TransMamba: A Sequence-Level Hybrid Transformer-Mamba Language Model

Yixing Li[2*], Ruobing Xie[1†], Zhen Yang[1], Xingwu Sun[1,3], Shuaipeng Li[1], Weidong Han[1], Zhanhui Kang[1], Di Wang[1], Yu Cheng[2†]

[1]Large Language Model Department, Tencent
[2]The Chinese University of Hong Kong
[3]University of Macau
li.yixing@outlook.com, xrbsnowing@163.com,
{andreasyang, sammsun, jonnyhan}@tencent.com, chengyu@cse.cuhk.edu.hk

Abstract

Transformers are the cornerstone of modern large language models, but their quadratic computational complexity limits efficiency in long-sequence processing. Recent advancements in Mamba, a state space model (SSM) with linear complexity, offer promising efficiency gains but suffer from unstable contextual learning and multitask generalization. Some works conduct layer-level hybrid structures that combine Transformer and Mamba layers, aiming to make full use of both advantages. This paper proposes TransMamba, a novel sequence-level hybrid framework that unifies Transformer and Mamba through shared parameter matrices (QKV and CBx), and thus could dynamically switch between attention and SSM mechanisms at different token lengths and layers. We design the Memory Converter to bridge Transformer and Mamba by converting attention outputs into SSM-compatible states, ensuring seamless information flow at TransPoints where the transformation happens. The TransPoint scheduling is also thoroughly explored for balancing effectiveness and efficiency. We conducted extensive experiments demonstrating that TransMamba achieves superior training efficiency and performance compared to single and hybrid baselines, and validated the deeper consistency between Transformer and Mamba paradigms at sequence level, offering a scalable solution for next-generation language modeling.

Code — https://github.com/Yixing-Li/TransMamba

1 Introduction

Transformers (Vaswani et al. 2017; Achiam et al. 2023; Touvron et al. 2023) are the foundation and mainstream model of modern deep learning (Zhao et al. 2023), showing dominating power in language modeling. Recently, Mamba has emerged (Gu and Dao 2023) and been verified in various fields. Compared with Transformer, Mamba has linear computational complexity, high efficiency in processing long sequences, and lower training and inference costs (Qu et al. 2024). Nevertheless, its contextual learning and multi-task generalization capabilities are unstable (Waleffe et al. 2024). Transformer and Mamba have their own strengths and could complement each other.

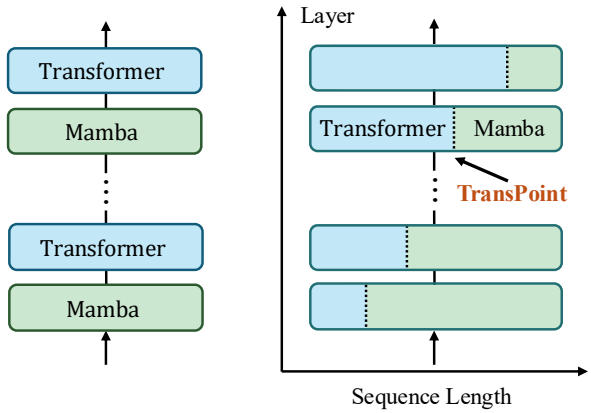
Recent work has explored constructing hybrid structures through mixed layers of Transformer and Mamba to achieve better performance, benefiting from the advantages of both models (Lieber et al. 2024; Team et al. 2025). However, these *layer-level hybrid models* adopt the same stacked structure at all token lengths (Yuan et al. 2024; Yang et al. 2024), which cannot well address the intrinsic efficiency flaws of both models (e.g., Transformer has faster training for short contexts while Mamba has better efficiency in longer contexts, see Table 2). Moreover, the naive static hybrid model has structural restrictions such as layer orders and mandatory ratios (Lieber et al. 2024), which may not be the optimal selection at all token lengths, greatly limiting the exploration and breakthrough of hybrid models.

Recently, Mamba2 (Dao and Gu 2024) further enhances the performance of Mamba series, which reveals the surprising consistency of the attention of Transformer and the State Space Model (SSM) of Mamba. Furthermore, Wang et al. (2024) performed distillation between Mamba and Transformer, distilling the QKV parameters of attention to obtain CBx of SSM, verifying that the parameters can be interactively transferred (as shown in Table 1). These motivate us that we can build a new block that utilizes one set of shared parameters to represent QKV (Transformer mode) or CBx (Mamba mode). This block can flexibly switch between Transformer or Mamba at different layers and tokens. In this case, we could deeply combine two structures *at both the layer level and the sequence level*, taking advantage of both structures to balance effectiveness and efficiency while ensuring structural flexibility.

Based on this block with shared QKV/CBx parameters reflecting two modes, we could introduce a novel ***sequence-level hybrid Transformer-Mamba structure***, where some tokens are processed via Transformer mode, while other are processed via Mamba mode in one layer. Intuitively, to obtain the efficiency advantages of both structures, we can make the block adopt the Transformer mode for training on relatively short contexts and the Mamba mode on long contexts. As shown in Figure 1(b), such prototype framework has only one set of parameters to flexibly switch between Transformer and Mamba modes for LM. In the former to-

[*]Work conducted during internship at Tencent.
[†]Corresponding author.

Copyright © 2026, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.



(a) Layer-level hybrid. (b) Sequence-level hybrid.

Figure 1: (a) Layer-level hybrid. (b) Sequence-level hybrid. QKV/CBx of Transformer and Mamba share parameters.

kens of the sequence, the output is calculated via the Transformer mode. At a specific position in the sequence (which we call ***TransPoints*** from Transformer mode to Mamba mode), the TransMamba model switches to Mamba mode for subsequent token generation.

The implementation of our proposed sequence-level hybrid Transformer-Mamba framework is non-trivial, which has the following challenges: (1) In sequence modeling, the latter mode (Mamba) should rely on the information of the previous tokens encoded by the former mode (Transformer) via an appropriate adaptation method that the latter mode could understand. How to losslessly transfer the knowledge learned by the previous Transformer so that the latter Mamba could fully utilize is essential. (2) We could flexibly decide when (e.g., at what token lengths for different layers) to transfer from Transformer to Mamba in our hybrid framework. Jointly considering effectiveness and efficiency, the selection of an appropriate set of TransPoints at various layers requires careful explorations under insightful principles. (3) Actually, if we set TransPoints separately at different layers, the functioning structure of this sequence-level hybrid framework could be diverse at different token lengths (e.g., pure Transformer, Mamba, or certain layer-level hybrid structures), in which case the possible influence should be concerned.

To address these issues, we propose a novel ***TransMamba*** framework that utilizes the same set of shared parameters to flexibly switch between attention and SSM modes in token generation at different sequence lengths and layers, combining the advantages in effectiveness and efficiency of Transformer and Mamba. It brings in the new *sequence-level hybrid paradigm* in addition to the classical layer-level hybrid paradigm. Specifically, we design a sophisticated ***Memory Converter*** to convert the intermediate results of the Transformer mode into the state readable by the Mamba mode, ensuring lossless transformation of the learned knowledge of previous tokens when passing through TransPoints. Moreover, we have conducted comprehensive research on the ***TransPoint schedule***, exploring the overall optimal TransPoint setting and insights in different layers and sequence lengths. A smoothed TransPoint schedule could reduce the harm brought by mode changes while accelerating model training. Our contributions are summarized as follows:

Attention	SSM
$\mathbf{Q} = \delta(\mathbf{H}\mathcal{W}_{\mathbf{Q}})$	$\mathbf{C} = \delta(\mathbf{H}\mathcal{W}_{\mathbf{C}})$
$\mathbf{K} = \delta(\mathbf{H}\mathcal{W}_{\mathbf{K}})$	$\mathbf{B} = \delta(\mathbf{H}\mathcal{W}_{\mathbf{B}})$
$\mathbf{V} = \delta(\mathbf{H}\mathcal{W}_{\mathbf{V}})$	$\mathbf{X} = \delta(\mathbf{H}\mathcal{W}_{x}) \circ \Delta$
$\mathbf{y} = (\mathbf{L} \circ \mathbf{Q}\mathbf{K}^T)\mathbf{V}$	$\mathbf{y} = (\mathbf{A}^{\times} \circ \mathbf{C}\mathbf{B}^T)\mathbf{X}$

Table 1: Compare the matrix form of the attention and SSM. The core mechanisms of attention and SSM show consistency in dual form, which is the mathematical basis that enables us to unify Transformer and Mamba.

Model	Training FLOPs / Layer
Transformer	$O(\mathrm{T}^2\mathrm{N})$
Mamba	$O(\mathrm{TN}^2)$
TransMamba	$O(\mathrm{P}^2\mathrm{N} + (\mathrm{T}-\mathrm{P})\mathrm{N}^2)$

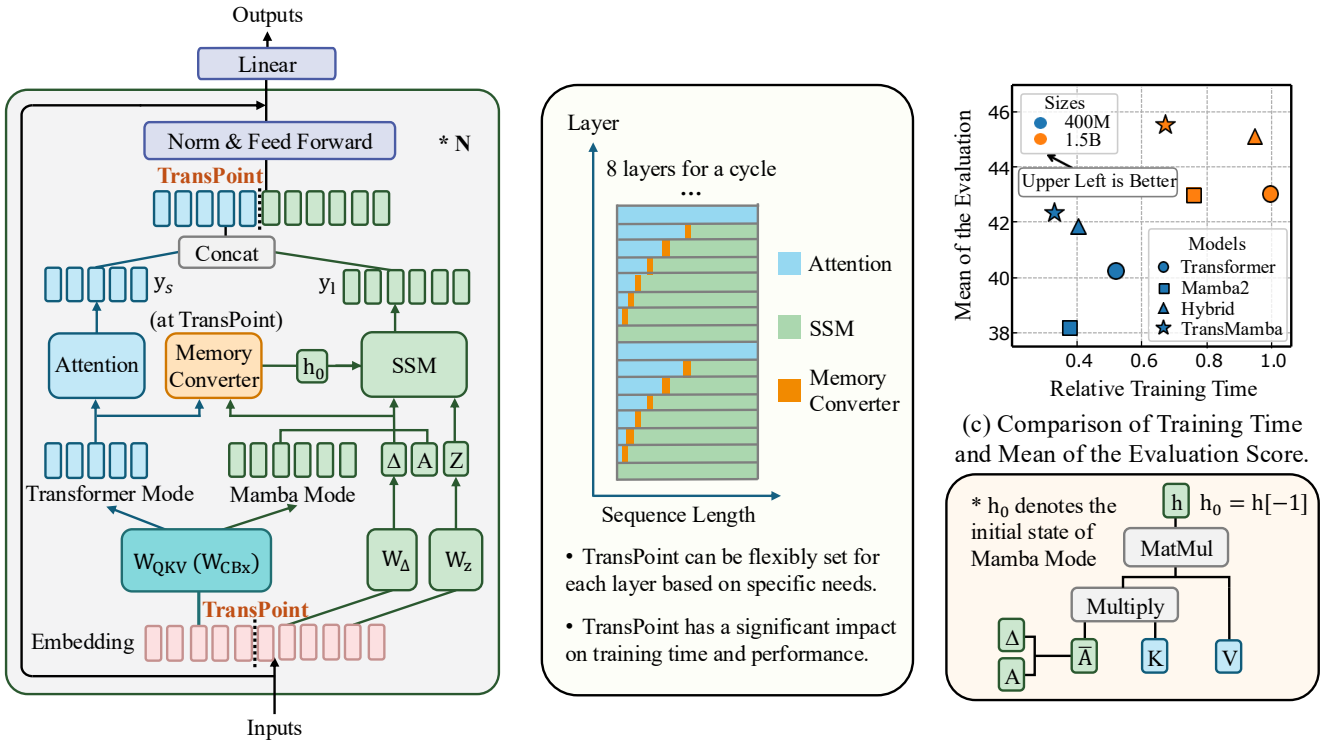
Table 2: Compare the training FLOPs of Transformer, Mamba and optimal TransMamba. The FLOPs of TransMamba is a quadratic function of the TransPoint, and its specific value is related to the speed optimization coefficients of Transformer and Mamba respectively.

- We propose a novel TransMamba framework, which verifies the consistency of Transformer and Mamba in a deeper degree, using one shared set of parameters while outputting tokens via two modes. It introduces a novel sequence-level hybrid Transformer-Mamba structure.
- We design the Memory Converter that conforms to the theoretical solution to ensure the consistency of information in TransMamba during the generation process, and explore the optimal TransPoint scheduling at different layers and token lengths.
- We conduct extensive experiments to verify the advantages of TransMamba on both effectiveness and efficiency. In conclusion, TransMamba could be a promising hybrid Transformer-Mamba structure for LM.

2 Preliminary

2.1 Basic Notions

We use the classic notation from the Transformer and Mamba papers. $\mathbf{QKV}$ denotes the key parameters (query, key, value) of attention (Vaswani et al. 2017), and $\mathbf{L}$ denotes the additional mask matrix. $\mathbf{CBx}$ represents the key parameters in the SSM (Gu and Dao 2023), Δ is used to control the discrete step size in SSM, and $\mathbf{A}$ is used to describe the global dependencies of the hidden state. In order to satisfy the classic symbolic representation, we use $\mathbf{H}$ to denote the input embeddings for attention and SSM. The corresponding calculations are shown in Table 1.



(a) The TransMamba. (b) The TransPoint Scheduling (c) Comparison of Training Time and Mean of the Evaluation Score. (d) The Memory Converter.

Figure 2: (a) Structure of TransMamba. Attention and SSM have shared parameters $\mathbf{W_{QKV}}$ and $\mathbf{W_{CBx}}$. Tokens are either processed via the green path (Mamba mode) or the blue path (Transformer mode). (b) The TransPoint scheduling. (c) TransMamba generally shows better efficiency and performance with different sizes. (d) The Memory Converter.

2.2 Efficiency of Attention and SSM

As shown in Table 2, recent work (Dao and Gu 2024) theoretically summarizes the FLOPs of attention and SSM. T denotes the sequence length, N denotes the state dimension and P denotes the TransPoint value. In the context of long texts where T is much larger than N, Mamba is efficient at training on long sequences due to its linear complexity.

This advantage of Mamba's training efficiency on longer contexts is present with most of the commonly-used model sizes, which also forms the motivation of TransMamba that attempts to unleash the maximum potential of the flexible sequence level hybrid Transformer-Mamba structure in terms of effectiveness and efficiency. It is important to note that conclusions about FLOPs require a condition on the feature dimension, which we strictly adhered to during model setup.

2.3 The Consistency of Attention and SSM and the Idea of Sequence-Level Hybrid Model

Mamba2 (Dao and Gu 2024) compares the underlying mechanisms of Transformer and Mamba, and introduces the dual form of SSM to illustrate the consistency between the two. In Table 1, we can find that the core mechanisms of Transformer and Mamba (attention and SSM) are completely symmetrical. Wang et al. (2024) aligned the QKV of the transformer weights with CBx of Mamba and performed distillation, achieving improved results on chat and long-text benchmarks. This once again shows that the core weights of Transformer and Mamba are transferable and unified.

The above theories and research inspired us to build a bolder framework of TransMamba with a unified architecture of Transformer and Mamba. In our framework, Transformer and Mamba are combined not only at the layer level, but also at the sequence level with parameter sharing. We have designed a sophisticated combination mechanism to make the hybrid model combine naturally and reasonably.

3 Method

3.1 Overall Framework of TransMamba

As shown in Figure 2(a), TransMamba is a layer-stacked decoder-only autoregressive model. Each layer of TransMamba contains all the parameters of Mamba, including the parameters required to calculate C, B, x, A and Δ. Based on the aforementioned consistency between Transformer and Mamba, we boldly let Q/K/V and C/B/x share the same parameters (i.e., Q↔C, K↔B, V↔x). In other words, our model ***has the ability to switch between Transformer and Mamba structures, but with only one set of parameters***.

Specifically, TransMamba contains the crucial Memory Converter used for lossless information conversion when the mode is switched from QKV to CBx (in Section 3.2), armed with our TransPoint schedule that decides whether we should use Transformer mode or Mamba mode at a cer-

tain layer or token length (in Section 3.3). To ensure better training efficiency, we only set a single TransPoint (i.e., the token position where the mode switches from attention to SSM) for each layer. The sequence before the TransPoint is calculated using attention, and the rest is calculated through SSM (Transformer→Mamba, i.e., TransMamba). Complex structures with multiple TransPoints may have more magical properties and effects, which can be explored in future research. At different token lengths, TransMamba could be flexibly regarded as different structures (e.g., pure Transformer, Mamba, or Hybrid Transformer-Mamba).

Formalized calculation process. We denote the hidden state of the input tokens as $\mathbf{h}$. The remaining critical mathematical symbols are the same as given in Section 2.1. TransMamba calculates intermediate results through linear project and convolution modules. Since the parts of the input token sequence that are shorter and longer than the TransPoint will be calculated through different mechanisms, for the sake of clarity, we use different symbols to represent the two parts: (a) For the former input part before TransPoint:

$$\begin{aligned} \mathbf{h_s} &= \mathbf{h}[:\mathbf{TransPoint}], & \mathbf{Q} &= \delta(\mathbf{h_s}\mathcal{W}_\mathbf{Q}), \\ \mathbf{K} &= \delta(\mathbf{h_s}\mathcal{W}_\mathbf{K}), & \mathbf{V} &= \delta(\mathbf{h_s}\mathcal{W}_\mathbf{V}), \end{aligned} \tag{1}$$

The output $\mathbf{y_s}$ will be calculated through the Transformer mode before TransPoint:

$$\mathbf{y_s} = \text{softmax}(\mathbf{Q}\mathbf{K}^T) \cdot \mathbf{V}. \tag{2}$$

(b) For the latter part of the input after TransPoint:

$$\begin{aligned} \mathbf{h_l} &= \mathbf{h}[\mathbf{TransPoint}:], & \Delta &= \sigma(\mathbf{h_l}\mathcal{W}_\Delta + b_\Delta), \\ \overline{\mathbf{A}} &= e^{-\Delta e^{\log \mathcal{W}_\mathbf{A}}}, & \mathbf{C} &= \delta(\mathbf{h_l}\mathcal{W}_\mathbf{C}), \\ \mathbf{B} &= \delta(\mathbf{h_l}\mathcal{W}_\mathbf{B}), & \mathbf{x} &= \delta(\mathbf{h_l}\mathcal{W}_\mathbf{x}), \\ \mathcal{W}_\mathbf{Q} &= \mathcal{W}_\mathbf{C}, \quad \mathcal{W}_\mathbf{K} = \mathcal{W}_\mathbf{B}, & \mathcal{W}_\mathbf{V} &= \mathcal{W}_\mathbf{x}. \end{aligned} \tag{3}$$

Note that **C/B/x** are calculated exactly the same as **Q/K/V**. TransMamba utilizes the Mamba mode to generate outputs $\mathbf{y_l}$ after TransPoint at each layer. The initial state h_0 will be obtained through our proposed Memory Converter as:

$$\begin{aligned} h_0 &= \text{Memory Converter}(\mathbf{K}, \mathbf{V}), & y_k &= \mathbf{C}_k h_k, \\ h_k &= \overline{\mathbf{A}_{k-1}} h_{k-1} + \mathbf{B}_k, \Delta_k x_k, & \mathbf{y_l} &= [y_0, \cdots, y_k], \end{aligned} \tag{4}$$

or in the matrix form as:

$$\mathbf{y_l} = (\mathbf{A}^\times \circ \mathbf{C}\mathbf{B}^T)(\Delta \circ \mathbf{x}). \tag{5}$$

The final output of our TransMamba can be expressed as the combination of $\mathbf{y_s}, \mathbf{y_l}$:

$$\mathbf{y} = [\mathbf{y_s}, \mathbf{y_l}]. \tag{6}$$

Implementation details. For each layer of TransMamba, the model calculates three intermediate results through the shared QKV/CBx parameters, and then performs the attention (before TransPoint) or SSM (after TransPoint) calculations respectively. The detailed calculations of attention/SSM modes are the same in all layers as Eq. (1) and Eq. (3), only with different TransPoint positions. For the remaining components of Mamba (e.g. Delta and Z parameters), they are not shared and only participate in the calculation of the Mamba mode (refer to Appendix E for more experimental results).

3.2 Memory Converter

The Memory Converter aims to convert $\mathbf{K}$ and $\mathbf{V}$ calculated before *TransPoint* into the hidden state $\mathbf{h}$ required for the Mamba mode after TransPoint. First, we expand the mathematical form of SSM in detail:

$$\begin{aligned} \Delta_k &= \sigma(x_k\mathcal{W}_\Delta + b_\Delta), & \overline{\mathbf{A}_k} &= e^{-\Delta_k e^{\log \mathcal{W}_\mathbf{A}}}, \\ \mathbf{B}_k &= \delta(x_k\mathcal{W}_\mathbf{B}), & \mathbf{C}_k &= \delta(x_k\mathcal{W}_\mathbf{C}), \\ h_0 &= \mathbf{B}_0\Delta_0 x_0, & h_k &= \overline{\mathbf{A}_{k-1}} h_{k-1} + \mathbf{B}_k\Delta_k x_k. \end{aligned} \tag{7}$$

Abbreviate h to matrix form as:

$$h = (\mathbf{A}^\times \circ \mathbf{B}^T)(\Delta \circ \mathbf{x}) = (\mathbf{A}^\times \circ \mathbf{B}^T)\mathbf{X}, \tag{8}$$

where $\mathbf{A}^\times$ is the lower triangular matrix obtained by arranging the elements of $\overline{\mathbf{A}}$ (more details are shown in Appendix C). Based on the consistency of the mathematical structure of attention and SSM shown in Section 2.1, we can calculate the estimated hidden state from the intermediate results $\mathbf{K}$, $\mathbf{V}$ of attention as follows:

$$h_s = (\mathbf{A}^\times \circ \mathbf{K}^T)\mathbf{V}. \tag{9}$$

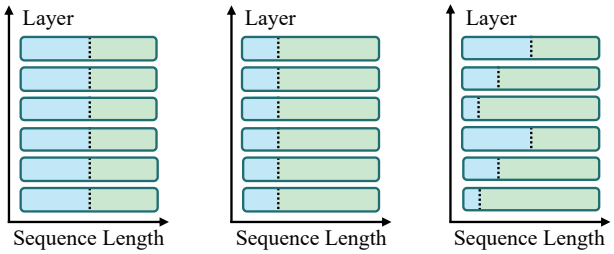
The initial state of the TransPoint can be obtained as $h_0 = h_s[-1]$. Therefore, TransMamba can losslessly transform from attention to SSM during sequence generation. It should be noted that our Memory Converter does not require additional parameters, but is a theoretical solution calculated from existing results.

3.3 Flexible TransPoint Scheduling

TransPoint represents the token position of the segmentation of the sequence where the Transformer→Mamba mode switch happens via the above Memory Converter for each layer. The position of TransPoint in the sequence can control the ratio of attention and SSM in this layer of TransMamba. For example, when TransPoint is set to the midpoint of the sequence, this layer is a 1:1 combination of Transformer and Mamba in the sequence level; if it is set to the beginning of the sequence, this layer is equal to pure Mamba. The TransPoint schedule decides the sequence-level hybrid structure of TransMamba, impacting effectiveness and efficiency.

(1) Principles of our TransPoint scheduling. TransPoint scheduling aims to to maximize the respective advantages of attention and SSM in short and long context training to optimize the overall efficiency and performance. TransPoint scheduling should meet the following requirements:

- **Efficiency.** TransPoint has a great impact on training time. The distribution of TransPoints could be closer to the optimal position in Table 2 for better training efficiency.
- **Effectiveness.** TransPoints at different layers cannot be too concentrated at one position. Under the premise of the first requirement, it needs to be distributed over the entire length of the sequence to prevent possible degradation brought by the mutations of simultaneous Transformer-to-Mamba transformations for better effectiveness.



(a) Layer-shared (4096). (b) Layer-shared (2048). (c) Layer-specific.

Figure 3: Different TransPoint schedules affect effectiveness and efficiency. Finally, TransMamba adopts layer-specific, broad-range, and fine-grained TransPoint schedule.

(2) TransPoint scheduling for efficiency. We set the TransPoints of each layer at different token lengths following the above principles. Suppose the number of layers is L and the total length of sequence is T, there are $L * T$ possible TransPoint schedules for our TransMamba. We denote the value of TransPoint as P (indicating that the tokens before position P are modeled via Transformer and those after P are encoded via Mamba), and we have the FLOPs for this TransMamba layer as follows:

$$\text{FLOPS}_{TransMamba} = O(P^2N + (T-P)N^2). \quad (10)$$

Theoretically, its training time is a quadratic function of the TransPoint position P. In Section 4.3, our experiments confirm that the training efficiency of TransMamba indeed shows a quadratic function trend as TransPoint changes, and the optimal efficient point of TransPoint P is nearly $2,048$ for our setting ($N = 1,536$ and $T = 8,192$). It indicates that TransPoints should be close to $2,048$ for efficiency.

(3) TransPoint scheduling for effectiveness. We find that simply setting the same Transpoint position for all layers (e.g., simultaneously at length $2,048$) will result in unsatisfactory performance due to the sudden switching. To achieve better results, we need to set more diverse TransPoints at the layer level, which could gradually guide the model structure from pure Transformer to Mamba differently at various layers. To enable a smoothed transformation, our TransPoints are placed separately but as close as possible to the optimal efficient P according to Eq. (10). Specifically, our TransPoints cycle is performed every 8 layers referring to the work (Dao and Gu 2024) on layer-level hybrid structure. The mean of TransPoints is set slightly smaller than the optimal efficient point to enable more Mamba layers for better efficiency. TransPoints gradually transition from the beginning to the end of the sequence in a logarithmic trend , ensuring dispersion and smoothness.

Diverse Inference Strategy. Intuitively, the inference of TransMamba could adopt the same TransPoint schedule as that in training. However, due the flexibility of TransMamba, we can also choose completely different TransPoints during inference. It provides us a whimsical but inspiring idea that we can train TransMamba with the most efficient structure, and choose a different structure that best suits the task during inference. We experiment with its potential in Section 4.4.

4 Experiments

4.1 Experiment Setup

Model families and dataset. We developed three baseline model families with various sizes (400M, 1.5B): pure models Transformer (Vaswani et al. 2017), Mamba2 (Dao and Gu 2024), and layer-level hybrid model similar to (Wang et al. 2024) (noted as Hybrid-L). All models are developed based on the Megatron-LM (Shoeybi et al. 2019) library. The pre-training dataset is compiled from Common Crawl (Crawl 2023) and PILE (Gao et al. 2020), and we trained all models for 83 billion tokens.

It should be noted that TransMamba focuses on introducing the novel sequence-level hybrid Transformer-Mamba architecture, and thus we mainly compare with pure models and layer-level hybrid model as baselines. As for other methods such as recent Mamba or Transformer variants (Chou et al. 2024; Pan et al. 2025), sliding windows (Zhu et al. 2021; Liang et al. 2023), etc., they are mostly orthogonal to our idea of sequence-level hybrid mechanism. TransMamba can be flexibly combined with some of these enhanced models, therefore they are not included in the baseline.

Evaluation. We aim to achieve robust conclusions across diverse domains, so we conducted comprehensive evaluations involving 9 English tasks, including ARC-E, ARC-C (Clark et al. 2018), CoQA (Reddy, Chen, and Manning 2019), OBQA (Mihaylov et al. 2018), PIQA (Bisk et al. 2020), PhoneBook (Waleffe et al. 2024), BoolQ (Clark et al. 2019), LongBench-v2 (Bai et al. 2024), MMLU (Bommasani et al. 2021). More details are shown in Appendix D.

4.2 Main Results

Evaluations on General Tasks. We evaluate our TransMamba and baselines on multiple tasks including question answering and reading comprehension as shown in Table 3 (note that the input contexts of these tasks are longer enough than some of our TransPoints to trigger TransMamba). (1) TransMamba achieves the overall best performance. On the question answering and understanding tasks, TransMamba achieves the best performance or is comparable to the Hybrid-L model, while it consistently outperforms the original Transformer and Mamba2. (2) The PhoneBook task is given the contact information of multiple people and requires the model to accurately answer the contact of a specific person. As introduced in (Waleffe et al. 2024), Mamba has a significant disadvantage compared to Transformer in this precise search task, and this disadvantage also brings to Hybrid-L. However, due to our smart combination of Transformer-Mamba at the sequence level, TransMamba can give accurate answers at the beginning of the sequence with almost the same accuracy as Transformer. (3) Table 4 shows the results on the *long-context* benchmark LongBench-v2, where TransMamba still outperforms all baselines.

Efficiency Analysis. As described in Table 2 and Section 2.2, Transformer/Mamba have efficiency advantages on short/long text, respectively. Combining both advantages, TransMamba is more efficient compared to baselines. Specifically, taking max token length $T = 8k$ and state dimension $N = 1.5k$ as an example, the theoretical FLOPs of

Model	ARC-E ACC ↑	ARC-C ACC ↑	CoQA F1-Score ↑	OBQA ACC ↑	PIQA ACC ↑	PhoneBook Similarity ↑	BoolQ ACC ↑	MMLU ACC ↑
Transformer-400M	60.57	58.72	5.07	42.4	52.75	38.70	60.72	40.02
Mamba2-400M	56.15	52.27	4.68	40.8	51.10	13.07	57.51	37.95
Hybrid-L-400M	62.33	55.78	5.52	43.6	53.89	17.60	61.66	39.46
TransMamba-400M	**62.50**	**59.33**	**6.23**	**44.8**	**55.76**	**39.69**	**64.15**	**40.53**
Transformer-1.5B	60.87	59.43	5.93	48.6	56.66	**41.04**	61.42	41.57
Mamba2-1.5B	63.64	56.00	5.30	44.0	58.97	19.08	59.20	40.74
Hybrid-L-1.5B	63.92	57.97	6.21	**51.0**	59.25	26.63	65.48	42.19
TransMamba-1.5B	**64.75**	**63.33**	**6.97**	50.6	**59.61**	40.92	**66.73**	**42.65**

Table 3: Main evaluation results. TransMamba generally shows better performance.

Model	LongBench-v2		
	Overall	Easy	Hard
Transformer	31.61	34.38	29.90
Mamba2	30.62	32.81	29.26
Hybrid-L	35.79	38.02	34.41
TransMamba	**38.76**	**40.10**	**37.94**

Table 4: Evaluation results of our TransMamba and baselines on the long text benchmark LongBench-v2. The number of parameters of all models is 1.5B.

Model	Relative Train Time	Normalized Flops / (Layer, 10^{10})
Transformer	1.00	3.94
Mamba2	0.77	2.01
Hybrid-L	0.78	2.97
TransMamba	**0.75**	**1.85**

Table 5: Comparison of average training time of models.

Transformer is 2.12 times that of the most efficient TransMamba, while that of Mamba is 1.09 times (the FLOPs shown in Table 5 are normalized by the factor 2.67 mentioned in Appendix A). We calculated the average training time of actually implemented models of main experiments on 3 machines in Table 5. TransMamba has a maximum efficiency improvement of 25% compared to Transformer.

4.3 In-depth Analyses on TransPoint Schedule

Analyses on Layer-Shared TransPoint Schedule. TransPoint scheduling has a significant impact on model effectiveness and efficiency. We first conducted experiments on Layer-shared TransPoint scheduling for a straightforward understanding, where the TransPoint of all layers is set to the same value. We studied the relationship between model training time and TransPoint settings (0–8192 with a step size of 64). Figure 4(a) shows relative training time follows a quadratic trend, with the optimal TransPoint at 2048.

V1∼V3 in Table 6 shows different Layer-shared schedules at various positions, whose loss and PPL are not satisfactory due to the simultaneous Transformer→Mamba mutation for all layers. Hence, we move to explore Layer-specific TransPoint scheduling for better performance.

Model Setting		Validation Loss ↓	Validation PPL ↓
Transformer		3.098	22.15
Layer-shared	V1	3.356	28.67
	V2	3.297	27.03
	V3	3.308	27.33
Layer-specific	V4	3.125	22.76
	V5	3.100	22.20
	V6	3.135	22.99
Broad-range	V7	3.084	21.85
	V8	3.022	20.53
Fine-grained	V9	**2.898**	**18.14**

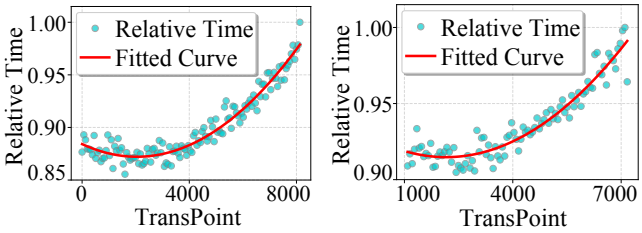
(a) Validation Score of Model Settings

Model Setting	Detailed TransPoint Schedule
V1	[2048]
V2	[4096]
V3	[6144]
V4	[3072, 4096, 5120]
V5	[2048, 3072, 4096]
V6	[512, 1024, 2048]
V7	[2048, 4096, 6144]
V8	[0, 1024, 2048, 6144, 8192]
V9	[0, 128, 256, 512, 1024, 2048, 4096, 8192]

(b) TransPoint Schedule of Model Settings

Table 6: Results of different TransPoint schedules. The input token sequence length of the training data is 8192. The validation loss and PPL is calculated at 21 billion tokens. The TransPoint of each layer in the model cyclically alternates through the predefined TransPoints sequence.

Analyses on Layer-Specific TransPoint Schedule. During explorations on Layer-specific scheduling, we observed that three characteristics can bring better results: ***Layer-***



(a) Layer-Shared TransPoint. (b) Layer-Specific TransPoint.

Figure 4: Experiments on TransPoint scheduling and training efficiency reveal that both layer-shared and layer-specific schedules exhibit a quadratic trend.

Model (400M)	ARC-E	OBQA	PIQA
TransMamba	**62.50**	44.8	**55.76**
TransMamba-Inf @Transformer	27.27	34.8	50.49
TransMamba-Inf @Mamba2	50.82	38.2	50.89
TransMamba-Inf @Hybrid-L	52.20	**45.0**	51.30

Table 7: Results of inconsistent training/inference TransPoint scheduling. Although the "Inf" TransMamba versions perform worse than the original consistent version in bold, the close performance inspires future explorations.

specific, ***Broad-range***, and ***Fine-grained***, and conduct three groups of evaluations (V4∼V9 in Table 6). Note that these Layer-specific schedules also possess the same quadratic curve trend on efficiency as shown in Figure 4(b).

We condensed the following rules for TransPoints scheduling: **(a) TransPoints of each layer should be *layer-specific*.** Setting concentrated TransPoints for all layers has relatively poor validation results. V4∼V6 set TransPoints at three positions and achieve better loss and PPL compared to V1∼V3 with shared TransPoints. **(b) The scheduling of TransPoints should cover *broader range* of the sequence.** V4∼V6 have TransPoints of all layers under the concentrated setting vary within the range of 2K tokens, while the loss and PPL are significantly higher compared to V7∼V8. More diverse TransPoints help the model performance gradually improve. **(c) *Fine-grained* transformation of TransPoints improves the performance.** Compared to the vanilla broad range setting V7 and V8, V9 (i.e., the final TransMamba setting) have finer-grained and smoother scheduling cycling every 8 layers, achieving the best result.

4.4 Explorations on Inconsistent Training/Inference TransPoint Scheduling

Due to the flexibility of the Transformer and Mamba mode transformation based on the unified parameters in TransMamba, we can set different TransPoint schedules for training and inference. For this bold exploration, we train our TransMamba with the selected schedule V9, and then inference with different schedules. As shown in Table 7, TransMamba can still maintain a certain level in most cases under completely different reasoning structures. Even in some cases, such as the score of OBQA evaluated with the Hybrid structure, it can exceed the original TransMamba and all baselines. This gives us a lot of inspiration for future research directions. The structural decoupling of reasoning can bring many research possibilities and unexplored performance. Ideally, we could train one TransMamba model efficiently and inference with different modes/structures according to practical demands and tasks.

Experiment	Training Loss
Global z with h	3.503
Global z with residual	3.447
Attention w/o z	3.390
Memory Converter with MLP	3.209
Memory Converter with SSM	3.173

Table 8: Core parts of ablation studies on model architecture.

4.5 Ablation Study on Other Model Components

We further conduct ablation study on components of TransMamba, and critical conclusions include: (1) In TransMamba, the attention block is not suitable for mapping with the z of SSM; (2) Memory Converter optimization is necessary. The detailed experiments are shown in Appendix E.

5 Related Works

Transformer and Mamba. Transformer has always been the focus of language model research (Beltagy, Peters, and Cohan 2020; Liu et al. 2021; Tang et al. 2024), but its limitations in processing long sequences (Zhou et al. 2021; Behrouz, Zhong, and Mirrokni 2024) and the memory pressure caused by KV cache (Wang et al. 2020; Dao et al. 2022) are also difficult to solve. Mamba has the advantage of linear complexity based on the state space model (Gu and Dao 2023; Zhang et al. 2024), but struggles in modeling complex contexts (Xiao et al. 2024).

Hybrid and other feature mixture models. Hybrid models (Chen et al. 2024; Lou, Fu, and Yu 2024; Ren et al. 2025) that combine the two are emerging, but most of the work simply cascades them (Hatamizadeh and Kautz 2024; Lieber et al. 2024). Works about architecture (Munkhdalai, Faruqui, and Gopal 2024; Dong et al. 2024) combine the features of Transformer and Mamba, but they are essentially different from our sequence-level hybrid.

Consistency between Transformer and Mamba. Recent works (Dao and Gu 2024; Han et al. 2024; Wang et al. 2024) have revealed the consistency of the underlying mathematics between them. However, there is no work that truly attempts to unify Transformer and Mamba in sequence level. More details of related work are shown in Appendix B.

6 Conclusion

We proposes TransMamba to unify Transformer and Mamba at the sequence level and proves its superiority in efficiency and performance. Furthermore, we conduct a detailed exploration of TransPoints and summarize three criteria of TransPoint Scheduling. In short, our attempt provides insight and inspiration for the next generation of sequence modeling.

A Detailed Conclusion, Future Work and Limitations

This paper proposes TransMamba to unify Transformer and Mamba at the sequence level and proves its superiority in efficiency and performance. Specifically, we combine the advantages of Transformer and Mamba in long and short contexts to significantly improve the training speed during training. At the same time, conversion modules such as Memory Converter ensure lossless model conversion, with the TransPoint scheduling to ensure model performance.

We conduct a detailed exploration of TransPoints and model framework, from naive layer-shared TransPoint scheduling to sophisticated layer-specific design, and summarize three standards of the TransPoint scheduling. At the same time, due to the flexible model architecture of TransMamba under shared parameters, we boldly tried training and reasoning isomorphism and obtained surprising results. In short, our attempt provides insight and inspiration for the next generation of sequence modeling.

This paper also has some limitations for future research, including: (a) Because the model architecture of TransMamba is completely redesigned and novel, we needed to pre-train and fine-tune the model from scratch. However, due to resource constraints, we cannot scale to a 7B or larger models. Therefore, we conducted comprehensive and thorough research and ablation experiments on various aspects of TransMamba, focusing on models under 2B (which is comparable with the main Mamba models explored in (Dao and Gu 2024; Wang et al. 2024)). Future work could try larger models and explore the form of the scaling law of TransMamba. (b) Transformer and Mamba have their own (enhanced) variants. These models are orthogonal to our architecture TransMamba, which is why we do not include them in the baseline (mentioned in Sections 4.1 and B). But follow-up work can try to combine different enhanced Transformer/Mamba variants into our TransMamba framework. In addition, the constant term introduced by the Transformer and Mamba optimizations is not currently calculated in detail in TransMamba's theoretical FLOPs. We can perform targeted optimizations in the future. (Based on current experimental results, the constant term ratio is approximately Transformer: Mamba = 2.67:1).

B Full Related Work and Discussion

Transformer and Mamba. Transformer has always been the focus of language model research (Beltagy, Peters, and Cohan 2020; Liu et al. 2021; Tang et al. 2024), but its limitations in processing long sequences (Zhou et al. 2021; Behrouz, Zhong, and Mirrokni 2024) and the memory pressure caused by KV cache (Wang et al. 2020; Dao et al. 2022) are also difficult to solve. Mamba has the advantage of linear complexity based on the state space model (Gu and Dao 2023; Zhang et al. 2024), but struggles in modeling complex contexts (Xiao et al. 2024).

Hybrid and other feature mixture models. Hybrid models (Chen et al. 2024; Lou, Fu, and Yu 2024; Ren et al. 2025) that combine the two are emerging, but most of the work simply cascades them (Hatamizadeh and Kautz 2024; Lieber et al. 2024). Works about architecture (Munkhdalai, Faruqui, and Gopal 2024; Dong et al. 2024) combine the features of Transformer and Mamba. Among them, Infini-Transformer (Munkhdalai, Faruqui, and Gopal 2024) reshapes the memory mechanism and state and concatenates the features of the two. Hymba (Dong et al. 2024) splits the features and calculates them through attention and SSM respectively, and then takes the average as the output of this layer. But they are essentially different from our sequence-level hybrid.

Consistency between Transformer and Mamba mechanisms. Recent work (Dao and Gu 2024; Han et al. 2024; Wang et al. 2024) has revealed the consistency of the underlying mathematics between them. MILA (Han et al. 2024) integrate the advantages of Mamba's forget gate and block design into linear attention and reformulate Mamba as a variant of the linear attention Transformer. (Wang et al. 2024) presents a method to distill large pretrained Transformer models into Mamba for efficient deployment. It involves reusing linear projection weights from Transformer attention layers to initialize a modified Mamba architecture. However, there is no work that truly attempts to unify Transformer and Mamba in sequence level.

C Model Details, Efficiency, and Feasibility of TransMamba.

Memory Converter. The matrix utilized in Equation 8 is as follows:

$$\mathbf{A}^{\times} = \begin{bmatrix} \frac{1}{\overline{\mathbf{A}}_1} & & & \\ \frac{1}{\overline{\mathbf{A}}_2\overline{\mathbf{A}}_1} & \frac{1}{\overline{\mathbf{A}}_2} & & \\ \frac{1}{\overline{\mathbf{A}}_3\overline{\mathbf{A}}_2\overline{\mathbf{A}}_1} & \frac{1}{\overline{\mathbf{A}}_3\overline{\mathbf{A}}_2} & \frac{1}{\overline{\mathbf{A}}_3} & 1 \end{bmatrix}. \tag{11}$$

Feasibility of TransMamba. The feasibility of our design comes from three key points: (1) Due to the consistency of the attention and SSM mechanisms verified in Section 2.3, the output of TransMamba can be calculated either through the Transformer or Mamba mode flexibly for different tokens. (2) Due to the power of our Memory Converter, TransMamba does not lose any information when converting from attention to SSM as the token length increases across the TransPoint. The sequence state required by SSM can be perfectly preserved by the $\mathbf{K}$ and $\mathbf{V}$ of attention. (3) Our exploration and insight in TransPoint schedule help to achieve better effectiveness and efficiency performance.

D Experiment Details

Experiment Setup Details. The benchmarks used in this paper introduced in Section 4.1 are described as follows:

- ARC (Clark et al. 2018) dataset, developed by the Allen Institute for Artificial Intelligence (AI2), is a collection of 5,197 elementary-level science questions designed to evaluate natural language understanding and reasoning capabilities.
- CoQA (Reddy, Chen, and Manning 2019) dataset is a large-scale collection of 127,000 question-answer pairs from 8,000 dialogues across seven diverse domains (e.g., news, literature, science), designed to evaluate machines'

ability to answer context-dependent, free-form questions in multi-turn conversations.

- OBQA (Mihaylov et al. 2018) dataset is a novel question-answering benchmark designed to evaluate AI systems' ability to integrate external commonsense knowledge and perform multi-step reasoning.
- PIQA (Bisk et al. 2020) dataset is a benchmark designed to evaluate AI systems' reasoning capabilities about physical commonsense knowledge through context-dependent questions.
- PhoneBook is introduced in (Waleffe et al. 2024) and aims to evaluate the exact phone number of a specific person given a phone book of multiple people. There are two ways to construct a specific phone book: conventional construction and reverse construction. We use the open source tool (Faraglia and Other Contributors 2020) to construct a completely random test set PhoneBook.
- BoolQ (Clark et al. 2019) dataset is a natural language understanding benchmark comprising 15,942 yes/no questions paired with contextual paragraphs.
- LongBench-v2 (Bai et al. 2024) dataset is a comprehensive multilingual benchmark designed to evaluate large language models' deep understanding and reasoning capabilities in ultra-long contexts (8k–2M words) through 503 challenging multiple-choice questions spanning six task categories.
- MMLU (Bommasani et al. 2021) is a benchmark designed to measure text models' multitask accuracy, covering 57 tasks such as elementary mathematics, US history, computer science, and law.

Settings	Value
Global Batch Size	1024
Micro Batch Size	2
Sequence Length	8192
Training Tokens	81B
MLP Ratio	0.5
Initial LR	2.5e-4
Min LR	2.5e-5
LR Decay Style	cosine
Weight Decay	0.1
Clip Grad	1.0
Normalization	RMSNorm
Adam β_1	0.9
Adam β_2	0.95
BF16	True
RoPE θ	10000

Table 9: Global parameter settings.

Model Parameters Setting. In this section we introduce the parameter settings of the baseline and TransMamba used in this paper. Empirically, we set the ratio of TransMamba layer to MLP layer to 1:1. (The baseline also has the same setting). Table 9 shows the overall training settings, and Table 10 shows the specific parameters of each model size.

Model Setting		Value
400M	Num-Layers	24
	Hidden-Size	1536
	FFN-Hidden-Size	4096
	Num-attention-Heads	16
1.5B	Num-Layers	64
	Hidden-Size	1536
	FFN-Hidden-Size	4096
	Num-attention-Heads	16

Table 10: Model parameter setting.

E Model Components

In the process of building TransMamba, we conducted detailed experiments on each component of the model. We mainly study two model settings: parameter sharing strategy and memory converter structure setting. Table 8 shows some of the key results. The experimental setup is as follows:

1. The "Global z" option experiments with sharing z globally (between the Transformer and Mamba). Global z sharing can be implemented in two ways: by mapping z only with the hidden state h, and by adding a global residual to the previous approach.
2. The "Attention without z" option does not share z in Mamba with the Transformer.
3. The "Memory Converter" experiment explores its specific implementation. "MLP" refers to using the MLP as the approximate state projection. "SSM" refers to the computation method described in this paper.
4. In addition to the main results listed in Table 8, we also conducted additional ablation experiments, such as whether the memory converter is accelerated using the original Mamba2 method. The table only shows the parameters that have the greatest impact on the model architecture.

Our most critical conclusions include: (1) In TransMamba, the attention block is not suitable for mapping with the z of SSM; (2) Memory Converter optimization is necessary. After we modified from a simple MLP fitting to the theoretical solution Memory Converter, the running speed and training effect of the model were significantly improved. In addition, the SSM block acceleration mentioned in the Mamba paper can also be used for Memory Converter.

Acknowledgments

Ruobing Xie is supported by the Young Elite Scientists Sponsorship Program by CAST (2023QNRC001).

References

Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F. L.; Almeida, D.; Altenschmidt, J.; Altman, S.; Anadkat, S.; et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Bai, Y.; Tu, S.; Zhang, J.; Peng, H.; Wang, X.; Lv, X.; Cao, S.; Xu, J.; Hou, L.; Dong, Y.; et al. 2024. LongBench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. *arXiv preprint arXiv:2412.15204*.

Behrouz, A.; Zhong, P.; and Mirrokni, V. 2024. Titans: Learning to memorize at test time. *arXiv preprint arXiv:2501.00663*.

Beltagy, I.; Peters, M. E.; and Cohan, A. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Bisk, Y.; Zellers, R.; Gao, J.; Choi, Y.; et al. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, 7432–7439.

Bommasani, R.; Hudson, D. A.; Adeli, E.; Altman, R.; Arora, S.; von Arx, S.; Bernstein, M. S.; Bohg, J.; Bosselut, A.; Brunskill, E.; Brynjolfsson, E.; Buch, S.; Card, D.; Castellon, R.; Chatterji, N. S.; Chen, A. S.; Creel, K. A.; Davis, J.; Demszky, D.; Donahue, C.; Doumbouya, M.; Durmus, E.; Ermon, S.; Etchemendy, J.; Ethayarajh, K.; Fei-Fei, L.; Finn, C.; Gale, T.; Gillespie, L. E.; Goel, K.; Goodman, N. D.; Grossman, S.; Guha, N.; Hashimoto, T.; Henderson, P.; Hewitt, J.; Ho, D. E.; Hong, J.; Hsu, K.; Huang, J.; Icard, T. F.; Jain, S.; Jurafsky, D.; Kalluri, P.; Karamcheti, S.; Keeling, G.; Khani, F.; Khattab, O.; Koh, P. W.; Krass, M. S.; Krishna, R.; Kuditipudi, R.; Kumar, A.; Ladhak, F.; Lee, M.; Lee, T.; Leskovec, J.; Levent, I.; Li, X. L.; Li, X.; Ma, T.; Malik, A.; Manning, C. D.; Mirchandani, S. P.; Mitchell, E.; Munyikwa, Z.; Nair, S.; Narayan, A.; Narayanan, D.; Newman, B.; Nie, A.; Niebles, J. C.; Nilforoshan, H.; Nyarko, J. F.; Ogut, G.; Orr, L.; Papadimitriou, I.; Park, J. S.; Piech, C.; Portelance, E.; Potts, C.; Raghunathan, A.; Reich, R.; Ren, H.; Rong, F.; Roohani, Y. H.; Ruiz, C.; Ryan, J.; R'e, C.; Sadigh, D.; Sagawa, S.; Santhanam, K.; Shih, A.; Srinivasan, K. P.; Tamkin, A.; Taori, R.; Thomas, A. W.; Tramèr, F.; Wang, R. E.; Wang, W.; Wu, B.; Wu, J.; Wu, Y.; Xie, S. M.; Yasunaga, M.; You, J.; Zaharia, M. A.; Zhang, M.; Zhang, T.; Zhang, X.; Zhang, Y.; Zheng, L.; Zhou, K.; and Liang, P. 2021. On the Opportunities and Risks of Foundation Models. *ArXiv*.

Chen, J.; Zhang, Z.; Yu, J.; Huang, H.; Zhang, R.; Xu, X.; Sheng, B.; and Yan, H. 2024. DSDFormer: An Innovative Transformer-Mamba Framework for Robust High-Precision Driver Distraction Identification. *arXiv preprint arXiv:2409.05587*.

Chou, Y.; Yao, M.; Wang, K.; Pan, Y.; Zhu, R.-J.; Wu, J.; Zhong, Y.; Qiao, Y.; Xu, B.; and Li, G. 2024. MetaLA: Unified optimal linear approximation to softmax attention map. *Advances in Neural Information Processing Systems*, 37: 71034–71067.

Clark, C.; Lee, K.; Chang, M.-W.; Kwiatkowski, T.; Collins, M.; and Toutanova, K. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*.

Clark, P.; Cowhey, I.; Etzioni, O.; Khot, T.; Sabharwal, A.; Schoenick, C.; and Tafjord, O. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Crawl, C. 2023. Common Crawl Dataset (CC-MAIN-2023-14). https://commoncrawl.org/. Accessed: 2025-08-02.

Dao, T.; Fu, D.; Ermon, S.; Rudra, A.; and Ré, C. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35: 16344–16359.

Dao, T.; and Gu, A. 2024. Transformers are SSMs: Generalized Models and Efficient Algorithms Through Structured State Space Duality. *ArXiv*, abs/2405.21060.

Dong, X.; Fu, Y.; Diao, S.; Byeon, W.; Chen, Z.; Mahabaleshwarkar, A. S.; Liu, S.-Y.; Van Keirsbilck, M.; Chen, M.-H.; Suhara, Y.; et al. 2024. Hymba: A hybrid-head architecture for small language models. *arXiv preprint arXiv:2411.13676*.

Faraglia, D.; and Other Contributors. 2020. Faker. GitHub repository.

Gao, L.; Biderman, S.; Black, S.; Golding, L.; Hoppe, T.; Foster, C.; Phang, J.; He, H.; Thite, A.; Nabeshima, N.; et al. 2020. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

Gu, A.; and Dao, T. 2023. Mamba: Linear-Time Sequence Modeling with Selective State Spaces. *ArXiv*, abs/2312.00752.

Han, D.; Wang, Z.; Xia, Z.; Han, Y.; Pu, Y.; Ge, C.; Song, J.; Song, S.; Zheng, B.; and Huang, G. 2024. Demystify mamba in vision: A linear attention perspective. *arXiv preprint arXiv:2405.16605*.

Hatamizadeh, A.; and Kautz, J. 2024. Mambavision: A hybrid mamba-transformer vision backbone. *arXiv preprint arXiv:2407.08083*.

Liang, X.; Tang, Z.; Li, J.; and Zhang, M. 2023. Open-ended Long Text Generation via Masked Language Modeling. In Rogers, A.; Boyd-Graber, J.; and Okazaki, N., eds., *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 223–241. Toronto, Canada: Association for Computational Linguistics.

Lieber, O.; Lenz, B.; Bata, H.; Cohen, G.; Osin, J.; Dalmedigos, I.; Safahi, E.; Meirom, S.; Belinkov, Y.; Shalev-Shwartz, S.; et al. 2024. Jamba: A hybrid transformer-mamba language model. *arXiv preprint arXiv:2403.19887*.

Liu, Z.; Lin, Y.; Cao, Y.; Hu, H.; Wei, Y.; Zhang, Z.; Lin, S.; and Guo, B. 2021. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, 10012–10022.

Lou, M.; Fu, Y.; and Yu, Y. 2024. SparX: A Sparse Cross-Layer Connection Mechanism for Hierarchical Vision Mamba and Transformer Networks. *arXiv preprint arXiv:2409.09649*.

Mihaylov, T.; Clark, P.; Khot, T.; and Sabharwal, A. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*.

Munkhdalai, T.; Faruqui, M.; and Gopal, S. 2024. Leave no context behind: Efficient infinite context transformers with infini-attention. *arXiv preprint arXiv:2404.07143*, 101.

Pan, Y.; An, Y.; Li, Z.; Chou, Y.; Zhu, R.; Wang, X.; Wang, M.; Wang, J.; and Li, G. 2025. Scaling Linear Attention with Sparse State Expansion. *arXiv preprint arXiv:2507.16577*.

Qu, H.; Ning, L.; An, R.; Fan, W.; Derr, T.; Liu, H.; Xu, X.; and Li, Q. 2024. A survey of mamba. *arXiv preprint arXiv:2408.01129*.

Reddy, S.; Chen, D.; and Manning, C. D. 2019. Coqa: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7: 249–266.

Ren, W.; Ma, W.; Yang, H.; Wei, C.; Zhang, G.; and Chen, W. 2025. VAMBA: Understanding Hour-Long Videos with Hybrid Mamba-Transformers. *arXiv preprint arXiv:2503.11579*.

Shoeybi, M.; Patwary, M.; Puri, R.; LeGresley, P.; Casper, J.; and Catanzaro, B. 2019. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*.

Tang, Y.; Wang, Y.; Guo, J.; Tu, Z.; Han, K.; Hu, H.; and Tao, D. 2024. A survey on transformer compression. *arXiv preprint arXiv:2402.05964*.

Team, T. H.; Liu, A.; Zhou, B.; Xu, C.; Zhou, C.; Zhang, C.; Xu, C.; Wang, C.; Wu, D.; Wu, D.; et al. 2025. Hunyuan-TurboS: Advancing Large Language Models through Mamba-Transformer Synergy and Adaptive Chain-of-Thought. *arXiv preprint arXiv:2505.15431*.

Touvron, H.; Lavril, T.; Izacard, G.; Martinet, X.; Lachaux, M.-A.; Lacroix, T.; Rozière, B.; Goyal, N.; Hambro, E.; Azhar, F.; et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Waleffe, R.; Byeon, W.; Riach, D.; Norick, B.; Korthikanti, V.; Dao, T.; Gu, A.; Hatamizadeh, A.; Singh, S.; Narayanan, D.; et al. 2024. An empirical study of mamba-based language models. *arXiv preprint arXiv:2406.07887*.

Wang, J.; Paliotta, D.; May, A.; Rush, A. M.; and Dao, T. 2024. The mamba in the llama: Distilling and accelerating hybrid models. *arXiv preprint arXiv:2408.15237*.

Wang, S.; Li, B. Z.; Khabsa, M.; Fang, H.; and Ma, H. 2020. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*.

Xiao, C.; Li, M.; Zhang, Z.; Meng, D.; and Zhang, L. 2024. Spatial-Mamba: Effective Visual State Space Models via Structure-Aware State Fusion. *arXiv preprint arXiv:2410.15091*.

Yang, K.; Ackermann, J.; He, Z.; Feng, G.; Zhang, B.; Feng, Y.; Ye, Q.; He, D.; and Wang, L. 2024. Do efficient transformers really save computation? *arXiv preprint arXiv:2402.13934*.

Yuan, D.; Liu, J.; Li, B.; Zhang, H.; Wang, J.; Cai, X.; and Zhao, D. 2024. ReMamba: Equip Mamba with Effective Long-Sequence Modeling. *arXiv preprint arXiv:2408.15496*.

Zhang, H.; Zhu, Y.; Wang, D.; Zhang, L.; Chen, T.; Wang, Z.; and Ye, Z. 2024. A survey on visual mamba. *Applied Sciences*, 14(13): 5683.

Zhao, W. X.; Zhou, K.; Li, J.; Tang, T.; Wang, X.; Hou, Y.; Min, Y.; Zhang, B.; Zhang, J.; Dong, Z.; et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 1(2).

Zhou, H.; Zhang, S.; Peng, J.; Zhang, S.; Li, J.; Xiong, H.; and Zhang, W. 2021. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, 11106–11115.

Zhu, C.; Ping, W.; Xiao, C.; Shoeybi, M.; Goldstein, T.; Anandkumar, A.; and Catanzaro, B. 2021. Long-short transformer: Efficient transformers for language and vision. *Advances in neural information processing systems*, 34: 17723–17736.