How Does Sequence Modeling Architecture Influence Base Capabilities of Pre-trained Language Models? Exploring Key Architecture Design Principles to Avoid Base Capabilities Degradation

Xin Lu[1], **Yanyan Zhao**[1,*], **Si Wei**[2,*], **Shijin Wang**[2], **Bing Qin**[1], **Ting Liu**[1]
[1]Research Center for Social Computing and Interactive Robotics, Harbin Institute of Technology
[2]iFLYTEK Co., Ltd
[1]{xlu, yyzhao, qinb, tliu}@ir.hit.edu.cn
[2]{siwei, sjwang3}@iflytek.com

Abstract

Pre-trained language models represented by the Transformer have been proven to possess strong base capabilities, and the representative self-attention mechanism in the Transformer has become a classic in sequence modeling architectures. Different from the work of proposing sequence modeling architecture to improve the efficiency of attention mechanism, this work focuses on the impact of sequence modeling architectures on base capabilities. Specifically, our concern is: How exactly do sequence modeling architectures affect the base capabilities of pre-trained language models? In this work, we first point out that the mixed domain pre-training setting commonly adopted in existing architecture design works fails to adequately reveal the differences in base capabilities among various architectures. To address this, we propose a limited domain pre-training setting with out-of-distribution testing, which successfully uncovers significant differences in base capabilities among architectures at an early stage. Next, we analyze the base capabilities of stateful sequence modeling architectures, and find that they exhibit significant degradation in base capabilities compared to the Transformer. Then, through a series of architecture component analysis, we summarize a key architecture design principle: A sequence modeling architecture need possess full-sequence arbitrary selection capability to avoid degradation in base capabilities. Finally, we empirically validate this principle using an extremely simple Top-1 element selection architecture and further generalize it to a more practical Top-1 chunk selection architecture. Experimental results demonstrate our proposed sequence modeling architecture design principle and suggest that our work can serve as a valuable reference for future architecture improvements and novel designs.

1 Introduction

Recent research has discovered that pre-trained language models [35, 12, 6, 30], represented by Transformer [47], possess strong base capabilities and can achieve excellent performance in **language modeling**, **few-shot learning**, etc. Delving into the specific architecture design of Transformer, its self-attention mechanism is widely regarded as one of the key components behind its success and has since become a classic in sequence modeling architectures.

* Email corresponding.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

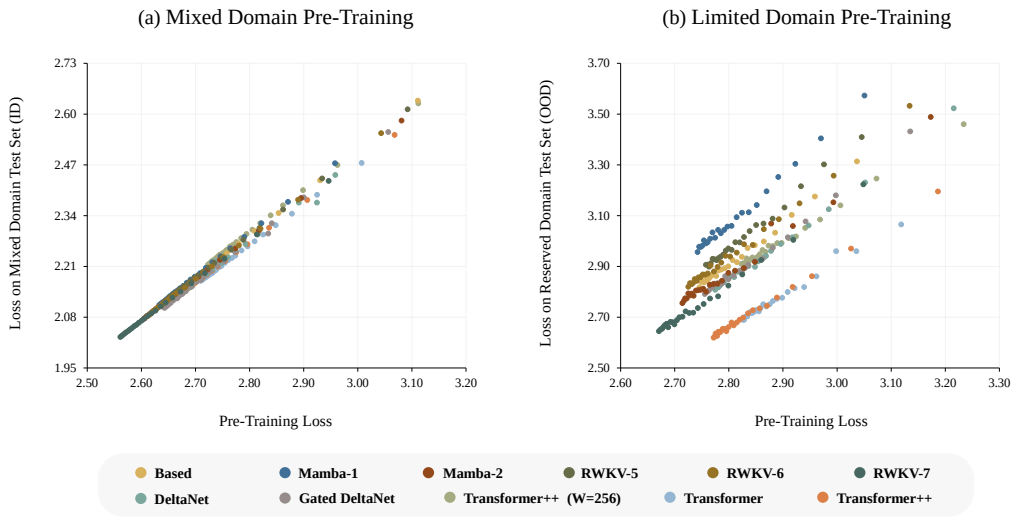


Figure 1: Language modeling test results of various sequence modeling architectures under two pre-training settings. (Model parameters≈110M, pre-trained tokens=100B and sequence length=2k)

However, due to its quadratic time complexity, the self-attention mechanism has long been plagued by high computational costs in long-sequence scenarios. To improve the efficiency of sequence modeling architectures, numerous novel stateful sequence modeling architectures have recently emerged, such as Mamba [16, 11], RWKV [31, 32, 33], Gated DeltaNet [51]. These architectures primarily inherit stateful modeling mechanisms of linear attention [23] and linear RNNs [28], offering advantages in time and space efficiency over self-attention. Additionally, they introduce new mechanisms like data-dependent decay and delta rule to enhance the expressive power of linearized modeling.

Although experimental results demonstrate that these stateful sequence modeling architectures can match or even surpass Transformer in performance while maintaining significant efficiency advantages, some studies have noted their deficiencies in specialized capabilities such as retrieval [49], copy [21], associative recall [1], and dynamic programming [50], supported by empirical validation on synthetic datasets and tasks. While these studies focus on specific issues, they are highly insightful and directly raise our new questions about sequence modeling architectures:

Are the limitations of stateful sequence modeling architectures not only confined to specialized capabilities but also present in base capabilities?

What architecture factors truly influence base capabilities?

What design principles can prevent the degradation of base capabilities?

To address these questions, we first point out that existing research on sequence modeling architectures typically adopts the same mixed domain pre-training settings used in large model development. While beneficial for practical applications, these settings are detrimental to architecture analysis, as they turn base capability tests (e.g., language modeling) into in-distribution evaluations, failing to reveal differences in base capabilities during early pre-training stages. To address this, we propose a limited domain pre-training approach, **employing out-of-distribution language modeling performance to measure base capabilities**, successfully uncovering significant differences in base capabilities among architectures at an early stage.

Next, under this setting, we analyze the base capabilities of stateful sequence modeling architectures like Mamba, RWKV, Gated DeltaNet. Our findings reveal that these architectures exhibit notable degradation in base capabilities compared to Transformer, confirming that stateful sequence modeling architectures suffer from deficiencies not only in specialized capabilities but also in base capabilities.

We then investigate the architecture factors that truly impact base capabilities. Through ablation studies on the Mamba family of architectures and analyses of common sequence modeling factors, we identify that mechanisms like data-dependent decay, convolution and position encoding only affect convergence speed rather than base capabilities. Conversely, we determine that full-sequence visibility, real relation calculation and non-uniform distribution are critical architecture factors

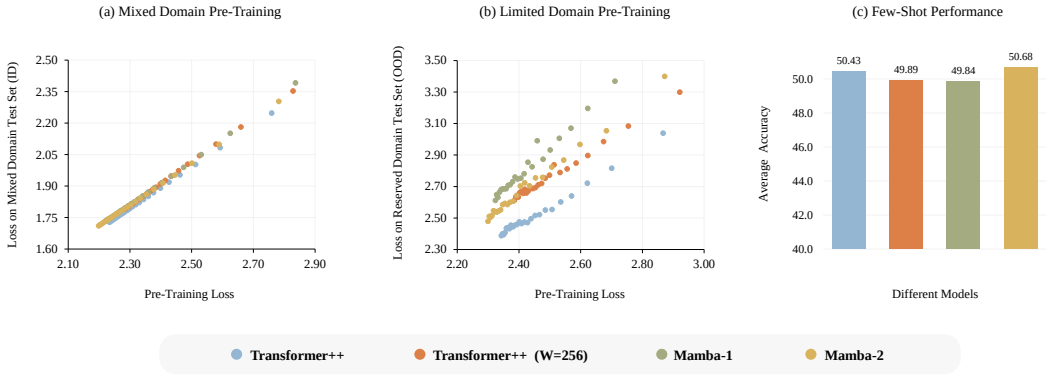


Figure 2: The Illustration include: (a) and (b) Language modeling test results of various sequence modeling architectures under two pre-training settings. (c) Few-shot learning performance results of these architectures. (Model parameters≈1.3B, pre-trained tokens=100B and sequence length=2k)

influencing base capabilities. Based on this, we summarize the principle that **"a sequence modeling architecture need to possess full-sequence arbitrary selection capability"** as the key design rule to avoid degradation in base capabilities.

Finally, we validate this principle using an extremely simple Top-1 Element Selection architecture, experimentally confirming its ability to achieve base capabilities nearly on par with the Transformer. Furthermore, we extend this validation to the more practical Top-1 Chunk Selection architecture — a direct generalization of the Top-1 Element Selection architecture, and implement GPU Kernels to ensure its time efficiency. Experiments show that the Top-1 Chunk Selection architecture, which adheres to our design principle, outperforms stateful architectures in base capabilities for both short (2k) and long (100k) sequences while maintaining competitive time efficiency.

The main contributions of our work are as follows:

- We propose a more indicative limited domain pre-training and out-of-distribution testing framework, successfully revealing the base capability degradation in stateful sequence modeling architectures during early pre-training stages. (Section 2)
- We investigate the impact of various common sequence modeling architecture factors on base capabilities and summarize the principle that "full-sequence arbitrary selection capability" is critical to avoiding degradation in base capabilities. (Section 3)
- We validate the principle using extremely simple Top-1 Element Selection architecture and demonstrate its generalization to the more practical Top-1 Chunk Selection architecture, accompanied by open-source GPU Kernels[1] to ensure time efficiency. (Section 4 and 5)
- Through our analysis and experiments, we prove the effectiveness of the proposed architecture design principle, providing valuable insights for future architecture designs.

2 Base Capabilities Evaluation

This work focuses on how sequence modeling architectures influence the base capabilities of pre-trained language models, thus necessitating the design of an evaluation framework and the implementation of assessments for different sequence modeling architectures.

2.1 Evaluation Scheme for Architecture Analysis

Existing sequence modeling architecture design works typically adopt the same **Mixed Domain Pre-Training** setting as large language model development, where corpora from as many domains as possible are collected and mixed as pre-training data. While this setting benefits practical large language model applications, it is detrimental to architecture analysis. It turns base capability tests

[1]https://github.com/luxinxyz/TSA

like language modeling into in-distribution evaluations, failing to reveal differences in architecture base capabilities during early pre-training stages. Moreover, it poorly predicts the true usability of different architectures when applied to unknown out-of-distribution domain tasks.

We tested this setting. Specifically, we pretrained models by mixing all domains (cc, c4, arxiv, book, github, stack and wiki) from the SlimPajama dataset [42] and retained a mixed-domain test set for evaluation. For models with $\approx$ 110M parameters, we pretrained Transformer [6], Transformer++ (with Rotary Embedding [43], GeGLU [41] and RMSNorm [56]), Transformer++ (Window=256), Based [2], Mamba-1 [16], Mamba-2 [11], RWKV-5 [32], RWKV-6 [32], RWKV-7 [33], DeltaNet [40] and Gated DeltaNet [51], with a sequence length of 2K and 100B tokens of pre-training. The scatter plots of pre-training loss versus mixed-domain test loss are shown in Figure 1(a). For models with $\approx$ 1.3B parameters, we pretrained Transformer++, Transformer++ (Window=256), Mamba-1 [16] and Mamba-2 [11] under similar settings, with results plotted in Figure 2(a).

From Figure 1(a) and Figure 2(a), it is evident that under the Mixed Domain Pre-Training setting, different sequence modeling architectures achieve similar test performance in language modeling when reaching comparable pre-training performance levels. Thus, this setting fails to reveal base capabilities difference among architectures during early pre-training and is unsuitable for architecture design and analysis.

To address this issue, we propose a **Limited Domain Pre-Training** with out-of-distribution (OOD) testing framework, where models are pretrained on a restricted set of domains and evaluated on unseen domains.

Since the SlimPajama dataset has already undergone deduplication across domain subsets, we easily tested this setting. Specifically, we pretrained models on the cc and c4 domains of SlimPajama and evaluated them on the arxiv, github and stack domains. Other pre-training settings remained similar, and the scatter plots of pre-training loss versus OOD test loss are shown in Figures 1(b) and 2(b).

From Figure 1(b) and Figure 2(b), significant differences emerge among sequence modeling architectures. At the same pre-training performance level, their OOD test performance varies substantially, confirming that base capabilities difference exist across architectures.

Additionally, we evaluated models with $\approx$ 1.3B parameter using the commonly adopted few-shot learning evaluation in prior work, which results in Figure 2(c). Similarly, no significant performance gaps were observed, suggesting that this evaluation method is also suboptimal for architecture design and analysis.

Based on these findings, we establish **Limited Domain Pre-Training** with OOD testing as our evaluation framework for base capabilities assessment. All subsequent tests in this work follow it.

2.2 Architecture-Induced Degradation of Base Capabilities

We further analyzed Figure 1(b) and Figure 2(b). The results show that only the standard attention-based Transformer and Transformer++ achieve optimal base capabilities—their OOD test performance is consistently the best at the same pre-training level, with no significant difference between them. In contrast, stateful sequence modeling architectures (e.g., Mamba, RWKV) exhibit varying degrees of base capabilities degradation, performing significantly worse than standard attention-based models under identical pre-training conditions.

These findings suggest that sequence modeling architectures directly induce base capabilities degradation, independent of data or other factors. The standard self-attention mechanism likely contains key architecture factors critical to base capabilities, some of which may be missing in stateful sequence modeling architectures.

3 Sequence Modeling Architecture Analysis

Based on the previous results, we know that the sequence modeling architecture can directly determine the model's base capabilities, and certain key architecture design factors are likely to play a significant role in these base capabilities. Therefore, identifying which architecture design factors are truly critical could be of great importance for subsequent architecture improvements or the design of new architectures.

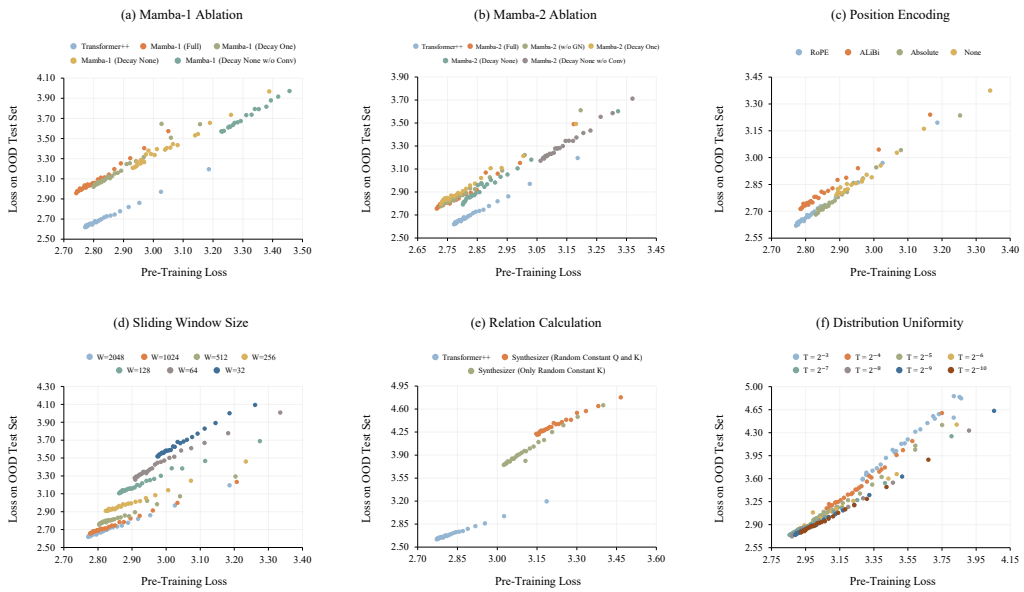


Figure 3: Analysis of the influence of various sequence modeling architecture components on base capabilities. (Model parameters≈110M, pre-trained tokens=100B or 25B and sequence length=2k)

3.1 Non-Determinative Factors of Base Capabilities

3.1.1 Mamba Ablation

Since the classical stateful sequence modeling architecture, Mamba, has been observed to exhibit degradation in base capabilities, we conducted ablation studies on some of its key components to assess their actual impact on base capabilities.

First, we focused on the **Data-Dependent Decay** in Mamba. By replacing the multi-dimensional independent data-dependent decay (Full) in Mamba-1 and Mamba-2 with either a shared data-dependent decay (Decay One) or no explicit decay (Decay None), we evaluated the effect of data-dependent decay on base capabilities, as shown in Figures 3(a) and 3(b). The results indicate that data-dependent decay only accelerates pre-training convergence but does not positively impact base capabilities. Specifically, at the same pre-training level, the out-of-distribution test performance of the ablated models did not decrease and even improved slightly.

Next, we examined the **Convolution** in Mamba. We further removed the convolution from the Decay None architecture (Decay None w/o Conv) to assess its effect on base capabilities, as shown in Figures 3(a) and 3(b). The results show that while convolution significantly speeds up pre-training convergence, it still does not contribute positively to base capabilities.

Finally, we investigated the **GroupNorm** in Mamba-2. By removing GroupNorm from the full architecture (Full $\rightarrow$ w/o GN), we evaluated its impact on base capabilities, as shown in Figure 3(b). The results confirm that GroupNorm also does not enhance base capabilities.

3.1.2 Position Encoding

Previous results have shown that Transformer and Transformer++ exhibit nearly identical base capabilities, with their primary difference in sequence modeling lying in their position encoding schemes. Therefore, we analyzed the impact of position encoding on base capabilities.

Specifically, we tested four common position encoding schemes: no position encoding, absolute position embedding, AliBi[34] and rotary position embedding [43], as illustrated in Figure 3(c). The results indicate that no position encoding, absolute position embedding and rotary position embedding yield very similar base capabilities, differing only in convergence speed. In contrast, AliBi exhibits some degradation in base capabilities. Based on these findings, we conclude that position encoding does not positively influence base capabilities.

3.2 Determinative Factors of Base Capabilities

3.2.1 Full-Sequence Visibility

The first factor we identified as critical to base capabilities is sliding window size. This observation was prompted by the significant difference in base capabilities between Transformer++ and Transformer++ (Window=256), suggesting that sliding window size may be a key factor.

To investigate, we tested various sliding window sizes, as shown in Figure 3(d). The results demonstrate that as the sliding window size increases, both pre-training convergence speed and base capabilities improve significantly. Conversely, reducing the sequence context leads to gradual degradation in base capabilities. Thus, we conclude that **Full-Sequence Visibility** is an essential requirement for sequence modeling architectures.

3.2.2 Real Relation Calculation

In previous results, we observed that under mixed domain pre-training settings, different models exhibited nearly identical test performance, creating the illusion that base capabilities is architecture-independent. This reminded us of the Synthesizer [45] series of models, where one variant replaced the true query-key computed scores with trainable random constant scores yet did not exhibit significant degradation in language modeling tasks. We hypothesized that this might also be due to mixed domain pre-training and that real relation computation between queries and keys could be a determinant of base capabilities.

To test this, we conducted experiments where we replaced keys with trainable random constants or replaced both queries and keys with trainable random constants, as shown in Figure 3(e). The results reveal that models without real relation computation suffer substantial degradation in base capabilities. Therefore, we conclude that **Real Relation Computation** is a necessary feature for sequence modeling architectures.

3.2.3 Non-Uniform Distribution

Another key factor we examined is the uniformity of attention distribution in self-attention mechanisms. We reasoned that if attention distribution were entirely uniform, the model would degenerate into a naive averaging structure over values, which exhibits poor base capabilities. However, if we start from this structure and gradually introduce non-uniformity, the model would evolve toward normal attention distribution, which has strong base capabilities. This led us to suspect that distribution uniformity directly impacts base capabilities.

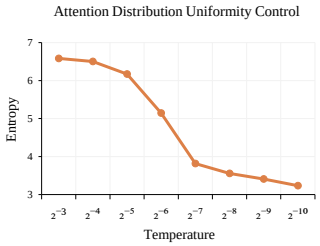


Figure 4: The relationship between attention distribution entropy and temperature.

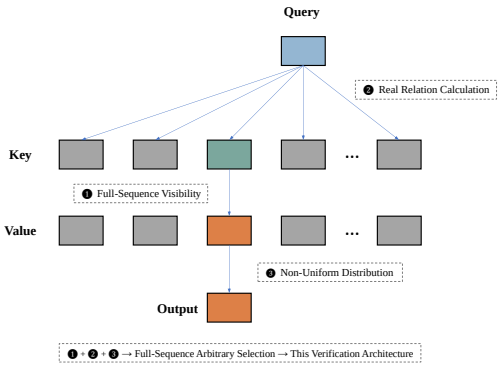
To control attention distribution uniformity, we employed two techniques: 1) Adjusting distribution uniformity via the temperature of the Softmax function. 2) Applying normalization to queries and keys to ensure consistent temperature interpretation, unaffected by the optimization of query and key transformation parameters. We tested a series of models with varying temperatures and plotted the relationship between temperature and attention distribution entropy in Figure 4. The results show that as temperature decreases, attention distribution entropy also decreases, confirming that the distribution becomes more non-uniform.

Subsequently, we evaluated the out-of-distribution performance of these models, as shown in Figure 3(f). The results indicate that as temperature decreases (i.e., distribution becomes more non-uniform), base capabilities improves. Thus, we conclude that **Non-Uniform Distribution** is a necessary feature for sequence modeling architectures.

3.3 Key Architecture Design Principle: Full-Sequence Arbitrary Selection

Integrating the three key elements derived from the preceding analysis: Full-Sequence Visibility, Real Relation Calculation and Non-Uniform Distribution, we summarize them into a unified expression:

(a) Top-1 Element Selection Architecture



(b) Top-1 Chunk Selection Kernel Design

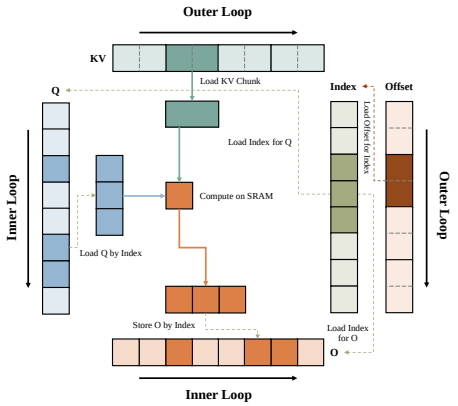


Figure 5: The Illustration include: (a) The overall architecture design of the Top-1 Element Selection architecture. (b) The kernel design for key component of the Top-1 Chunk Selection architecture.

Supporting full-sequence arbitrary selection in sequence modeling architecture is the key architecture design principle for preventing the degradation of base capabilities.

This principle directly impacts the model's base capabilities, and violating it will lead to degradation.

Note that this is an architecture design principle rather than a capability principle. The latter is often expressed as the model's retrieval capability. However, retrieval capability itself is not inherently tied to architecture design principles. For instance, stateful sequence modeling architectures like Mamba exhibit certain retrieval capabilities, yet they lack the architecture design of full-sequence arbitrary selection. It is this aspect of architecture design that our work truly focuses on.

4 Top-1 Element Selection Architecture

In previous section, we proposed a key architecture design principle for avoiding degradation in base capabilities. However, this conclusion was derived inductively and has not yet been experimentally validated. To address this, we designed an extremely minimalist **Top-1 Element Selection** architecture, which directly adheres to this design principle while maintaining strong base capabilities.

4.1 Architecture Design

We posit that "sequence modeling architectures with full-sequence arbitrary selection" is a key architecture design principle for preventing degradation in base capabilities. The simplest implementation of this principle is to directly select the element with the highest probability in the attention distribution as the output, as illustrated in Figure 5(a). We refer to this architecture as the **Top-1 Element Selection** architecture. In practice, it involves two additional operations: 1) applying normalization to queries and keys to enhance stability. 2) during training, attention scores are still computed, but the straight-through trick is introduced to reconcile top-1 selection with gradient updates.

As shown in Figure 5(a), the Top-1 Element Selection architecture is remarkably simple yet satisfies the design principle. It also incorporates three key elements: full-sequence visibility, real relation calculation and non-uniform distribution, making it an excellent candidate for validating our analysis.

4.2 Out-of-Distribution Generalization Evaluation

We pre-trained models with $\approx$ 110M and 1.3B parameters, maintaining the same pre-training settings as in previous experiments, and evaluated their OOD performance. The results, shown in Figures 6(a) and 6(b), demonstrate that the Top-1 Element Selection architecture achieves OOD performance

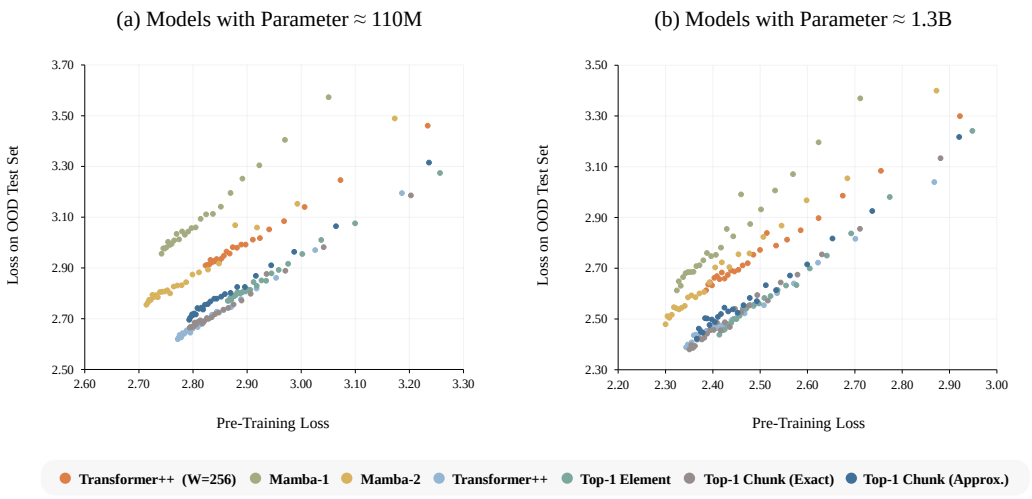


Figure 6: The out-of-distribution language modeling results of baselines and our Top-1 Element / Chunk Selection architectures. (Pre-trained tokens=100B, sequence length=2k and chunk size=128)

nearly on par with Transformer++ at both scales. This confirms that the architecture does not suffer from base capability degradation and validates the correctness of our proposed design principle.

4.3 Few-Shot Learning Evaluation

Although we previously argued that few-shot learning evaluation is not an ideal method for comparing base capabilities across architectures, it remains necessary to verify that the new architecture does not exhibit significant degradation in few-shot learning performance. Thus, we evaluated the few-shot learning performance of the 1.3B parameter model, as detailed in Table 2. The results show that the Top-1 Element Selection architecture achieves few-shot learning performance comparable to models like Transformer++, Mamba-1 and Mamba-2, indicating its strong performance in this setting.

Beyond standard tasks, we also evaluated retrieval tasks, a common benchmark in prior work. As shown in Table 1, the Top-1 Element Selection architecture significantly outperforms stateful sequence modeling architectures like Mamba-1 and Mamba-2 in retrieval tasks, further demonstrating its effectiveness.

Model	**SWDE** (acc↑)	**SQuAD** (acc↑)	**FDA** (acc↑)
Original Sequence Modeling Architecture			
Transformer++	**75.07**	**45.54**	**30.22**
Stateful Sequence Modeling Architecture			
Mamba-1	36.09	31.03	6.44
Mamba-2	37.98	35.25	17.60
Transformer++ (W=256)	12.96	41.72	1.63
Architecture Based on the Analyzed Principles			
Top-1 Element	68.23	43.77	26.95
Top-1 Chunk (Exact)	69.22	44.40	22.32
Top-1 Chunk (Approx.)	61.66	45.38	27.13

Table 1: Results of retrieval tasks. (Model parameters ≈ 1.3B, pre-trained tokens= 100B, seq len=2k and chunk size=128)

5 Top-1 Chunk Selection Architecture

In the previous section, we designed the Top-1 Element Selection architecture, successfully validating our proposed architecture design principles. However, this architecture serves as a proof-of-concept and lacks practical utility. To address this, we extended it to the **Top-1 Chunk Selection** architecture and implemented GPU kernels to optimize efficiency while maintaining strong base capabilities.

5.1 Architecture Design

5.1.1 Basic Design

The Top-1 Element Selection architecture has been verified to possess strong base capabilities but performs poorly in terms of pre-training convergence and time efficiency. To resolve this, we

Model	MMLU (acc↑)	OBQA (acc_n↑)	ARC-e (acc_n↑)	ARC-c (acc_n↑)	BoolQ (acc↑)	RACE (acc↑)	SIQA (acc↑)	SCIQ (acc_n↑)	Hella. (acc_n↑)	COPA (acc↑)	PIQA (acc_n↑)	Wino. (acc↑)	WSC (acc↑)	Avg. Score
Original Sequence Modeling Architecture														
Transformer++	25.21	33.60	48.48	25.94	**60.43**	**33.88**	39.56	**77.60**	48.42	74.00	68.99	53.91	65.57	50.43
Stateful Sequence Modeling Architecture														
Mamba-1	24.30	**34.60**	**52.82**	27.22	54.65	32.73	39.10	75.90	50.53	74.00	70.67	53.91	57.51	49.84
Mamba-2	24.74	32.80	50.04	**28.92**	56.27	33.68	40.28	75.70	**51.45**	**78.00**	**71.44**	**55.09**	60.44	50.68
Transformer++ (W=256)	25.54	34.00	48.82	26.96	59.14	**33.88**	40.02	73.70	48.19	72.00	70.89	52.49	63.00	49.89
Architecture Based on the Analyzed Principles														
Top-1 Element	25.94	32.60	47.85	26.45	60.15	31.29	**40.94**	73.40	45.05	72.00	69.15	53.75	62.27	49.30
Top-1 Chunk (Exact)	**25.96**	32.80	50.55	27.22	**60.43**	31.77	40.07	77.40	48.75	75.00	69.64	53.51	**66.30**	**50.72**
Top-1 Chunk (Approx.)	25.24	33.00	49.33	26.02	58.87	33.30	39.82	75.30	48.59	70.00	69.70	55.01	64.47	49.90

Table 2: The few-shot learning experimental results, where MMLU is 5-shot and the remaining tasks are 0-shot. (Model parameters≈1.3B, pre-trained tokens=100B, seq length=2k and chunk size=128)

generalized it to the Top-1 Chunk Selection architecture. The key difference from the Top-1 Element Selection architecture is that the query selection target shifts from fine-grained kv elements to coarse-grained kv chunks, and the attention mechanism operates only within the selected chunks. This preserves the design principles while enabling efficiency optimizations.

More specifically, the kv sequence is divided into multiple kv chunks. Each query attends to one selected full kv chunk (remote chunk) and the nearest partial kv chunk (local chunk), performing attention operations on them. Similar to the Top-1 Element Selection architecture, both queries and keys undergo normalization to enhance stability.

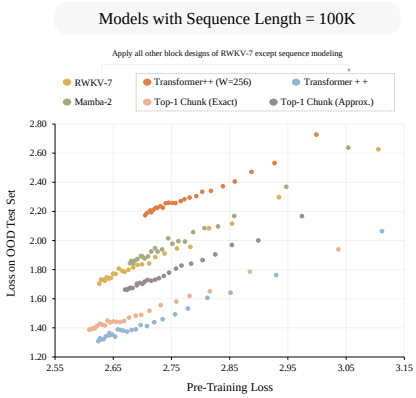


Figure 7: The OOD language modeling results of baselines and ours. (Model parameters ≈ 135M, pre-trained tokens=100B, seq length=100k and chunk size=128)

5.1.2 Exact and Approximate Variants

The selection of full kv chunks can be implemented in two ways, corresponding to two variants:

Top-1 Chunk (Exact): The query computes the full attention distribution with all keys, obtaining the probability for each kv chunk. The chunk is selected based on these exact probabilities.

Top-1 Chunk (Approx.): The exponential function in probability computation can be approximated by a first-order linear function. Under this approximation, the selection process simplifies to computing the mean vector of each key chunk and performing a dot product between the query and the mean vector to select the chunk. This significantly reduces computational overhead.

5.1.3 GPU Kernels Design

To achieve better time efficiency, we designed and implemented GPU kernels for the Top-1 Chunk Selection architecture. An overview is given in this section, with details in Appendix D .

For exact chunk selection, approximate chunk selection and local chunk attention, we directly implemented Triton kernels using simple, naive designs that already deliver good performance.

For remote chunk attention, we conducted specialized optimizations when implementing Triton Kernels, shown in Figure 5(b). Specifically, parallel processing is performed along the kv dimension in complete chunks, meaning different kv chunks are processed independently in parallel. This partitioning is feasible because each query selects only one remote chunk, ensuring no overlap between queries associated with different kv chunks. Within each kv chunk, we load index via offset, then load the queries relevant to the current kv chunk by the index and compute attention in SRAM, storing results by the index. Efficient offset and index handling is achieved by custom CUDA kernels.

During our research, we observed that DeepSeek and Kimi released two sparse attention works — NSA [53] and MoBA [27] in February 2025. Their sequence modeling approaches closely resemble our Top-1 Chunk (Approx.). **Although we discovered this architecture independently, we no longer claim Top-1 Chunk (Approx.) as our primary contribution.** Instead, our key contributions include Top-1 Element, Top-1 Chunk (Exact), analytical experiments and the released GPU kernels.

5.2 Out-of-Distribution Generalization and Few-Shot Learning Evaluation

We pre-trained models with $\approx$ 110M and 1.3B parameters (chunk size = 128), as shown in Figures 6(a) and 6(b). Results show both Top-1 Chunk Selection variants achieve OOD performance close to Transformer++ and outperform Mamba-1/2. We also evaluated few-shot learning tasks, as detailed in Tables 2 and 1. Top-1 Chunk Selection exhibits results and conclusions similar to Top-1 Element Selection: strong performance on general tasks without degradation and advantages in retrieval tasks.

5.3 Long Sequence and Architecture Combination

In this section, we evaluate two aspects: first, base capabilities, time efficiency and long-sequence retrieval capability of Top-1 Chunk Selection architecture on long sequences (100k); second, performance of Top-1 Chunk Selection architecture when transferred to other models. We designed an integrated setting where we ported Transformer++, Transformer++ (W=256), Top-1 Chunk (Exact) and Top-1 Chunk (Approx.) to the RWKV-7 architecture, replacing only the sequence modeling while retaining the rest of RWKV-7. Pre-training data remained unchanged, and OOD test data were filtered to retain samples from arxiv and github with original lengths exceeding 100k. Since stack contained almost no data with original lengths over 100k, we supplemented the test set with samples from AutoMathText [57] met this criterion.

The base capabilities results are shown in Figure 7. Transformer++ demonstrated the strongest base capabilities, followed closely by Top-1 Chunk (Exact). Top-1 Chunk (Approx.) outperformed other stateful architectures in base capabilities, but due to its approximate nature, it exhibited significant pretraining convergence degradation and base capabilities degradation compared to Transformer++ (similar issues may also affect DeepSeek NSA and Kimi MoBA, which employ analogous approaches).

Model	Pre-Train Speed	Inference Speed	S-NIAH (Passkey Retrieval)				
			16k	32k	64k	96k	100k
Original Architecture (Unmodified)							
Mamba-2	5.57×	1.01×	5.40	1.40	0.40	0.20	0.80
RWKV-7	2.47×	0.73×	70.80	14.80	1.20	0.00	0.20
Apply all other block designs of RWKV-7 except sequence modeling							
Transformer++	1.00×	1.00×	76.20	**49.80**	**25.60**	16.20	**15.80**
Transformer++ (W=256)	5.29×	1.72×	1.60	0.40	0.20	0.20	0.40
Top-1 Chunk (Exact)	1.90×	1.19×	76.80	**49.80**	22.00	15.00	11.80
Top-1 Chunk (Approx.)	4.50×	1.21×	**78.80**	48.60	21.40	**21.80**	15.00

Table 3: Results of time efficiency and long sequence retrieval. (Model parameter≈135M, pre-trained tokens=100B, seq length=100k and chunk size=128)

Time efficiency and long-sequence retrieval results are presented in Table 3. Top-1 Chunk (Approx.) achieved substantial speed improvements over Transformer++, reaching time efficiency levels comparable to Mamba-2. Top-1 Chunk (Exact) also showed speed gains, achieving time efficiency similar to RWKV-7. In long-sequence retrieval, results on S-NIAH [20] shown Top-1 Chunk Selection delivered performance close to Transformer++, while other stateful models suffered severe degradation.

6 Limitations

This work primarily focuses on models pre-trained with language modeling objectives, without exploring other pre-training objectives. As a result, our conclusions are limited to language models, narrowing the current applicability of our findings. We plan to expand this research with additional experiments in the future.

7 Conclusion

This work investigates how sequence modeling architectures influence base capabilities of pre-trained language models. We first reveal the degradation of base capabilities in stateful sequence modeling architectures by limited domain pre-training. Subsequently, via architecture analysis, we identify "full-sequence arbitrary selection" as the key architecture design principle for preventing such degradation. Finally, we validate our analysis by proposing Top-1 Element Selection and Top-1 Chunk Selection architecture, which may provide valuable references and foundations for future research.

Acknowledgments

This work was supported by the New Generation Artificial Intelligence-National Science and Technology Major Project 2023ZD0121100, the National Natural Science Foundation of China (NSFC) via grant 62441614 and 62176078.

References

[1] S. Arora, S. Eyuboglu, A. Timalsina, I. Johnson, M. Poli, J. Zou, A. Rudra, and C. Re. Zoology: Measuring and improving recall in efficient language models. In *The Twelfth International Conference on Learning Representations*, 2024.

[2] S. Arora, S. Eyuboglu, M. Zhang, A. Timalsina, S. Alberti, J. Zou, A. Rudra, and C. Re. Simple linear attention language models balance the recall-throughput tradeoff. In *ICLR 2024 Workshop on Mathematical and Empirical Understanding of Foundation Models*, 2024.

[3] S. Arora, B. Yang, S. Eyuboglu, A. Narayan, A. Hojel, I. Trummer, and C. Ré. Language models enable simple systems for generating structured views of heterogeneous data lakes, 2025.

[4] I. Beltagy, M. E. Peters, and A. Cohan. Longformer: The long-document transformer, 2020.

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

[6] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

[7] R. Child, S. Gray, A. Radford, and I. Sutskever. Generating long sequences with sparse transformers, 2019.

[8] C. Clark, K. Lee, M.-W. Chang, T. Kwiatkowski, M. Collins, and K. Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[9] P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, and O. Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *ArXiv*, abs/1803.05457, 2018.

[10] T. Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. 2023.

[11] T. Dao and A. Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In R. Salakhutdinov, Z. Kolter, K. Heller, A. Weller, N. Oliver, J. Scarlett, and F. Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 10041–10071. PMLR, 21–27 Jul 2024.

[12] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[13] J. Ding, S. Ma, L. Dong, X. Zhang, S. Huang, W. Wang, N. Zheng, and F. Wei. Longnet: Scaling transformers to 1,000,000,000 tokens, 2023.

[14] J. Dong, B. Feng, D. Guessous, Y. Liang, and H. He. Flex attention: A programming model for generating optimized attention kernels, 2024.

[15] L. Gao, J. Tow, B. Abbasi, S. Biderman, S. Black, A. DiPofi, C. Foster, L. Golding, J. Hsu, A. Le Noac'h, H. Li, K. McDonell, N. Muennighoff, C. Ociepa, J. Phang, L. Reynolds, H. Schoelkopf, A. Skowron, L. Sutawika, E. Tang, A. Thite, B. Wang, K. Wang, and A. Zou. The language model evaluation harness, 07 2024.

[16] A. Gu and T. Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling*, 2024.

[17] A. Gu, K. Goel, and C. Re. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*.

[18] J. He, J. Qiu, A. Zeng, Z. Yang, J. Zhai, and J. Tang. Fastmoe: A fast mixture-of-expert training system. *arXiv preprint arXiv:2103.13262*, 2021.

[19] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt. Measuring massive multitask language understanding, 2021.

[20] C.-P. Hsieh, S. Sun, S. Kriman, S. Acharya, D. Rekesh, F. Jia, and B. Ginsburg. RULER: What's the real context size of your long-context language models? In *First Conference on Language Modeling*, 2024.

[21] S. Jelassi, D. Brandfonbrener, S. M. Kakade, and E. Malach. Repeat after me: Transformers are better than state space models at copying. In R. Salakhutdinov, Z. Kolter, K. Heller, A. Weller, N. Oliver, J. Scarlett, and F. Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 21502–21521. PMLR, 21–27 Jul 2024.

[22] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. de las Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, L. R. Lavaud, M.-A. Lachaux, P. Stock, T. L. Scao, T. Lavril, T. Wang, T. Lacroix, and W. E. Sayed. Mistral 7b, 2023.

[23] A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In H. D. III and A. Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5156–5165. PMLR, 13–18 Jul 2020.

[24] G. Lai, Q. Xie, H. Liu, Y. Yang, and E. Hovy. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794, Copenhagen, Denmark, Sept. 2017. Association for Computational Linguistics.

[25] H. Levesque, E. Davis, and L. Morgenstern. The winograd schema challenge. In *Thirteenth international conference on the principles of knowledge representation and reasoning*, 2012.

[26] C. Lockard, P. Shiralkar, and X. L. Dong. OpenCeres: When open information extraction meets the semi-structured web. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3047–3056, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[27] E. Lu, Z. Jiang, J. Liu, Y. Du, T. Jiang, C. Hong, S. Liu, W. He, E. Yuan, Y. Wang, Z. Huang, H. Yuan, S. Xu, X. Xu, G. Lai, Y. Chen, H. Zheng, J. Yan, J. Su, Y. Wu, N. Y. Zhang, Z. Yang, X. Zhou, M. Zhang, and J. Qiu. Moba: Mixture of block attention for long-context llms, 2025.

[28] E. Martin and C. Cundy. Parallelizing linear recurrent neural nets over sequence length. In *International Conference on Learning Representations*, 2018.

[29] T. Mihaylov, P. Clark, T. Khot, and A. Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391, Brussels, Belgium, Oct.-Nov. 2018. Association for Computational Linguistics.

[30] OpenAI. Gpt-4 technical report, 2023.

[31] B. Peng, E. Alcaide, Q. Anthony, A. Albalak, S. Arcadinho, S. Biderman, H. Cao, X. Cheng, M. Chung, L. Derczynski, X. Du, M. Grella, K. Gv, X. He, H. Hou, P. Kazienko, J. Kocon, J. Kong, B. Koptyra, H. Lau, J. Lin, K. S. I. Mantri, F. Mom, A. Saito, G. Song, X. Tang, J. Wind, S. Woźniak, Z. Zhang, Q. Zhou, J. Zhu, and R.-J. Zhu. RWKV: Reinventing RNNs for the transformer era. In H. Bouamor, J. Pino, and K. Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 14048–14077, Singapore, Dec. 2023. Association for Computational Linguistics.

[32] B. Peng, D. Goldstein, Q. Anthony, A. Albalak, E. Alcaide, S. Biderman, E. Cheah, X. Du, T. Ferdinan, H. Hou, P. Kazienko, K. K. GV, J. Kocoń, B. Koptyra, S. Krishna, R. M. Jr., J. Lin, N. Muennighoff, F. Obeid, A. Saito, G. Song, H. Tu, C. Wirawan, S. Woźniak, R. Zhang, B. Zhao, Q. Zhao, P. Zhou, J. Zhu, and R.-J. Zhu. Eagle and finch: Rwkv with matrix-valued states and dynamic recurrence, 2024.

[33] B. Peng, R. Zhang, D. Goldstein, E. Alcaide, X. Du, H. Hou, J. Lin, J. Liu, J. Lu, W. Merrill, G. Song, K. Tan, S. Utpala, N. Wilce, J. S. Wind, T. Wu, D. Wuttke, and C. Zhou-Zheng. Rwkv-7 "goose" with expressive dynamic state evolution, 2025.

[34] O. Press, N. Smith, and M. Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. In *International Conference on Learning Representations*, 2022.

[35] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[36] P. Rajpurkar, R. Jia, and P. Liang. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[37] M. Roemmele, C. A. Bejan, and A. S. Gordon. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *2011 AAAI Spring Symposium Series*, 2011.

[38] K. Sakaguchi, R. L. Bras, C. Bhagavatula, and Y. Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

[39] M. Sap, H. Rashkin, D. Chen, R. Le Bras, and Y. Choi. Social IQa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4463–4473, Hong Kong, China, Nov. 2019. Association for Computational Linguistics.

[40] I. Schlag, K. Irie, and J. Schmidhuber. Linear transformers are secretly fast weight programmers. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 9355–9366. PMLR, 18–24 Jul 2021.

[41] N. Shazeer. Glu variants improve transformer, 2020.

[42] Z. Shen, T. Tao, L. Ma, W. Neiswanger, Z. Liu, H. Wang, B. Tan, J. Hestness, N. Vassilieva, D. Soboleva, and E. Xing. Slimpajama-dc: Understanding data combinations for llm training, 2024.

[43] J. Su, Y. Lu, S. Pan, A. Murtadha, B. Wen, and Y. Liu. Roformer: Enhanced transformer with rotary position embedding, 2023.

[44] Y. Sun, L. Dong, S. Huang, S. Ma, Y. Xia, J. Xue, J. Wang, and F. Wei. Retentive network: A successor to transformer for large language models, 2023.

[45] Y. Tay, D. Bahri, D. Metzler, D.-C. Juan, Z. Zhao, and C. Zheng. Synthesizer: Rethinking self-attention for transformer models. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 10183–10192. PMLR, 18–24 Jul 2021.

[46] P. Tillet, H. T. Kung, and D. 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.

[47] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[48] J. Welbl, N. F. Liu, and M. Gardner. Crowdsourcing multiple choice science questions, 2017.

[49] K. Wen, X. Dang, and K. Lyu. RNNs are not transformers (yet): The key bottleneck on in-context retrieval. In *The Thirteenth International Conference on Learning Representations*, 2025.

[50] K. Yang, J. Ackermann, Z. He, G. Feng, B. Zhang, Y. Feng, Q. Ye, D. He, and L. Wang. Do efficient transformers really save computation? In R. Salakhutdinov, Z. Kolter, K. Heller, A. Weller, N. Oliver, J. Scarlett, and F. Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 55928–55947. PMLR, 21–27 Jul 2024.

[51] S. Yang, J. Kautz, and A. Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. In *The Thirteenth International Conference on Learning Representations*, 2025.

[52] S. Yang and Y. Zhang. Fla: A triton-based library for hardware-efficient implementations of linear attention mechanism, Jan. 2024.

[53] J. Yuan, H. Gao, D. Dai, J. Luo, L. Zhao, Z. Zhang, Z. Xie, Y. X. Wei, L. Wang, Z. Xiao, Y. Wang, C. Ruan, M. Zhang, W. Liang, and W. Zeng. Native sparse attention: Hardware-aligned and natively trainable sparse attention, 2025.

[54] M. Zaheer, G. Guruganesh, K. A. Dubey, J. Ainslie, C. Alberti, S. Ontanon, P. Pham, A. Ravula, Q. Wang, L. Yang, and A. Ahmed. Big bird: Transformers for longer sequences. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 17283–17297. Curran Associates, Inc., 2020.

[55] R. Zellers, A. Holtzman, Y. Bisk, A. Farhadi, and Y. Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy, July 2019. Association for Computational Linguistics.

[56] B. Zhang and R. Sennrich. Root Mean Square Layer Normalization. In *Advances in Neural Information Processing Systems 32*, Vancouver, Canada, 2019.

[57] Y. Zhang, Y. Luo, Y. Yuan, and A. C.-C. Yao. Autonomous data selection with zero-shot generative classifiers for mathematical texts, 2025.

A Model Specifications and Pre-training Procedures

This work implemented and pre-trained multiple models with different architectures under various configurations to support experiments in different sections.

The first configuration involves models with approximately 110M (small scale) and 1.3B (large scale, available only for certain architectures) parameters, with a sequence length of 2k. It includes the following architectures: Transformer [6], Transformer++, Transformer++ (W=256), Based [2], Mamba-1 [16], Mamba-2 [11], RWKV-5 [32], RWKV-6 [32], RWKV-7 [33], DeltaNet [40], Gated DeltaNet [51], Top-1 Element, Top-1 Chunk (Exact) and Top-1 Chunk (Approx.).

Transformer adopts the GPT [35] architecture, is based on FlashAttention-2[2] [10], and has a hidden dimension of 768, 12 layers and 12 attention heads for the small-scale version; while the large-scale version has a hidden dimension of 2048, 24 layers and 16 attention heads. Transformer++ introduces Rotary Embedding [43], GeGLU [41] and RMSNorm [56] over the Transformer, also built using FlashAttention-2, with identical model specifications as the Transformer. Transformer++ (W=256) incorporates a window size of 256, retains the same model specification, and is also based on FlashAttention-2.

Based [2] is implemented according to the official open-source code[3], with a hidden dimension of 768 and 15 layers for the small-scale version; among them, there are 3 layers of window attention with 12 attention heads and a window size of 128, and 3 layers of TaylorExp linear attention with 12 attention heads. Mamba-1 [16] is implemented based on the official open-source code[4], with a hidden dimension of 768, 24 layers and a state size of 16 for the small-scale version; the large-scale version has a hidden dimension of 2048, 48 layers and the same state size of 16. Mamba-2 [11] is implemented based on the official open-source code (URL is the same as Mamba-1), with a hidden dimension of 768, 24 layers and a state size of 128 for the small-scale version; the large-scale version has a hidden dimension of 2048, 48 layers and the same state size of 128. RWKV-5/6/7 [32, 33] are implemented based on the official open-source code[5], with a hidden dimension of 768, 12 layers and 12 attention heads for the small-scale versions. DeltaNet [40] is based on Flash Linear Attention[6] [52], with a hidden dimension of 768, 12 layers and 8 attention heads for the small-scale version. Gated DeltaNet [51] is also based on Flash Linear Attention, with a hidden dimension of 768, 12 layers, 6 attention heads and a head dimension of 96 for the small-scale version.

Top-1 Element is implemented using PyTorch[7] and the transformers[8] library, matching the model specifications of the Transformer++. Top-1 Chunk (Exact/Approx.) shares the same model specifications as Transformer++, with a chunk size of 128. Its sequence modeling part is implemented using Triton[9] [46], and its offset and index CUDA kernel is modified from the fastmoe[10] [18].

In addition, all of the above models use the Mixtral [22] vocabulary, which contains 32,000 tokens. Large-scale models involved in the training process were trained using the DeepSpeed[11] open-source project. All models involving Rotary Embedding have a base of 10,000.

The second configuration involves models with approximately 135M parameters (small scale) and a sequence length of 100k. This mainly consists of architectures obtained by replacing the sequence modeling in RWKV-7. Specific models include: Transformer++, Transformer++ (W=256), Top-1 Chunk (Exact) and Top-1 Chunk (Approx.).

Transformer++ is based on FlashAttention-2, while Transformer++ (W=256) uses FlexAttention[12] [14]. The implementations of Top-1 Chunk (Exact) and Top-1 Chunk (Approx.) follow those described in the previous configuration. All four models have a hidden dimension of 768, 12

[2] https://github.com/Dao-AILab/flash-attention

[3] https://github.com/HazyResearch/based

[4] https://github.com/state-spaces/mamba

[5] https://github.com/BlinkDL/RWKV-LM

[6] https://github.com/fla-org/flash-linear-attention

[7] https://pytorch.org/

[8] https://github.com/huggingface/transformers

[9] https://github.com/triton-lang/triton

[10] https://github.com/laekov/fastmoe

[11] https://github.com/deepspeedai/DeepSpeed

[12] https://pytorch.org/blog/flexattention/

layers and 12 attention heads, with a chunk size of 128 for the two Top-1 Chunk models. Additionally, RWKV-7 was implemented using Flash Linear Attention when the sequence length was set to 100k to improve time efficiency. Gradient checkpointing was applied to reduce GPU memory consumption across all models. All models use the Mixtral vocabulary with a size of 32,000 tokens. Models incorporating Rotary Embedding maintain a base of 1,000,000.

All models were pre-trained from scratch on the SlimPajama [42] dataset using language modeling objective across 100B tokens (mixed domain or limited domain). These settings apply to most of the main experiments and analysis experiments in this work (with the exception being the "distribution uniformity" analysis experiment, where some models were pre-trained on 25B tokens). More detailed test data partitions and other related settings are introduced in Section 2.1 and Section 5.3. Pre-trained models were trained using bfloat16 precision and the Adam optimizer. Small-scale models used a learning rate of 2e-4, while large-scale models used a learning rate of 5e-5. The optimizer's β_1 and β_2 values were set to 0.9 and 0.95, respectively. Learning rates for small-scale models were warmed up over the first 2,500 steps, while those for large-scale models were warmed up over the first 5,000 steps, followed by linear decay. The batch sizes for small-scale models were 256 (sequence length=2k) and 8 (sequence length=100k), while that for large-scale models was 128 (sequence length=2k). All models were trained in a distributed manner across 8 Nvidia Tesla A100 GPUs, totaling approximately 1,221 days of single-GPU equivalent training time.

B Evaluation Procedures

For language modeling evaluation, since all models in this work were pre-trained from scratch using the same vocabulary, we directly use the loss on the test set for evaluation.

For few-shot learning evaluation, general tasks include MMLU [19], OpenBookQA [29], ARC Easy & Challenge [9], BoolQ [8], RACE [24], SIQA [39], SCIQ [48], HellaSwag [55], COPA [37], PIQA [5], WinoGrande [38] and Winograd [25]; retrieval tasks include SWDE [26], SQuAD [36] and FDA [3]. Among these, MMLU uses a 5-shot setting, while all others are evaluated in a 0-shot setting. To ensure the stability and reproducibility of the evaluation results, we use the lm-evaluation-harness[13] [15] open-source evaluation framework for evaluation.

For time efficiency evaluation (Table 3), we compare the pre-training speed of different models with an input length of 100k tokens and a total batch size of 8. For inference speed comparison, we use an input length of 50k tokens and generate 5k tokens, with a total batch size of 64.

For long-sequence retrieval evaluation (Table 3), to ensure the stability and reproducibility of the results, we use the RULER[14] [20] open-source evaluation framework for evaluation.

C Related Work

Our work is related to research on multiple topics, primarily including stateful sequence modeling architectures, sparse attention architectures and the analysis of sequence modeling architectures.

Stateful sequence modeling architectures have gradually evolved to address the inefficiency of standard self-attention [47] in handling long sequences. The core of these architectures lies in linear RNNs [28] and linear attention [23], with key characteristics being their ability to support both parallel and efficient training while also being expressible in the form of iterative state updates. Additionally, they offer certain advantages in terms of time and space efficiency compared to standard self-attention. Subsequent works have continuously introduced new structures or mechanisms to enhance the expressive power of stateful sequence modeling architectures. Early works incorporated data-independent decays, such as S4 [17], RetNet [44], RWKV-5 [32], among others. Later works added data-dependent decays, including Mamba-1 [16], Mamba-2 [11], and RWKV-6 [32]. The most recent developments have introduced the delta rule, exemplified by Gated DeltaNet [51] and RWKV-7 [33]. Our work does not involve the development of new stateful sequence modeling architectures; instead, it primarily reveals existing deficiencies in base capabilities within established stateful sequence modeling architectures.

[13] https://github.com/EleutherAI/lm-evaluation-harness

[14] https://github.com/NVIDIA/RULER

Sparse attention architectures represent another approach to addressing the efficiency issues of standard self-attention. These works mainly achieve improved computational efficiency by reducing the number of terms involved in attention calculations. Early efforts were dominated by manually designed fixed sparsity patterns, including Sparse Transformer [7], Longformer [4], BigBird [54], LongNet [13], and others. Recent works, such as NSA [53] and MoBA [27], introduce dynamic block selection-based sparsification schemes that effectively balance computational efficiency and practical performance. The core of our work is to reveal how sequence modeling architectures influence base capabilities. Although the improved architecture we ultimately propose also adopts a dynamic block selection-based sparsification scheme, unlike previous works that focus on heuristic designs from an efficiency perspective and only evaluate in-domain performance, we begin our analysis from base capabilities, gradually building toward a well-founded proposal for dynamic block selection. We further discover unique advantages of such architectures in out-of-distribution performance across domains and find that architectures like NSA and MoBA, which employ approximate selection strategies, may lead to limitations in out-of-distribution generalization capability. In addition, as mentioned in Section 5, although we discovered this architecture independently, we no longer claim Top-1 Chunk (Approx.) as our primary contribution. Instead, our key contributions include Top-1 Element, Top-1 Chunk (Exact), analytical experiments and the released GPU kernels.

Works focusing on the analysis of sequence modeling architectures mainly concentrate on revealing specific capability deficits in stateful sequence modeling architectures, particularly in tasks such as retrieval [49], copy [21], associative recall [1], and dynamic programming [50]. These studies have successfully conducted experimental validations on synthetic datasets and synthetic tasks. Additionally, some of these works analyze theoretical differences in representational capabilities between RNNs and Transformers. While our work also focuses on potential defects in stateful sequence modeling architectures, we primarily investigate deficiencies in base capabilities rather than specific ones — for example, whether such deficiencies can already be manifested in language modeling tasks. Moreover, our work does not delve into comparing strengths and weaknesses in model representation or retrieval capabilities but instead pays more attention to the architecture design itself, aiming to uncover truly effective components through changes in base capabilities and offering insights into architecture design.

D Algorithm Details

To achieve better time efficiency, we designed and implemented GPU kernels for the Top-1 Chunk Selection architecture.

Specifically, we designed Triton kernels for Top-1 Chunk Selection architecture, with its forward pass detailed in Algorithm 1 and its backward pass outlined in Algorithm 2 . Beyond these two core algorithms, the implementation involves several other components. In both algorithms, $\mathbf{F}$ and $\mathbf{I}$ represent the offset and index, respectively. The offset $\mathbf{F}$ sequentially records the number of times each chunk is selected by querys, while the index $\mathbf{I}$ sequentially records the indices of all queries associated with each chunk. These can be obtained through a naive PyTorch implementation; we developed hardware-efficient CUDA kernels based on the logic of the naive implementation. Additionally, the chunk indices for both exact selection and approximate selection can also be acquired via a naive PyTorch implementation. Similarly, we developed hardware-efficient Triton kernels following the logic of the naive implementation. Additionally, we designed Triton kernels for the decoding stage, which can effectively accelerate the decoding process. The source code has been released at: `https://github.com/luxinxyz/TSA` .

Algorithm 1 Top-1 Chunk Selection Forward Pass

Require: $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{T \times d}$, $\mathbf{F} \in \mathbb{R}^{\lceil \frac{T}{B_c} \rceil}$, $\mathbf{I} \in \mathbb{R}^{T - B_c}$, chunk size B_c, block size B_r.

Divide $\mathbf{K}, \mathbf{V}$ into $T_c = \left\lceil \frac{T}{B_c} \right\rceil$ blocks $\mathbf{K}_1, \ldots, \mathbf{K}_{T_c}$ and $\mathbf{V}_1, \ldots, \mathbf{V}_{T_c}$ of size $B_c \times d$ each, divide $\mathbf{F}$ into $T_c = \left\lceil \frac{T}{B_c} \right\rceil$ scalars $f_1, \ldots, f_{T_c}$.
Initialize $\mathbf{O} = (0)_{T \times d} \in \mathbb{R}^{T \times d}, \mathbf{M} = (-\infty)_T \in \mathbb{R}^T, \mathbf{L} = (0)_T \in \mathbb{R}^T, \mathbf{R} = (0)_T \in \mathbb{R}^T$.
for $1 \leq j \leq T_c$ **do**
  Load $\mathbf{K}_j, \mathbf{V}_j, f_j$ from HBM to on-chip SRAM.
  if $j = 1$ **then**
    On chip, Initialize $f_{j-1} = 0$.
  else
    Load f_{j-1} from HBM to on-chip SRAM.
  end if
  On chip, compute $M = f_j - f_{j-1}$.
  Divide $\mathbf{I}_{[f_{j-1}, \ldots, f_j]}$ into $M_r = \left\lceil \frac{M}{B_r} \right\rceil$ blocks $\mathbf{I}_1, \ldots, \mathbf{I}_{M_r}$ of size B_r each.
  for $1 \leq i \leq M_r$ **do**
    Load $\mathbf{I}_i$ from HBM to on-chip SRAM.
    Load $\mathbf{Q}_i \in \mathbb{R}^{B_r \times d}$ by index $\mathbf{I}_i \in \mathbb{R}^{B_r}$ from $\mathbf{Q}$ in HBM to on-chip SRAM.
    On chip, compute $\mathbf{S}_{ij} = \mathbf{Q}_i \mathbf{K}_j^T \in \mathbb{R}^{B_r \times B_c}$.
    On chip, compute $\tilde{m}_{ij} = \mathrm{rowmax}(\mathbf{S}_{ij}) \in \mathbb{R}^{B_r}$.
    On chip, compute $\tilde{\mathbf{P}}_{ij} = \exp(\mathbf{S}_{ij} - \tilde{m}_{ij}) \in \mathbb{R}^{B_r \times B_c}$.
    On chip, compute $\tilde{\ell}_{ij} = \mathrm{rowsum}(\tilde{\mathbf{P}}_{ij}) \in \mathbb{R}^{B_r}$.
    On chip, compute $\mathbf{O}_i = \tilde{\mathbf{P}}_{ij} \mathbf{V}_j \in \mathbb{R}^{B_r \times d}$.
    Write $\mathbf{O}_i \in \mathbb{R}^{B_r \times d}$ by index $\mathbf{I}_i \in \mathbb{R}^{B_r}$ to $\mathbf{O}$ in HBM.
    Write $\tilde{m}_{ij} \in \mathbb{R}^{B_r}$ by index $\mathbf{I}_i \in \mathbb{R}^{B_r}$ to $\mathbf{M}$ in HBM.
    Write $\tilde{\ell}_{ij} \in \mathbb{R}^{B_r}$ by index $\mathbf{I}_i \in \mathbb{R}^{B_r}$ to $\mathbf{L}$ in HBM.
  end for
end for
Divide $\mathbf{Q}, \mathbf{K}, \mathbf{V}, \mathbf{O}$ into $T_c = \left\lceil \frac{T}{B_c} \right\rceil$ blocks $\mathbf{Q}_1, \ldots, \mathbf{Q}_{T_c}$, $\mathbf{K}_1, \ldots, \mathbf{K}_{T_c}$, $\mathbf{V}_1, \ldots, \mathbf{V}_{T_c}$ and $\mathbf{O}_1, \ldots, \mathbf{O}_{T_c}$ of size $B_c \times d$ each.
Divide $\mathbf{M}, \mathbf{L}$ into $T_c = \left\lceil \frac{T}{B_c} \right\rceil$ blocks $m_1, \ldots, m_{T_c}$ and $\ell_1, \ldots, \ell_{T_c}$ of size B_c each.
for $1 \leq i \leq T_c$ **do**
  Load $\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i, \mathbf{O}_i, m_i, \ell_i$ from HBM to on-chip SRAM.
  On chip, compute $\mathbf{S}_i = \mathbf{Q}_i \mathbf{K}_i^T \in \mathbb{R}^{B_c \times B_c}$.
  On chip, compute $m_i^{(T)} = max(m_i, \mathrm{rowmax}(\mathbf{S}_i)) \in \mathbb{R}^{B_c}$.
  On chip, compute $\tilde{\mathbf{P}}_i = \exp(\mathbf{S}_i - m_i^{(T)}) \in \mathbb{R}^{B_c \times B_c}$.
  On chip, compute $\ell_i^{(T)} = e^{m_i - m_i^{(T)}} \ell_i + \mathrm{rowsum}(\tilde{\mathbf{P}}_i) \in \mathbb{R}^{B_c}$.
  On chip, compute $\mathbf{O}_i^{(T)} = \mathrm{diag}(\ell_i^{(T)})^{-1} (\mathrm{diag}(e^{m_i - m_i^{(T)}}) \mathbf{O}_i + \tilde{\mathbf{P}}_i \mathbf{V}_i) \in \mathbb{R}^{B_c \times d}$.
  On chip, compute $r_i^{(T)} = m_i^{(T)} + log(\ell_i^{(T)}) \in \mathbb{R}^{B_c}$.
  Write $\mathbf{O}_i^{(T)} \in \mathbb{R}^{B_c \times d}$ to $\mathbf{O}$ in HBM.
  Write $r_i^{(T)} \in \mathbb{R}^{B_c}$ to $\mathbf{R}$ in HBM.
end for
Return $\mathbf{O}, \mathbf{R}$.

Algorithm 2 Top-1 Chunk Selection Backward Pass

Require: $\mathbf{Q}, \mathbf{K}, \mathbf{V}, \mathbf{O}, \mathbf{dO} \in \mathbb{R}^{T \times d}$, $\mathbf{R} \in \mathbb{R}^{T}$, $\mathbf{F} \in \mathbb{R}^{\lceil \frac{T}{B_c} \rceil}$, $\mathbf{I} \in \mathbb{R}^{T - B_c}$, chunk size B_c, block size B_r.

Divide $\mathbf{K}, \mathbf{V}$ into $T_c = \left\lceil \frac{T}{B_c} \right\rceil$ blocks $\mathbf{K}_1, \ldots, \mathbf{K}_{T_c}$ and $\mathbf{V}_1, \ldots, \mathbf{V}_{T_c}$ of size $B_c \times d$ each, divide $\mathbf{F}$ into $T_c = \left\lceil \frac{T}{B_c} \right\rceil$ scalars $f_1, \ldots, f_{T_c}$.
Initialize $\mathbf{dQ} = (0)_{T \times d} \in \mathbb{R}^{T \times d}, \mathbf{dK} = (0)_{T \times d} \in \mathbb{R}^{T \times d}, \mathbf{dV} = (0)_{T \times d} \in \mathbb{R}^{T \times d}$.
for $1 \leq j \leq T_c$ **do**
  Load $\mathbf{K}_j, \mathbf{V}_j, f_j$ from HBM to on-chip SRAM.
  Initialize $\mathbf{dK}_j = (0)_{B_c \times d} \in \mathbb{R}^{B_c \times d}, \mathbf{dV}_j = (0)_{B_c \times d} \in \mathbb{R}^{B_c \times d}$.
  if $j = 1$ **then**
    On chip, Initialize $f_{j-1} = 0$.
  else
    Load f_{j-1} from HBM to on-chip SRAM.
  end if
  On chip, compute $M = f_j - f_{j-1}$.
  Divide $\mathbf{I}_{[f_{j-1}, \ldots, f_j]}$ into $M_r = \left\lceil \frac{M}{B_r} \right\rceil$ blocks $\mathbf{I}_1, \ldots, \mathbf{I}_{M_r}$ of size B_r each.
  for $1 \leq i \leq M_r$ **do**
    Load $\mathbf{I}_i$ from HBM to on-chip SRAM.
    Load $\mathbf{Q}_i \in \mathbb{R}^{B_r \times d}$ by index $\mathbf{I}_i \in \mathbb{R}^{B_r}$ from $\mathbf{Q}$ in HBM to on-chip SRAM.
    Load $\mathbf{O}_i \in \mathbb{R}^{B_r \times d}$ by index $\mathbf{I}_i \in \mathbb{R}^{B_r}$ from $\mathbf{O}$ in HBM to on-chip SRAM.
    Load $\mathbf{dO}_i \in \mathbb{R}^{B_r \times d}$ by index $\mathbf{I}_i \in \mathbb{R}^{B_r}$ from $\mathbf{dO}$ in HBM to on-chip SRAM.
    Load $r_i \in \mathbb{R}^{B_r}$ by index $\mathbf{I}_i \in \mathbb{R}^{B_r}$ from $\mathbf{R}$ in HBM to on-chip SRAM.
    On chip, compute $\mathbf{S}_{ij} = \mathbf{Q}_i \mathbf{K}_j^T \in \mathbb{R}^{B_r \times B_c}$.
    On chip, compute $\mathbf{P}_{ij} = \exp(\mathbf{S}_{ij} - r_i) \in \mathbb{R}^{B_r \times B_c}$.
    On chip, compute $\mathbf{dV}_j \leftarrow \mathbf{dV}_j + \mathbf{P}_{ij}^\top \mathbf{dO}_i \in \mathbb{R}^{B_c \times d}$.
    On chip, compute $\mathbf{dP}_{ij} = \mathbf{dO}_i \mathbf{V}_j^\top \in \mathbb{R}^{B_r \times B_c}$.
    On chip, compute $\mathbf{D}_i = \mathrm{rowsum}(\mathbf{dO}_i \circ \mathbf{O}_i) \in \mathbb{R}^{B_r}$.
    On chip, compute $\mathbf{dS}_{ij} = \mathbf{P}_{ij} \circ (\mathbf{dP}_{ij} - \mathbf{D}_i) \in \mathbb{R}^{B_r \times B_c}$.
    On chip, compute $\mathbf{dK}_j \leftarrow \mathbf{dK}_j + \mathbf{dS}_{ij}^\top \mathbf{Q}_i \in \mathbb{R}^{B_c \times d}$.
    On chip, compute $\mathbf{dQ}_i = \mathbf{dS}_{ij} \mathbf{K}_j \in \mathbb{R}^{B_r \times d}$.
    Write $\mathbf{dQ}_i \in \mathbb{R}^{B_r \times d}$ by index $\mathbf{I}_i \in \mathbb{R}^{B_r}$ to $\mathbf{dQ}$ in HBM.
  end for
  Write $\mathbf{dK}_j \in \mathbb{R}^{B_c \times d}$, $\mathbf{dV}_j \in \mathbb{R}^{B_c \times d}$ to $\mathbf{dK}$, $\mathbf{dV}$ in HBM.
end for
Divide $\mathbf{Q}, \mathbf{K}, \mathbf{V}, \mathbf{O}$ into $T_c = \left\lceil \frac{T}{B_c} \right\rceil$ blocks $\mathbf{Q}_1, \ldots, \mathbf{Q}_{T_c}$, $\mathbf{K}_1, \ldots, \mathbf{K}_{T_c}$, $\mathbf{V}_1, \ldots, \mathbf{V}_{T_c}$ and $\mathbf{O}_1, \ldots, \mathbf{O}_{T_c}$ of size $B_c \times d$ each.
Divide $\mathbf{dQ}, \mathbf{dK}, \mathbf{dV}, \mathbf{dO}$ into $T_c = \left\lceil \frac{T}{B_c} \right\rceil$ blocks $\mathbf{dQ}_1, \ldots, \mathbf{dQ}_{T_c}$, $\mathbf{dK}_1, \ldots, \mathbf{dK}_{T_c}$, $\mathbf{dV}_1, \ldots, \mathbf{dV}_{T_c}$ and $\mathbf{dO}_1, \ldots, \mathbf{dO}_{T_c}$ of size $B_c \times d$ each.
Divide $\mathbf{R}$ into $T_c = \left\lceil \frac{T}{B_c} \right\rceil$ blocks $r_1, \ldots, r_{T_c}$ of size B_c each.
for $1 \leq i \leq T_c$ **do**
  Load $\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i, \mathbf{O}_i, \mathbf{dQ}_i, \mathbf{dK}_i, \mathbf{dV}_i, \mathbf{dO}_i, r_i$ from HBM to on-chip SRAM.
  On chip, compute $\mathbf{S}_i = \mathbf{Q}_i \mathbf{K}_i^T \in \mathbb{R}^{B_c \times B_c}$.
  On chip, compute $\mathbf{P}_i = \exp(\mathbf{S}_i - r_i) \in \mathbb{R}^{B_c \times B_c}$.
  On chip, compute $\mathbf{dV}_i \leftarrow \mathbf{dV}_i + \mathbf{P}_i^\top \mathbf{dO}_i \in \mathbb{R}^{B_c \times d}$.
  On chip, compute $\mathbf{dP}_i = \mathbf{dO}_i \mathbf{V}_i^\top \in \mathbb{R}^{B_c \times B_c}$.
  On chip, compute $\mathbf{D}_i = \mathrm{rowsum}(\mathbf{dO}_i \circ \mathbf{O}_i) \in \mathbb{R}^{B_c}$.
  On chip, compute $\mathbf{dS}_i = \mathbf{P}_i \circ (\mathbf{dP}_i - \mathbf{D}_i) \in \mathbb{R}^{B_c \times B_c}$.
  On chip, compute $\mathbf{dQ}_i \leftarrow \mathbf{dQ}_i + \mathbf{dS}_i \mathbf{K}_i \in \mathbb{R}^{B_c \times d}$.
  On chip, compute $\mathbf{dK}_i \leftarrow \mathbf{dK}_i + \mathbf{dS}_i^\top \mathbf{Q}_i \in \mathbb{R}^{B_c \times d}$.
  Write $\mathbf{dQ}_i \in \mathbb{R}^{B_c \times d}$, $\mathbf{dK}_i \in \mathbb{R}^{B_c \times d}$, $\mathbf{dV}_i \in \mathbb{R}^{B_c \times d}$ to $\mathbf{dQ}$, $\mathbf{dK}$, $\mathbf{dV}$ in HBM.
end for
Return $\mathbf{dQ}, \mathbf{dK}, \mathbf{dV}$.

E More Results Under Mixed Domain Pre-Training Setting

Although our primary focus is on experiments under the limited domain pre-training setting, we also conducted preliminary experiments on the Top-1 Element Selection Architecture under mixed domain pre-training setting during the early stages of this work. The results, as shown in Figures 8 and 9, indicate that the performance of the Top-1 Element Selection Architecture aligns with the conclusions drawn in the paper, with no degradation in base capabilities. This suggests that the usability of the proposed architecture remains consistent across different pre-training settings.

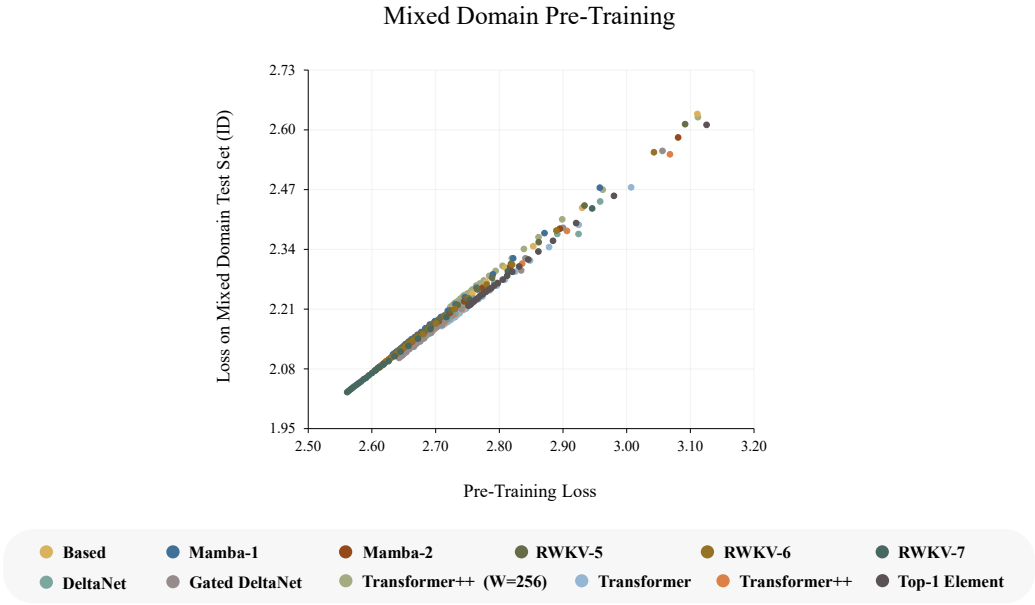


Figure 8: Language modeling results of various sequence modeling architectures under mixed domain pre-training settings. (Model parameters≈110M, pre-trained tokens=100B and sequence length=2k)

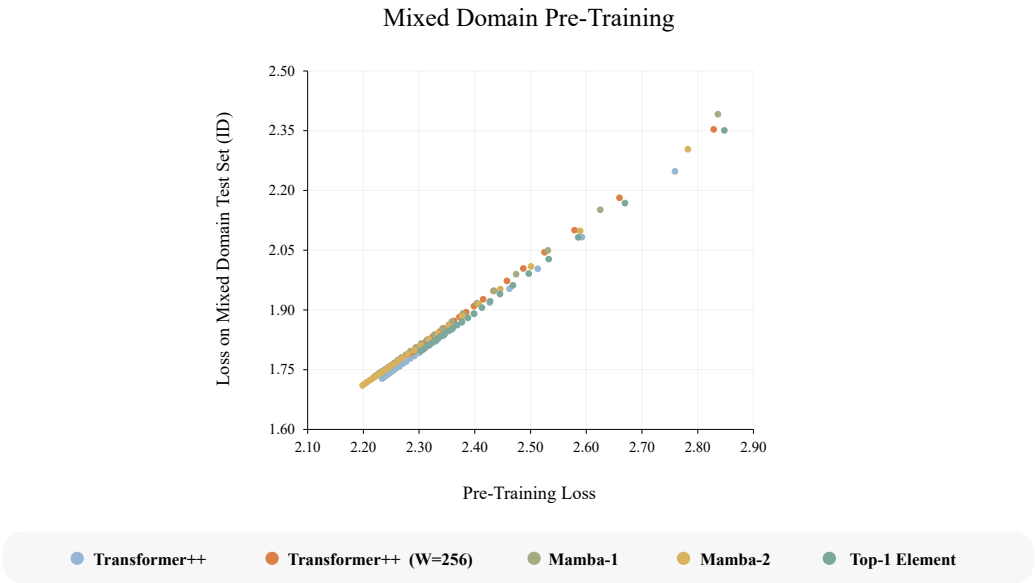


Figure 9: Language modeling results of various sequence modeling architectures under mixed domain pre-training settings. (Model parameters≈1.3B, pre-trained tokens=100B and sequence length=2k)

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: In Abstract and Section 1.

   Guidelines:

   - The answer NA 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 NA 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: In Section 6.

   Guidelines:

   - The answer NA 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: [NA]

Justification: Not involving proof.

Guidelines:

- The answer NA 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: In Appendix A and B.

   Guidelines:

   - The answer NA 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: We have provided the most critical GPU kernel implementation code in the supplemental material, and will also make the source code publicly available.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.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://nips.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, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: In Appendix A and B.

   Guidelines:

   - The answer NA 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: [Yes]

   Justification: We use default settings of open-source LLM evaluation frameworks to ensure the stability and reproducibility of the results.

   Guidelines:

   - The answer NA 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: In Appendix A.

   Guidelines:

   - The answer NA 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: Yes.

   Guidelines:

   - The answer NA 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: [NA]

    Justification: This work is a foundational research.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA 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., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: We do not release models.

    Guidelines:

    - The answer NA 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: The corresponding references are provided for the existing assets used in this paper.

    Guidelines:

    - The answer NA 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: [NA]

    Justification: N/A.

    Guidelines:

    - The answer NA 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: [NA]

    Justification: N/A.

    Guidelines:

    - The answer NA 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: [NA]

    Justification: N/A.

    Guidelines:

    - The answer NA 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 rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: N/A.

Guidelines:

- The answer NA 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 (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.