vd_{m}	NVd_{m}	Nd_{m}	NVd_{m}	$\approx N(K+2V)d_{\text{m}}$	$\approx \frac{PKV}{(K+2V)d_{\text{m}}}$
Multi-Key	Kd_{m}	NKd_{m}	Vd_{m}	NVd_{m}	Nd_{m}	NVd_{m}	$\approx N(K+2V)d_{\text{m}}$	$\approx \frac{PKV}{(K+2V)d_{\text{m}}}$
Multi-Value	Kd_{m}	Kd_{m}	NVd_{m}	NVd_{m}	Nd_{m}	NVd_{m}	$\approx 3NVd_{\text{m}}$	$\approx \frac{PK}{3d_{\text{m}}}$

Table 5: Number of parameters in all linear projections. N denotes the number of heads and d_{m} denotes the model's embedding dimension. For all our models, we use $K = 64$ and $V = 16$ which results in the largest state size for the multi-value attention formulation for constant number of total parameters (P).

Varying the state size at a fixed parameter count. To isolate the effect of state size, we vary the key and value head dimensions K and V of a 410M M^{2}RNN model, scaling the number of heads N accordingly to keep the parameter count constant at ≈410M. Larger states consistently improve Wikitext perplexity and real-world retrieval accuracy, by up to 7 points on average (Table 6).

		Perplexity (↓)	Real-world retrieval accuracy (↑)						
K	V	Wikitext	FDA	SWDE	NQ	TriviaQA	SQuAD	DROP	Avg
16	16	24.24	2.63	14.22	10.55	46.98	25.47	19.12	19.83
16	32	24.11	4.99	16.47	11.34	47.51	27.55	18.45	21.05
32	16	23.57	6.17	19.89	11.91	51.36	28.65	19.50	22.91
32	32	23.36	8.17	22.95	12.42	50.71	30.83	22.18	24.54
64	16	22.97	11.71	26.73	13.24	50.95	33.21	19.55	25.90
64	32	**22.83**	12.34	29.70	13.40	50.83	34.22	21.90	**27.07**

Table 6: Wikitext perplexity (↓) and real-world in-context retrieval accuracy (↑) of 410M M^{2}RNN models with different K, V.

C Representation Power of M^{2}RNN Layers

Proposition 1: The M^{2}RNN recurrence can represent all tasks representable by non-linear vector-valued RNNs and hence can represent regular languages.

Proof. We show that M^{2}RNN can simulate a vector-valued nonlinear RNN, which is known to recognize all regular languages [50].

Consider the parameter setting

$$f_t = 0, \qquad w_r = \mathbf{0}_V, \qquad q_t = k_t = e_1 \in \mathbb{R}^K,$$

where $e_1 = [1, 0, \ldots, 0]^\top$ is the first standard basis vector.

Under this choice, the matrix-valued recurrence reduces in its first row to

$$\mathbf{Z}_t = \tanh\left(\mathbf{H}_{t-1}\mathbf{W} + \begin{bmatrix} v_t^\top \\ \mathbf{0}_V^\top \\ \vdots \\ \mathbf{0}_V^\top \end{bmatrix}\right), \tag{6}$$

and the output is obtained from the first row,

$$y_t = (\mathbf{Z}_t)_{1,:}.$$

This is precisely a standard vector-valued nonlinear RNN with $\tanh$ activation applied to the first row of the state matrix. Since vector-valued nonlinear RNNs can represent any regular language [50], it follows that M^{2}RNN can represent any regular language. □

D Forget Gate Parameterization

Following LSTMs [35] and GRUs [10], we augment M^{2}RNN with a forget gate to mitigate the vanishing gradient problem. We use a scalar-valued (per-head) forget gate $f_t \in [0, 1]$. While we experimented with a vector-valued forget gate of size K per head, it yielded only marginal improvements while substantially increasing the parameter count. The gated recurrent update is $\mathbf{H}_t = f_t\mathbf{H}_{t-1} + (1 - f_t)\mathbf{Z}_t$, where $f_t = 1$ fully preserves the current state and $f_t = 0$ resets it entirely. Although the importance of forget gates is well established in the literature [10, 13, 35], we provide confirmatory ablations in Appendix B. Note that the forget gate (f_t) is only a function of the input x_t and is independent of the previous state $\mathbf{H}_{t-1}$ unlike LSTMs [35] and GRUs [10]. The forget gate ($f_{t,n}$)[5] (forget gate value for n^{th} head at time step t) for M^{2}RNN layers is computed as:

$$f_{t,n} = \psi(x_t) = \frac{1}{(1 + e^{x_t+\beta_n})^{\alpha_n}} \tag{7}$$

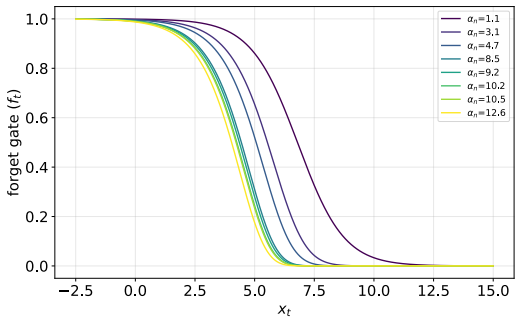

Figure 4: Forget gate (f_t) behavior as a function of the input (x_t) for different values of α_n and β_n. n denotes the nth head.

where α_n and β_n are learnable parameters that are independent of the input. A smaller value of α_n for the nth head results in a slower decay rate. The parameter β_n controls the location at which the forget gate saturates (see Figure 4). A similar gate is used in Mamba-1 [29], Mamba-2 [18] and Gated DeltaNet [94]. Each head is initialized to a different value of α_n and β_n, which leads to different forgetting characteristics. We initialize $\alpha_n \sim \text{Uniform}(\alpha_{\text{min}}, \alpha_{\text{max}})$ and $\beta_n \sim \text{LogUniform}(\beta_{\text{min}}, \beta_{\text{max}})$. Note that when $\alpha_n = 1$, the forget gate simplifies to $\psi(x_t) = \sigma(-x_t - \beta_n)$, which corresponds to a sigmoid with a reflected input and a bias shift.

E Training Matrix-to-Matrix RNN: Initialization Schemes and Stability

Transition Matrix Initialization. We experimented with three initialization schemes for the transition matrix $\mathbf{W}$: normal initialization ($\mathbf{W} \sim \mathcal{N}(0, \sigma^2)$), orthogonal initialization ($\mathbf{W}\mathbf{W}^\top = \mathbf{I}$), and identity initialization ($\mathbf{W} = \mathbf{I}$) [43]. Consistent with findings for vector-valued non-linear RNNs, orthogonal initialization outperforms normal initialization. Identity initialization performs on par with orthogonal initialization, so we adopt it for all models.

[5]We drop the notation for head $f_{t,n}$ in the rest of the paper and use f_t for simplicity.

Gradient Clipping. Training non-linear RNNs with backpropagation through time (BPTT) is susceptible to exploding gradients, as the gradient of the loss with respect to early hidden states involves repeated multiplication by the transition matrix $\mathbf{W}$ and the Jacobian of the $\tanh$ nonlinearity. Although the forget gate f_t attenuates gradients flowing through the state, the transition matrix $\mathbf{W}$ can still cause gradient magnitudes to grow unboundedly. To stabilize training, we apply per-step gradient clipping to the gradient of the recurrent state $\mathbf{H}_t$ during BPTT.

F M^{2}RNN Block

The queries, keys, and values (q_t, k_t, v_t) are computed via linear projections followed by a depthwise causal short-convolution (kernel size 4) and SiLU activation [22, 33]. The output gate g_t uses SiLU without a convolution, and the forget gate f_t uses the parameterized function from Appendix D to ensure $f_t \in [0, 1]$. Crucially, all pre-recurrence projections depend only on the input x_t and can therefore be computed in parallel across the sequence. The full M^{2}RNN layer (for a single head) is:

$$q_t = \text{SiLU}(\text{conv1d}(\mathbf{W}_q x_t + b_q)) \qquad \in \mathbb{R}^{K\times 1} \tag{8}$$

$$k_t = \text{SiLU}(\text{conv1d}(\mathbf{W}_k x_t + b_k)) \qquad \in \mathbb{R}^{K\times 1} \tag{9}$$

$$v_t = \text{SiLU}(\text{conv1d}(\mathbf{W}_v x_t + b_v)) \qquad \in \mathbb{R}^{V\times 1} \tag{10}$$

$$f_t = \psi(\mathbf{W}_f x_t) \qquad \in [0, 1] \tag{11}$$

$$g_t = \text{SiLU}(\mathbf{W}_g x_t) \qquad \in \mathbb{R}^{V\times 1} \tag{12}$$

$$\mathbf{Z}_t = \tanh(\mathbf{H}_{t-1}\mathbf{W} + k_t v_t^\top) \qquad \in \mathbb{R}^{K\times V} \tag{13}$$

$$\mathbf{H}_t = f_t \mathbf{H}_{t-1} + (1 - f_t)\mathbf{Z}_t \qquad \in \mathbb{R}^{K\times V} \tag{14}$$

$$y_t = \mathbf{H}_t^\top q_t + w_r \odot v_t \qquad \in \mathbb{R}^{V\times 1} \tag{15}$$

$$y_{gt} = \text{RMSNorm}(y_t \odot g_t) \qquad \in \mathbb{R}^{V\times 1} \tag{16}$$

$$o_t = \mathbf{W}_o y_{gt} \qquad \in \mathbb{R}^{d\times 1} \tag{17}$$

$$\tag{18}$$

where $x_t \in \mathbb{R}^d$.

Following Mamba-1 [29], Mamba-2 [18], and Gated DeltaNet [94], we adopt a hybrid architecture combining the H3 block [24] with a gated MLP, a design that has proven effective in production LLMs (Figure 1).

G Distributed Training and Systems Optimizations

In this section, we discuss the system considerations for M^{2}RNN layers, including the forward and backward algorithms (Section G.1, G.2) and tensor parallelism (Section G.3).

G.1 Forward Algorithm

We implement the forward computation for the M^{2}RNN layer using the Triton [87] DSL. The kernel parallelizes across the batch dimension B and the number of heads N, assigning each (b, n) pair to a separate GPU SM. We illustrate our algorithm in Algorithm 1 for the case of multi-value formulation ($N_q = N_k = 1$ and $N_v, N_f, N_w = N$) which we use for all our models. We avoid caching intermediate activations during the forward pass for use in backward computation, as storing the $\mathbf{H}$ matrix is prohibitively expensive in both I/O and memory footprint: $\mathbf{H}_t \in \mathbb{R}^{BNKV}$ is much larger than the per-time-step keys and values ($k_t \in \mathbb{R}^{N_k K}, v_t \in \mathbb{R}^{N_v V}$).

G.2 Backward Algorithm

For the backward computation, we recompute the forward recurrence without the final query readout and cache $\mathbf{H}_t$ at every time step to HBM. The backward kernel then reads $\mathbf{H}_t$ at every time step from HBM to compute the gradients for the inputs. The backward kernel is implemented in the Triton DSL [87], which does not expose direct programmability of shared memory. As a result, the kernel

Algorithm 1 Forward Computation for M^{2}RNN layer with multi-value attention formulation. The blue colour indicates computation or tensors residing in on-chip memory.

Input: : $\mathbf{H}_0 \in \mathbb{R}^{BNKV}, \mathbf{Q}, \mathbf{K} \in \mathbb{R}^{BTK}, \mathbf{V} \in \mathbb{R}^{BTNV}, \mathbf{W} \in \mathbb{R}^{NVV}, \mathbf{F} \in [0,1]^{BTN}$
Output: : $\mathbf{Y} \in \mathbb{R}^{BTNV}$

```
for n = 0 to N − 1 in parallel do                          ▷ Each head on a separate SM
  Load W ← W[n, :, :] from HBM                             ▷ Transition matrix to SRAM (reused across batch)
  for b = 0 to B − 1 in parallel do
    Load H ← H_0[b, n, :, :] from HBM                      ▷ Initial hidden state (K × V) to SRAM
    for t = 0 to T − 1 do                                  ▷ Sequential recurrence over time
      Load q, k, v, f ← Q[b, t, :], K[b, t, :], V[b, t, n, :], F[b, t, n] from HBM
      Compute Z ← tanh(HW + kv^⊤)                          ▷ Nonlinear transition with outer product input
      Compute H ← fH + (1 − f)Z                            ▷ Gated update of hidden state
      Compute y ← H^⊤ q                                    ▷ Query readout from hidden state
      Store Y[b, t, n, :] ← y to HBM                       ▷ Write output; no intermediate H cached
    end for
  end for
end for
```

is heavily memory-bandwidth bound; an optimized CUTLASS implementation is left for future work. Note that this means the backward pass does materialize all $\mathbf{H}_t$ in HBM, costing $O(BNKVT)$ memory and I/O during the backward pass for a sequence of length T; avoiding caching only applies to the forward pass. Ideally, the recomputation would be kept entirely on-chip in SRAM, but no chunkwise-parallel form exists for a non-linear recurrence, so the backward pass must reconstruct $\mathbf{H}_t$ by recomputation in reverse order, and Triton does not expose the shared-memory control needed to keep the recomputed states on-chip. Kernels written in TileLang or CuTe-DSL could keep (chunks of) $\mathbf{H}_t$ in SRAM, although caching multiple states per chunk would likely saturate the available SRAM. This limitation is caused by the combination of the non-linearity (which rules out associative recomputation) and the Triton implementation, not by the non-diagonal transition. Diagonal linear RNNs such as Mamba-1 avoid this cost by recomputing their states on-chip with a parallel scan [29]. On Hopper GPUs, intermediates could also be exchanged between SMs through distributed shared memory within thread-block clusters, which is a possible direction for future M^{2}RNN kernels. The algorithm is illustrated in Algorithm 2.

G.3 Tensor Parallelism

Shoeybi et al. [79] introduced tensor parallelism (TP), a model parallelism technique that partitions attention and MLP layers across multiple accelerators. This approach has become standard for training large-scale models [9, 12, 27, 81, 90]. The core challenge in applying TP for M^{2}RNN is that the multi-value formulation shares a single query and key head across all value heads ($N_q = N_k = 1$, $N_v = N$), preventing a straightforward partition of heads across GPUs. We present two strategies for implementing TP in M^{2}RNN layers: a grouped-value approach that avoids extra communication, and a multi-value approach that preserves the parameter count at the cost of additional communication during training. We represent the TP world size as N_{TP}.

G.3.1 TP Topology-Aware M^{2}RNN Layers

A straightforward way to implement TP for M^{2}RNN layers is to adopt a grouped-value formulation: the number of query and key heads is set equal to the TP world size ($N_q = N_k = N_{\text{TP}}$), while the number of value heads remains $N_v = N$. Each GPU is assigned one query head, one key head, and $\frac{N_v}{N_{\text{TP}}}$ value heads[6], and applies RMSNorm [98] with different weights on different GPUs in the TP group (refer to Figure 5). Since this is equivalent to running a multi-value M^{2}RNN recurrence on every GPU, this requires no modifications to the kernels.

Advantages. Each GPU operates independently on its assigned heads, so no communication is introduced beyond the standard TP communication required for an attention layer.

[6]We assume N_v is divisible by N_{TP}.

Algorithm 2 Backward Computation for M^{2}RNN layer with multi-value attention formulation. The blue colour indicates computation or tensors residing in on-chip memory.

Input: : $\mathbf{H}_0 \in \mathbb{R}^{BNKV}, \mathbf{Q} \in \mathbb{R}^{BTK}, \mathbf{K} \in \mathbb{R}^{BTK}, \mathbf{V} \in \mathbb{R}^{BTNV}, \mathbf{W} \in \mathbb{R}^{NVV}, \mathbf{F} \in [0,1]^{BTN}, \mathbf{H}_{\text{full}} \in \mathbb{R}^{BTNKV}$
Output: : $d\mathbf{Q} \in \mathbb{R}^{BTK}, d\mathbf{K} \in \mathbb{R}^{BTK}, d\mathbf{V} \in \mathbb{R}^{BTNV}, d\mathbf{W} \in \mathbb{R}^{NVV}, d\mathbf{F} \in [0,1]^{BTN}$

for $n = 0$ to $N-1$ **in parallel do** ▷ *Each head on a separate SM*
  Load $\mathbf{W} \leftarrow \mathbf{W}[n,:,:]$ from HBM ▷ *Transition matrix to SRAM*
  for $b = 0$ to $B-1$ **in parallel do**
    Load $\mathbf{H} \leftarrow \mathbf{H}_0[b,n,:,:]$ from HBM ▷ *Initial hidden state to SRAM*
    for $t = 0$ to $T-1$ **do** ▷ *Pass 1: forward recomputation, cache all $\boldsymbol{H}_t$*
      Load $k, v, f \leftarrow \mathbf{K}[b,t,:], \mathbf{V}[b,t,n,:], \mathbf{F}[b,t,n]$ from HBM
      Store $\mathbf{H}_{\text{full}}[b,t,n,:,:] \leftarrow f\mathbf{H} + (1-f)\tanh(\mathbf{H}\mathbf{W} + kv^\top)$ to HBM ▷ *Cache hidden state for backward*
    end for
    $d\mathbf{W} \leftarrow 0_{V\times V}$ ▷ *Initialize $d\boldsymbol{W}$ to 0*
    for $t = T-1$ to 0 **do** ▷ *Pass 2: reverse sweep to compute gradients*
      if $t == 0$ **then**
        Load $\mathbf{H}_{\text{prev}} \leftarrow \mathbf{H}_0[b,n,:,:]$ from HBM
      else
        Load $\mathbf{H}_{\text{prev}} \leftarrow \mathbf{H}_{\text{full}}[b,t-1,n,:,:]$ from HBM
      end if
      Load $q, k, v, f, dy \leftarrow \mathbf{Q}[b,t,:], \mathbf{K}[b,t,:], \mathbf{V}[b,t,n,:], \mathbf{F}[b,t,n], \mathbf{dY}[b,t,n,:]$ from HBM
      Compute $\mathbf{Z} \leftarrow \tanh(\mathbf{H}_{\text{prev}}\mathbf{W} + k_t v_t^\top)$ ▷ *Recompute pre-gate activation*
      Compute $\mathbf{H} \leftarrow f\mathbf{H}_{\text{prev}} + (1-f)\mathbf{Z}$ ▷ *Recompute hidden state at t*
      Compute $dq \leftarrow \mathbf{H} \cdot dy$ ▷ *Gradient w.r.t. query*
      Atomic Add $d\mathbf{Q}[b,t,:] \leftarrow dq$ to HBM ▷ *Accumulate across shared heads*
      Compute $r \leftarrow q \cdot dy^\top$ ▷ *Outer product for temporary variable*
      Compute $df \leftarrow 1_K^\top [r \odot (\mathbf{H}_{\text{prev}} - \mathbf{Z})]\, 1_V$ ▷ *Gradient w.r.t. forget gate (scalar)*
      Store $d\mathbf{F}[b,t,n] \leftarrow df$ to HBM
      Compute $d\mathbf{H} \leftarrow r \odot f$ ▷ *Gradient through the gated residual*
      Compute $d\mathbf{Z} \leftarrow r \odot (1-f)$ ▷ *Gradient through the nonlinear branch*
      Compute $d\mathbf{X} \leftarrow d\mathbf{Z} \odot (\mathbf{1}_{K\times V} - \mathbf{Z} \odot \mathbf{Z})$ ▷ *Backprop through* tanh
      Compute $d\mathbf{H} \leftarrow d\mathbf{Z} \odot (d\mathbf{X} \cdot \mathbf{W}^\top)$ ▷ *Gradient w.r.t. previous hidden state*
      Compute $d\mathbf{W} \leftarrow d\mathbf{W} + \mathbf{H}_{\text{prev}}^\top d\mathbf{X}$ ▷ *Accumulate gradient for transition matrix*
      Compute $dk \leftarrow d\mathbf{X} \cdot v$ ▷ *Gradient w.r.t. key*
      Atomic Add $d\mathbf{K}[b,t,:] \leftarrow dk$ to HBM ▷ *Accumulate across shared heads*
      Compute $dv \leftarrow d\mathbf{X}^\top k$ ▷ *Gradient w.r.t. value*
      Store $d\mathbf{V}[b,t,n,:] \leftarrow dv$ to HBM ▷ *Write to HBM*
    end for
    Atomic Add $d\mathbf{W}[n,:,:] \leftarrow d\mathbf{W}$ to HBM ▷ *Accumulate $d\boldsymbol{W}$ across batch*
  end for
end for

Disadvantages. The query and key projection sizes grow proportionally with N_{TP}, coupling the parameter count to the TP topology used during training[7]. Consequently, a model trained with TP world size N_{TP} cannot be straightforwardly deployed at a smaller world size without carrying the inflated query, key projections and the larger activation footprint that comes with them (specifically, the input to the output linear projection). This is analogous to the approach proposed in Mamba-2 [18] and the Desync-Residual strategy[8] [99].

G.3.2 TP Topology-Independent M^{2}RNN Layers

Alternatively, TP can be applied directly to the multi-value formulation ($N_q = N_k = 1$, $N_v = N$). Each GPU processes $\frac{N}{N_{\text{TP}}}$ value heads but shares the same query and key projections. This requires

[7]The RMSNorm weight size also grows with N_{TP} but is significantly smaller in comparison.
[8]Refer to `https://arxiv.org/abs/2501.06589v2` (v2) since later versions of the paper dropped Desync-Residual.

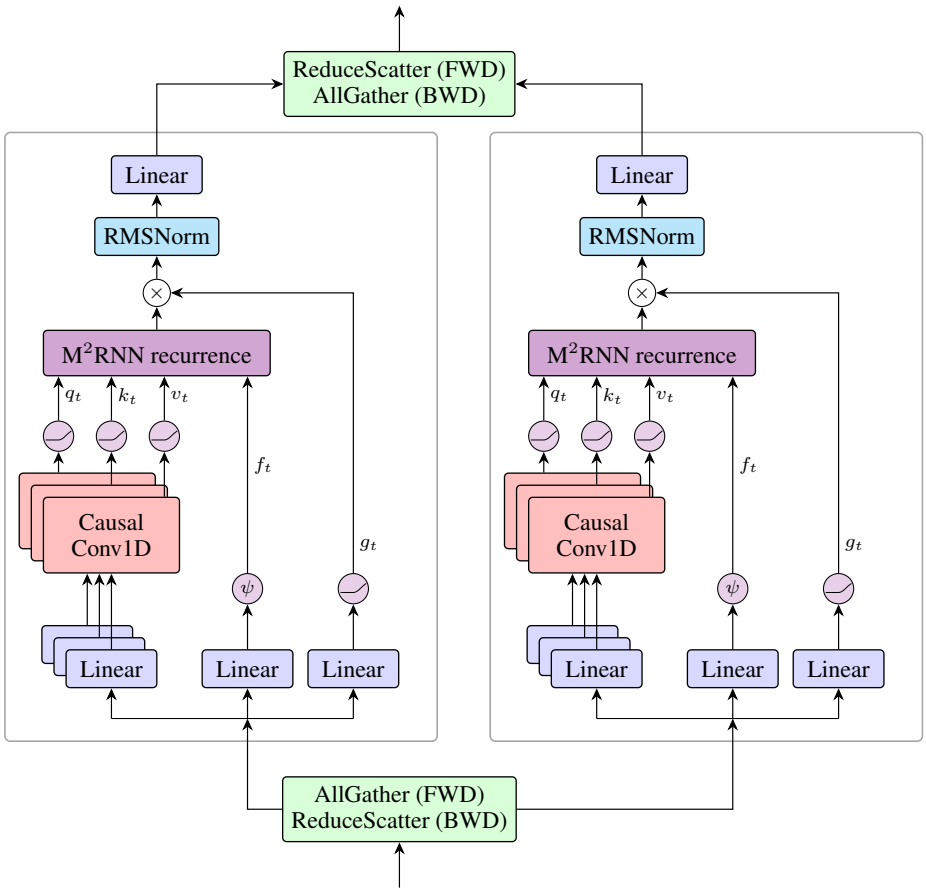

Figure 5: TP topology-aware M^{2}RNN layer running with TP on 2 GPUs. Note that RMSNorm$_1$ and RMSNorm$_2$ have different weights for the RMSNorm module on both GPUs.

replacing RMSNorm [98] modules with RMSNormTP (refer to Appendix H) where the weight for RMSNorm is sharded and an extra communication is introduced in RMSNormTP for the forward and backward computation. The communication in the forward is required to synchronize the RMSNorm normalization term and the synchronization in the backward is required for computing the activation gradient. Note that this idea is applicable to Mamba-2 [18] and Gated DeltaNet [94] layers as well. If the weights for the causal convolution and linear projections for queries and keys are initialized to the same values on all GPUs participating in TP, the sharding method illustrated in Figure 6 ensures that the weights remain synchronized during training.

Advantages. This TP approach preserves the parameter count regardless of the number of GPUs and makes the M^{2}RNN layers independent of the TP topology used during training.

Disadvantages. This approach introduces additional AllReduce communications beyond standard TP. In the forward pass, one extra AllReduce is needed to synchronize the RMSNorm normalization term[9]. The backward pass requires three extra AllReduces: one for the RMSNorm gradient and two to aggregate the query and key activation gradients, since each GPU computes partial gradients for the shared query and key heads from its assigned subset of value heads. Weight gradients for the query and key projections remain naturally synchronized across GPUs throughout training, provided the corresponding weights are initialized identically, so no additional communication is needed for them. One way to reduce backward communication is to make only the RMSNorm module TP

[9]The RMSNorm normalization term depends only on the batch size (B) and sequence length (T), so this communication is lightweight.

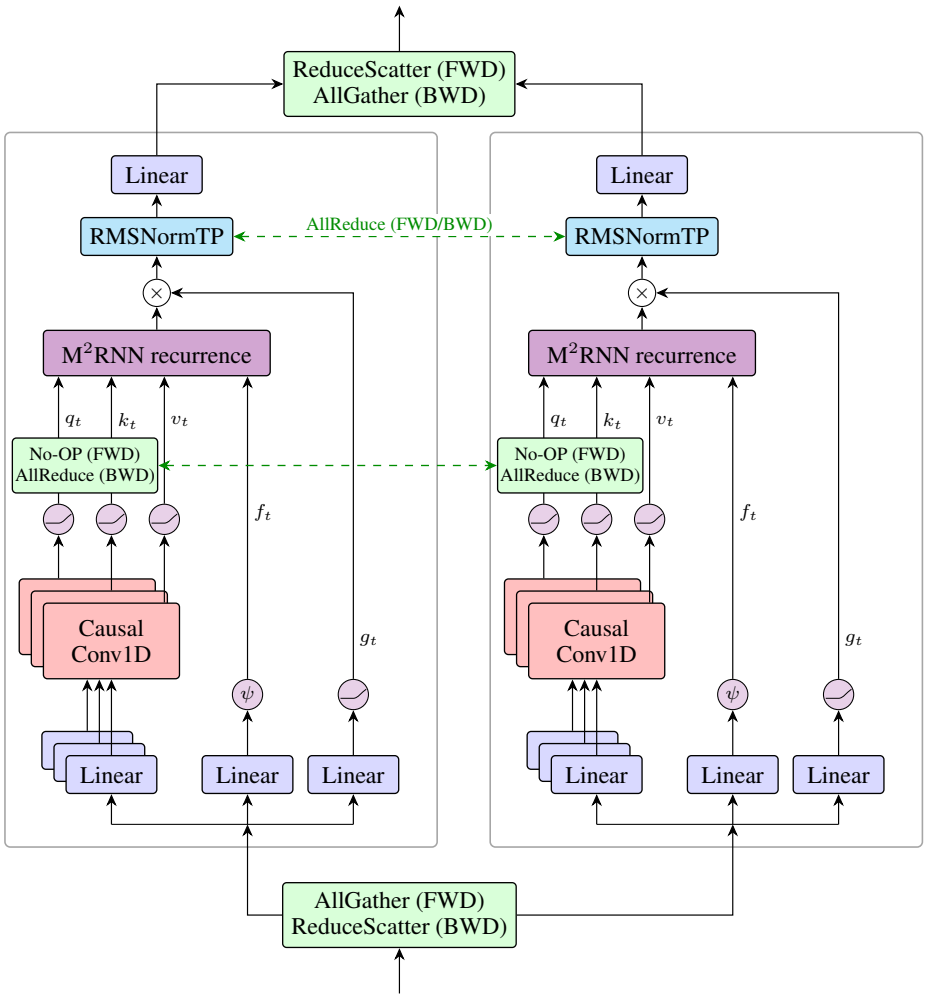

Figure 6: TP topology-independent M^{2}RNN layer. Note that RMSNormTP have weights sharded along the model width dimension for the RMSNorm module on both GPUs requiring a synchronization in both the forward and backward computation.

topology-dependent (as in Section G.3.1); since RMSNorm has very few parameters, this trade-off may be acceptable in practice.

G.4 Training Infrastructure

We use the lm-engine codebase [59] written in PyTorch [67] for training all our models. We also use FlashAttention-3 [76] and SonicMoE [32] with PyTorch compile [1] for training throughput optimization. We also use fast kernels for causal convolutions, Mamba-2, Gated DeltaNet, and M^{2}RNN where applicable. The models use mixed precision [57] in BF16 [39], with gradient AllReduce and accumulation in FP32 for numerical stability.

H RMSNormTP Module

The forward computation for RMSNorm [98] applied on the input activations x is given by:

$$s = \frac{1}{\sqrt{\frac{1}{d}\sum_{j=1}^{d} x_j^2}} \tag{19}$$

$$y_i = (w_i \odot x_i)s \tag{20}$$

where i indicates the ith feature of the input or output, d is the number of features and w is some learnable weight for the RMSNorm module. We cache s during the forward computation in HBM for backward computation. Similarly, the backward computation is given by:

$$p_i = w_i \odot \nabla_{y_i} \mathcal{L} \tag{21}$$

$$r = \sum_{j=1}^{d} p_j x_j \tag{22}$$

$$\nabla_{x_i} \mathcal{L} = s p_i - \frac{r s^3 x_i}{d} \tag{23}$$

$$\nabla_{w_i} \mathcal{L} = (\nabla_{y_i} \mathcal{L} \odot x_i) s \tag{24}$$

To use RMSNorm [98] module with M^{2}RNN layers with Tensor Parallelism (TP), we shard the weight w for RMSNorm among the GPUs participating in TP. Since the incoming activations x are also sharded along the feature dimension, to compute the inverse of the normalization term s, we require an AllReduce operation in the forward. We propose the RMSNormTP module which modifies the RMSNorm module to compute s in forward and r in backward as:

$$s = \frac{1}{\sqrt{\frac{1}{d} \sum_{n=1}^{N_{\text{TP}}} \sum_{j \in D_n} x_j^2}} \tag{25}$$

$$r = \sum_{n=1}^{N_{\text{TP}}} \sum_{j \in D_n} p_j x_j \tag{26}$$

where D_n indicates the sharded features on n^{th} GPU for TP. The summation in blue colour indicates an AllReduce operation across GPUs in Equations 25, 26. Note that for both AllReduce operations, we only need to communicate a single element.

I 410M Dense Model Results

In this appendix, we report results for the 410M dense model, paralleling the 7B MoE discussion in the main paper.

I.1 Language Modeling

Model	Wiki PPL (↓)	LMB PPL (↓)	LAMBADA	HellaSwag	PIQA	ARC-E	ARC-C	WinoGrande	BoolQ	OBQA	COPA	SciQ	Avg Acc
Mamba-2	22.93	38.56	32.78	47.70	70.29	62.04	30.55	53.28	58.69	33.20	69.00	85.50	54.30
Gated DeltaNet	22.70	34.17	34.21	47.16	71.00	61.24	28.07	51.07	58.84	32.80	70.00	85.50	53.99
RNN	33.74	317.38	13.00	44.12	69.70	62.54	30.63	49.72	45.75	33.40	63.00	77.10	48.90
GRU	25.80	63.83	26.10	48.70	71.49	64.90	30.97	49.64	57.68	33.80	71.00	82.90	53.72
M^{2}RNN	22.92	33.63	33.98	47.76	71.60	62.29	30.29	53.43	50.21	34.80	72.00	86.00	54.24
Transformer++	23.32	42.87	33.36	45.02	68.88	59.76	29.01	50.43	54.56	32.00	72.00	84.00	52.90
Hybrid Mamba-2	21.59	35.84	35.32	48.11	70.89	62.54	29.35	52.72	51.41	35.40	65.00	86.80	53.75
Hybrid Gated DeltaNet	21.89	29.02	35.78	47.19	70.67	61.91	29.78	52.17	52.57	34.00	70.00	86.30	54.04
Hybrid M^{2}RNN	21.53	37.31	35.67	48.74	70.29	63.38	30.29	51.46	50.09	33.80	71.00	86.80	54.15
Hybrid Mamba-2	21.59	35.84	35.32	48.11	70.89	62.54	29.35	52.72	51.41	35.40	65.00	86.80	53.75
Hybrid Mamba-2 + M^{2}RNN-1	21.48	36.98	34.47	48.23	70.46	64.14	31.57	54.46	49.54	33.60	70.00	85.80	54.23
Hybrid Mamba-2 + M^{2}RNN-3	21.39	35.77	35.77	48.47	70.57	62.33	31.83	50.75	51.44	33.00	65.00	87.40	53.65
Hybrid Gated DeltaNet	21.89	29.02	35.78	47.19	70.67	61.91	29.78	52.17	52.57	34.00	70.00	86.30	54.04
Hybrid Gated DeltaNet + M^{2}RNN-1	21.39	33.78	34.83	48.11	70.73	61.24	29.95	51.70	48.41	33.20	72.00	87.30	53.75
Hybrid Gated DeltaNet + M^{2}RNN-3	**21.26**	**28.93**	37.18	48.83	69.64	63.85	29.44	53.35	59.14	33.80	71.00	86.10	**55.23**

Table 7: Language Modeling on the 410M parameter dense model across commonsense reasoning benchmarks. All models are evaluated in the 0-shot setting. Accuracies (↑) are higher-is-better.

Homogeneous Architectures. At the 410M scale, Gated DeltaNet achieves the best Wikitext [49] perplexity among all baselines, with M^{2}RNN matching Mamba-2 [18] (within 0.01 points). On LAMBADA [66], M^{2}RNN outperforms both Mamba-2 and Gated DeltaNet [94] by 4.93 and 0.54 perplexity points respectively (Table 7). We attribute M^{2}RNN's gap behind Gated DeltaNet on Wikitext to its $3\times$ smaller state size: while increasing the state size can close this gap, it also raises the per-step FLOP count of the recurrence, making it a less favorable tradeoff in practice. On average downstream accuracy, M^{2}RNN matches Mamba-2 to within 0.06 points while consistently outperforming Gated DeltaNet (Table 7). Vanilla RNN and GRU models significantly underperform M^{2}RNN and other linear RNNs.

2-way Hybrid Models. Combining M^{2}RNN with attention [88] (Hybrid M^{2}RNN) yields substantial gains on Wikitext [49]. Hybrid M^{2}RNN outperforms Hybrid Mamba-2 [18] by 0.06 perplexity points and Hybrid Gated DeltaNet [94] by 0.4 points. Average downstream accuracy also improves over all other hybrids (Table 7).

3-way Hybrid Models. Hybrid Gated DeltaNet + M^{2}RNN-1 improves upon Hybrid Gated DeltaNet, reducing Wikitext perplexity by 0.5 points. We also observe an improvement of 0.1 perplexity points for Hybrid Mamba-2 + M^{2}RNN-1 over Hybrid Mamba-2. Increasing the number of M^{2}RNN layers to 3 yields an additional gain of 0.13 perplexity points (Table 7).

I.2 In-Context Retrieval on RULER

Replacing a few Gated DeltaNet layers with M^{2}RNN layers in the hybrid setting significantly improves retrieval at unseen context lengths on S-NIAH-2, and MQ/MK/MV-NIAH at the 410M parameter scale (Figure 7). Augmenting with M^{2}RNN layers also improves MQ/MK/MV-NIAH accuracy at context lengths seen during training ($\leq 4{,}096$).

A single M^{2}RNN layer with Hybrid Mamba-2 (Hybrid Mamba-2 + M^{2}RNN-1) is insufficient to improve long-context retrieval at the 410M scale. However, replacing three Mamba-2 layers with M^{2}RNN layers yields consistent improvement (Figure 8). We attribute this to the weaker state transition of Mamba-2 requiring more M^{2}RNN layers to compensate.

Model	SQuAD	NQ	DROP	TriviaQA	FDA	SWDE	Avg
Mamba-2	31.2	15.0	19.5	50.5	15.6	32.4	27.4
Gated DeltaNet	34.8	15.4	24.9	51.7	16.3	41.8	<u>30.8</u>
RNN	9.6	5.9	14.6	29.7	0.0	5.0	10.8
GRU	15.8	8.8	17.2	41.5	0.7	7.7	15.3
M^{2}RNN	34.0	13.0	22.9	52.3	10.6	28.5	26.9
Transformer++	14.0	15.5	28.2	52.4	46.0	51.7	34.6
Hybrid Mamba-2	37.6	21.4	26.4	51.7	65.3	61.7	44.0
Hybrid Gated DeltaNet	37.6	22.2	26.0	53.4	61.5	58.0	43.1
Hybrid M^{2}RNN	41.3	22.7	26.8	55.5	74.5	60.7	**46.9**
Hybrid Mamba-2	37.6	21.4	26.4	51.7	65.3	61.7	44.0
Hybrid Mamba-2 + M^{2}RNN-1	38.6	23.8	26.3	55.7	61.3	60.2	44.3
Hybrid Mamba-2 + M^{2}RNN-3	40.4	23.5	26.4	56.7	65.2	61.7	<u>45.6</u>
Hybrid Gated DeltaNet	37.6	22.2	26.0	53.4	61.5	58.0	43.1
Hybrid Gated DeltaNet + M^{2}RNN-1	39.8	23.5	25.6	54.6	67.2	62.5	<u>45.5</u>
Hybrid Gated DeltaNet + M^{2}RNN-3	39.3	21.4	27.2	54.9	65.1	56.5	44.1

Table 8: In-context retrieval performance (↑) on real-world data at the 410M parameter scale.

I.3 In-Context Retrieval on Real-World Data

At the 410M scale, Hybrid M^{2}RNN achieves the largest gain over Transformer++ (12.3 points), outperforming both Hybrid Mamba-2 and Hybrid Gated DeltaNet (Table 8). Adding M^{2}RNN layers to Hybrid Gated DeltaNet (Hybrid Gated DeltaNet + M^{2}RNN) further improves upon Hybrid Gated DeltaNet, and we observe similar improvements for Hybrid Mamba-2 combined with M^{2}RNN.

I.4 Long-Context Performance

On LongBench [2] (Table 9), in the 2-way hybrid setting Hybrid M^{2}RNN matches Hybrid Gated DeltaNet at the 410M parameter scale. Combining M^{2}RNN layers with Hybrid Gated DeltaNet yields up to 8 points of average accuracy improvement across long-context summarization, coding, and few-shot learning tasks. Similar improvements are observable with Hybrid Mamba-2 + M^{2}RNN-3.

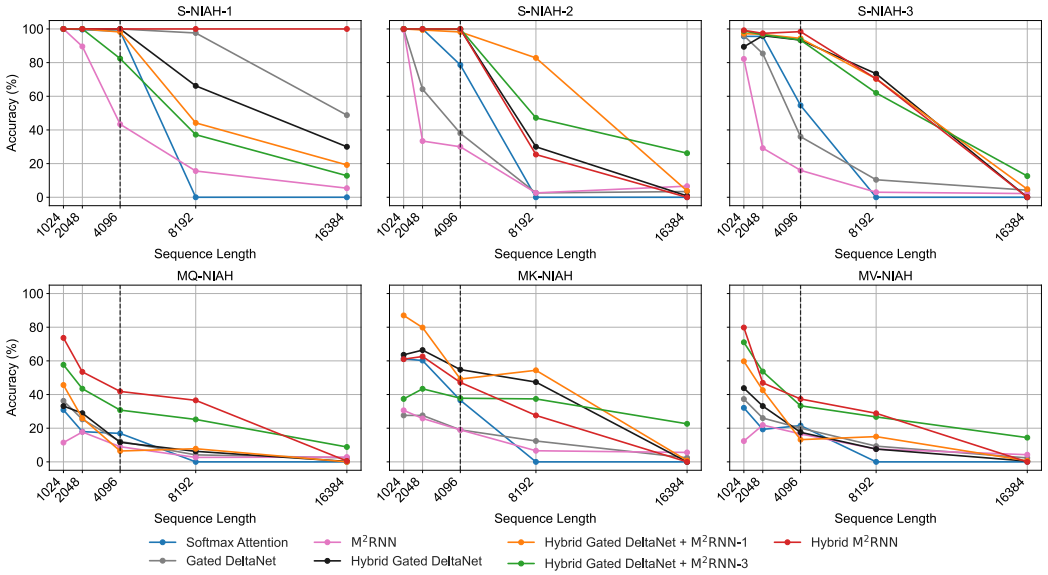

Figure 7: Zero-shot in-context retrieval performance on the RULER benchmark for M^{2}RNN layers and Gated DeltaNet models on 410M parameter scale. The vertical lines at 4096 indicate the training context length.

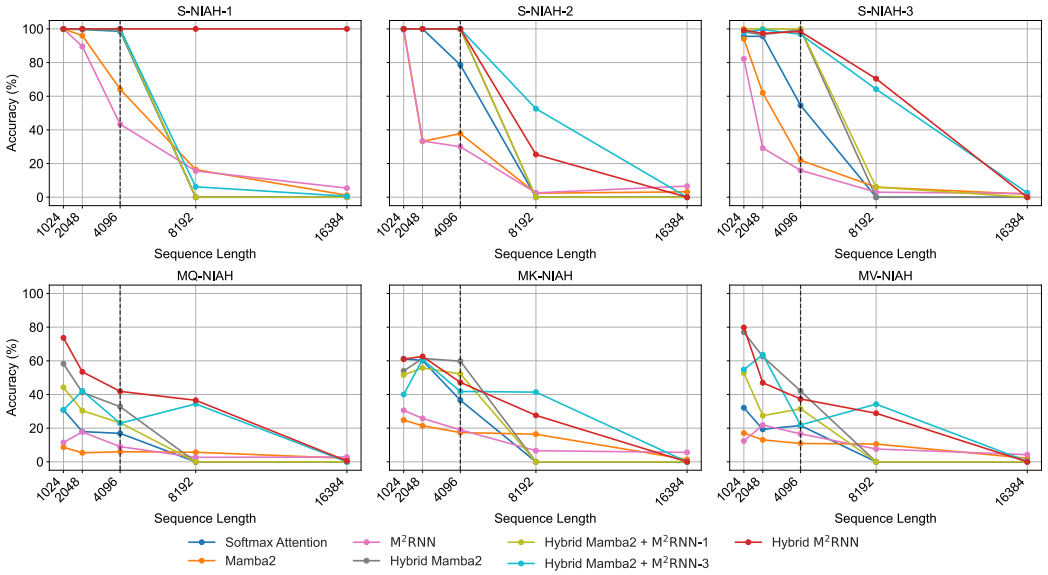

Figure 8: Zero-shot in-context retrieval performance on the RULER benchmark for M^{2}RNN layers and Mamba-2 models on 410M parameter scale. The vertical lines at 4096 indicate the training context length.

J RULER Benchmark Variants

We evaluate the in-context retrieval ability of M^{2}RNN relative to other linear RNNs on the RULER benchmark [36], focusing on the Single Needle In A Haystack (S-NIAH-1, 2, 3) [62], Multi-Query (MQ), Multi-Key (MK), and Multi-Value (MV) variants:

1. **S-NIAH**: A key-value pair is embedded in context and the model is tasked to retrieve the value given the key. S-NIAH-1 uses a synthetic passage with word keys and numeric values, probing

Model	GovReport	QMSum	MultiNews	TREC	TriviaQA	SAMSum	LCC	RepoBench-P	Avg
Mamba-2	6.6	14.3	9.9	26.5	23.2	11.5	13.7	8.9	14.3
Gated DeltaNet	7.6	18.2	12.0	19.0	34.1	21.3	14.9	11.8	<u>17.3</u>
RNN	9.3	15.2	9.2	0.5	12.2	5.0	17.9	14.9	10.5
GRU	7.0	16.5	10.8	1.0	21.4	8.5	12.5	9.0	10.8
M^2RNN	6.3	16.4	10.8	22.0	24.3	13.2	10.7	10.1	14.2
Transformer++	5.6	5.7	6.2	11.0	9.7	4.6	12.0	9.9	8.1
Hybrid Mamba-2	10.0	4.8	10.2	16.0	9.4	9.2	10.5	12.0	10.3
Hybrid Gated DeltaNet	8.2	16.8	16.1	19.5	28.7	15.9	8.7	9.2	<u>15.4</u>
Hybrid M^2RNN	10.0	14.0	7.2	17.5	35.5	14.9	13.4	10.3	<u>15.4</u>
Hybrid Mamba-2	10.0	4.8	10.2	16.0	9.4	9.2	10.5	12.0	10.3
Hybrid Mamba-2 + M^2RNN-1	9.8	9.0	7.5	22.5	13.3	12.6	13.2	9.8	12.2
Hybrid Mamba-2 + M^2RNN-3	10.5	15.8	13.2	29.0	31.7	12.1	10.7	13.3	<u>17.1</u>
Hybrid Gated DeltaNet	8.2	16.8	16.1	19.5	28.7	15.9	8.7	9.2	15.4
Hybrid Gated DeltaNet + M^2RNN-1	9.5	14.8	17.6	47.0	32.1	19.4	7.3	8.5	19.5
Hybrid Gated DeltaNet + M^2RNN-3	16.0	18.1	14.0	43.0	41.5	24.7	12.7	13.9	**23.0**

Table 9: Performance (↑) comparison on the 410M model across long-context summarization, coding and few-shot learning benchmarks.

long-term retention. S-NIAH-2 and S-NIAH-3 use real-world essays with numeric and UUID values, respectively, testing efficient long-context memory management [94].

2. **MQ-NIAH**: In the MQ-NIAH benchmark, multiple key–value pairs are embedded within the passage, and the task requires retrieving all needles associated with distinct keys.
3. **MK-NIAH**: The MK-NIAH benchmark inserts multiple key-value pairs in the passage. The task is to retrieve a single specified key while the other keys serve as distractors.
4. **MV-NIAH**: Multiple values share the same key and the model must retrieve all values associated with that key.

K In-Context Retrieval on RULER for Hybrid Mamba-2 Models (7B MoE)

Figure 9 reports zero-shot RULER accuracy on the 7B (1B active) MoE model for Mamba-2-based hybrids, complementing the Gated DeltaNet-based results in Figure 3 of the main paper.

Hybrid Mamba-2 + M^2RNN: We find that a single M^2RNN layer with Hybrid Mamba-2 (Hybrid Mamba-2 + M^2RNN-1) is insufficient to improve long-context retrieval at either model scale. However, replacing three Mamba-2 layers with M^2RNN layers at 410M and 5 layers at 7B MoE yields consistent improvement (Figures 8, 9). We attribute this to the weaker state transition of Mamba-2 requiring more M^2RNN layers to compensate.

L Additional Long-Context Results

Retrieval with vector-valued non-linear RNNs. On the RULER needle-in-a-haystack tasks (Section 4.4), vector-valued RNN and GRU models achieve close to 0% accuracy. This suggests that a large state is needed for retrieval in non-linear RNNs, but we do not have a mechanistic explanation of how key-value associations written with $k_t v_t^\top$ survive the non-linear state update, and the retrieval accuracy of homogeneous M^2RNN remains below that of linear RNNs with larger states such as Gated DeltaNet (Table 2). In hybrids, attention layers take over exact retrieval (Section 4.4).

BabiLong. Table 10 reports results on the BabiLong benchmark [42] for the 7B MoE models. Hybrid M^2RNN achieves the highest average accuracy, outperforming Hybrid Gated DeltaNet by 3.9 points and Transformer++ by 2.5 points.

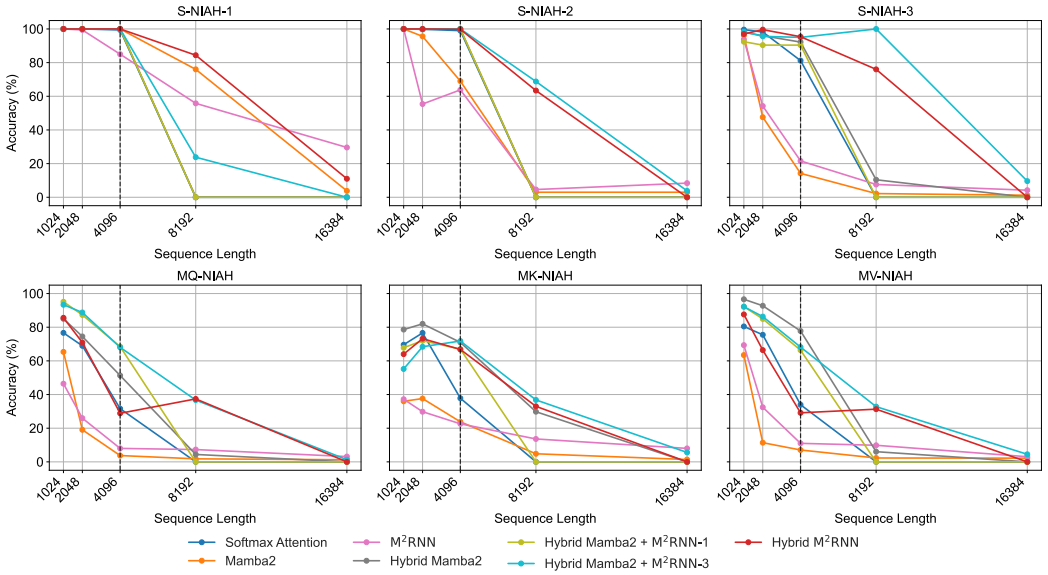

Figure 9: Zero-shot in-context retrieval performance on the RULER benchmark for M^{2}RNN layers and Mamba-2 models for the 7B MoE model. The vertical lines at 4096 indicate the training context length.

Task	Transformer++	Mamba-2	Gated DeltaNet	M^{2}RNN	Hybrid Gated DeltaNet	Hybrid M^{2}RNN
Single supporting fact QA	7.30	2.00	3.90	4.20	4.80	14.00
Two supporting facts QA	2.00	1.80	4.90	1.10	6.91	14.31
Three supporting facts QA	7.11	4.80	9.51	7.61	19.22	11.61
Two argument relations	3.00	2.50	5.21	2.30	15.62	24.02
Three argument relations	18.12	11.61	22.42	22.42	28.73	32.43
Yes/No questions	49.70	55.80	52.20	47.00	58.50	46.30
Counting	48.80	49.50	95.40	48.80	95.60	95.20
Lists and sets	51.20	48.30	54.20	46.50	41.60	53.90
Simple negation	96.90	64.30	63.80	64.10	47.70	66.50
Indefinite knowledge	46.60	43.90	43.90	43.70	46.10	42.10
Track person through temporal references	69.07	51.55	67.47	51.05	61.86	68.67
Conjunction	78.68	44.14	57.86	59.56	45.05	71.87
Compound coreference	71.67	35.24	60.46	66.97	56.86	65.07
Time reasoning	46.75	34.83	47.85	34.83	45.95	37.74
Basic deduction	57.66	54.55	47.65	45.35	57.16	49.35
Basic induction	41.04	51.15	40.04	61.86	37.34	46.75
Positional reasoning	51.95	51.95	51.85	51.95	51.95	51.95
Size reasoning	46.65	46.95	53.05	46.05	45.95	49.65
Path finding	8.41	9.21	8.21	9.21	9.21	9.41
Average	42.24	34.95	41.57	37.61	40.85	**44.78**

Table 10: BabiLong accuracy (↑) for the 7B MoE models.

M Additional Efficiency Results

Comparison with FlashRNN. We benchmark the FlashRNN [70] sLSTM kernels against our M^{2}RNN and GRU kernels for homogeneous models at the 7B scale across context lengths (Table 11). FlashRNN's CUDA-fused and Triton-fused sLSTM kernels run out of memory at 8k and 4k context, respectively, whereas our kernels scale to longer contexts for both GRU and M^{2}RNN. At 2k context, M^{2}RNN already outperforms FlashRNN's CUDA-fused sLSTM (13.19 vs. 12.14) and comes close to its Triton-fused kernel. M^{2}RNN performs more FLOPs per recurrent step than vector-valued RNNs, so our GRU kernel is faster than M^{2}RNN at long contexts (this is the cost of the larger state). Throughput alone does not tell the full story: sLSTM and GRU trail Transformers in perplexity by up to 4 points [70], whereas M^{2}RNN is competitive with linear RNNs (Table 1).

Inference. Table 12 compares the inference throughput and memory of M^{2}RNN and Transformer++ as the batch size grows, with a fixed 1,024-token prompt and 4,096 decoded tokens. Because M^{2}RNN maintains a constant-size state instead of a growing KV cache, its advantage grows with the batch size, reaching $2.67\times$ higher throughput with $0.78\times$ the memory at batch size 32.

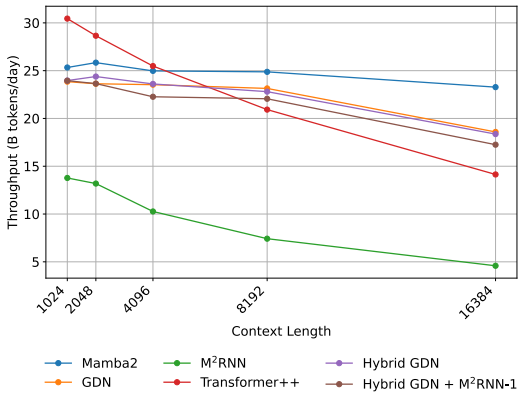

Figure 10: Training throughput measured in billion tokens per day on a machine with 8x NVIDIA HGX H100 GPUs for our 7B MoE configuration.

Model	1024	2048	4096	8192	16384
M^2RNN	13.77	13.19	10.27	7.42	4.59
GRU	12.81	12.62	12.44	10.12	7.63
sLSTM (CUDA)	7.25	5.57	3.19	1.68	0.79
sLSTM (CUDA fused)	15.37	12.14	7.55	OOM	OOM
sLSTM (Triton fused)	21.29	14.65	OOM	OOM	OOM

Table 11: Training throughput (↑) of homogeneous 7B models across context lengths. sLSTM uses the FlashRNN kernels; M^2RNN and GRU use our kernels. OOM denotes out of memory.

Batch size	Transformer++ tok/s	Transformer++ mem (GB)	M^2RNN tok/s	M^2RNN mem (GB)	Speedup	Memory ratio
1	371.58	5.19	**538.99**	**4.70**	1.45×	0.90×
4	910.26	12.11	**1416.98**	**10.09**	1.56×	0.83×
16	1569.26	39.50	**3586.21**	**31.04**	2.28×	0.79×
32	1815.06	77.36	**4847.33**	**60.28**	2.67×	0.78×

Table 12: Inference throughput and memory with a 1,024-token prompt and 4,096 decoded tokens.

Context parallelism. The scan-based context parallelism used for Mamba-2 [18] and Gated DeltaNet [94] relies on their transition being associative, which lets chunk-final states be composed across GPUs with a distributed prefix computation. The non-linear transition of M^2RNN has no associative combination rule, so this approach does not apply. A pipeline-style scheme is feasible instead: each GPU processes its local chunk, and for the M^2RNN layers only, chunk $i+1$ waits on a single $K \times V$ state handoff from chunk i (once per chunk boundary, not per token). This creates a standard pipeline bubble that can be hidden with microbatching. Since M^2RNN layers are only 1–5 of the 40 layers in our hybrids, the remaining layers use ordinary context parallelism (ring attention or Mamba-2 context parallelism), so the bubble affects a small fraction of the model. Microbatching is what fills the pipeline, and the available headroom shrinks as the batch size shrinks. We therefore measure the extreme case with a single microbatch, where the bubble cannot be hidden at all (Table 13). Even with this naive implementation, Hybrid Mamba-2 + M^2RNN-1 scales to 128k tokens at a context-parallel world size of 8 and is 2.6× faster than Transformer++ at 128k; a properly microbatched implementation should improve throughput further. Separately, since M^2RNN reaches its larger state through a large number of heads, tensor parallelism across heads can reduce per-device memory, although scaling to very large tensor-parallel domains incurs a significant communication cost.

Context length	CP world size	Transformer++	Hybrid Mamba-2 + M^{2}RNN-1	Hybrid Mamba-2
4k	1	27.76	25.13	25.46
8k	1	25.23	23.88	25.17
16k	1	20.43	22.52	24.49
32k	2	10.28	15.93	17.95
64k	4	5.41	11.94	15.14
128k	8	2.82	7.32	9.94

Table 13: Training throughput (↑) with context parallelism (CP) using a single microbatch. M^{2}RNN layers use a pipeline-style state handoff; attention and Mamba-2 layers use ring attention and Mamba-2 context parallelism, respectively.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims from abstract are reflected in the paper.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper has a Limitation section at the end of the paper.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.

- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: There is only one theoretical result in the paper and the proof is provided in Appendix C.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Most hyperparameters are discussed in the paper but we will provide the exact configs used to run the experiments in the final version of the paper.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The training code and model checkpoints are publicly released; links are provided on the first page of the paper.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We give a list of all our hyperparameters in detail.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Reporting this is computationally infeasible.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We describe the compute resources in detail for all our experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed the NeurIPS Code of Ethics and our research conforms to it.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: There is no societal impact.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We use the Nemotron-CC-v2 [63] dataset which we cite in the paper. The training codebase is publicly released and linked on the first page of the paper.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Yes, we release our code and models with documentation; links are provided on the first page of the paper.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The paper only uses LLMs for fixing grammatical errors.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.