

---

# DeltaFormer: Unlock the state space of Transformer

---

Anonymous Author(s)

Affiliation

Address

email

## Abstract

In recent years, large language models with Transformer architecture as the core have made breakthrough progress in many fields. At the same time, there are also some weaknesses in the large language model that have prompted people to reflect, among which the most fundamental one is the reflection on the Transformer architecture. The Transformer architecture has high parallelism and can fully utilize the computing power of GPUs, thus replacing models such as LSTM in the past few years. However, high parallelism is not a free lunch, as it fundamentally limits the performance of models. Especially, the problems that logarithmic precision Transformer architecture can solve are strictly limited to the  $TC^0$ . And there are many important issues that are usually considered out of  $TC^0$ , such as Python code evaluation, entity tracking, chess, and other state tracking tasks. Meanwhile, some recent state space methods based on Delta Rule have been able to break through the  $TC^0$  architecture, but they are limited by fixed size state spaces and perform poorly on many tasks. To this end, we have re-examined the Transformer from the perspective of a state space with kernel functions and propose an improved Transformer called DeltaFormer. We have theoretically and practically demonstrated that the proposed new architecture can break through the limitation of the inherent  $TC^0$  expressivity of Transformers and verified that it is not weaker than standard Transformer in language modeling tasks. We hope our work can provide inspiration for designing more expressive models.

## 1 Introduction

In the field of artificial intelligence (AI), the Transformer model [66] has attracted widespread attention for its outstanding performance and broad application prospects since its inception. As the core architecture of modern AI systems, Transformer has shown amazing performance in many fields [8, 27, 15, 50, 51]. Although Transformer has achieved significant success, it has performed poorly on many tasks without chain-of-thought, which has also sparked reflection among researchers.

Another noteworthy fact is that existing large models have demonstrated amazing reasoning abilities after reinforcement learning [25, 24, 59], recent studies have shown that models have not unlocked new abilities through RL, and their abilities are limited by pre-training [73, 60]. As the core of pre-training, the model architecture is also causing researchers to reflect. One of the most noteworthy things is the expressivity of Transformer models. The existing transformer class models have limited expressive power, and it has been proven that in the  $TC^0$  class [39], chain of thought needs to be utilized to solve problems in larger classes [20, 32]. At the same time, there are a large number of real-world tasks that are generally surpassed by  $TC^0$  and fall within the  $NC^1$ , such as entity tracking, evaluating Python code and tracking chess [38], if we accept a widely accepted hypothesis  $TC^0 \neq NC^1$ . We speculate that this is likely the fundamental reason why existing large-scale models achieve randomness in entity tracking tasks [12]. So, is there a way to break through the  $TC^0$  expressivity where the Transformer is located?

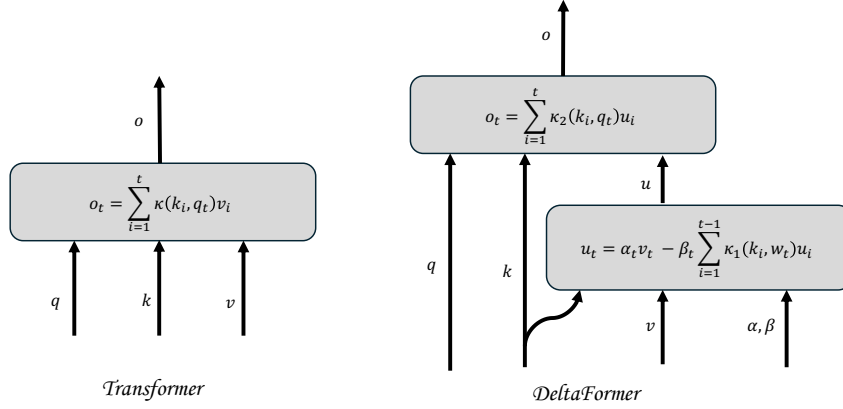


Figure 1: Comparison of core Components between Transformer and DeltaFormer.

In the field of finite state space methods, there have been recent works demonstrating the use of delta rule to break through the expressivity of  $TC^0$  [47, 61]. But they use finite state spaces, which undoubtedly face inherent problems with finite state spaces, such as difficulty in performing retrieval tasks [26]. So can we draw some inspiration from it to help us design more expressive Transformers?

To this end, we utilized the delta rule of kernel functions to re-examine the Transformer architecture and proposed a new architecture called DeltaFormer. We have verified through theory and practice that it has expressive power beyond Transformers. And it can be no weaker than standard transformers in language modeling tasks.

The main contributions of this work are as follows:

- We have re examined delta rule from the perspective of kernel functions and proposed a new architecture that implicitly assigns the state space to the Transformer, called DeltaFormer. And we design a chunk-wise algorithm that can efficiently implement DeltaFormer in parallel on GPUs.
- We have theoretically demonstrated that DeltaFormer has stronger expressivity than Transformer. And we prove that when combining delta rule and non-linear kernel we can use KV cache of  $O(T \log n)$  size to track  $T$  exchanges of  $n$  objects. Further, if we allow KV cache compression every  $O(n)$  steps, we only requires  $O(n \log n)$  space, which is much smaller than linear kernel.

## 2 Related Work

**Circuit Complexity** Boolean circuits have been introduced to study parallel complexity. Among the key classes in this context are  $NC$  and  $TC$ .  $TC$  class focuses on problems solvable by Boolean circuits with a majority gate as the primary operation, where  $TC^0$  specifically denotes problems solvable by constant-depth, polynomial-size threshold circuits. Given input length  $n$ , previous work have shown that constant-depth Transformer with finite-precision ( $poly(n)$ ) embedding size can only solve problems in  $TC^0$  without chain-of-thought [39]. Recently, there have been some studies showing that DeltaNet [58, 72] and some variants [71, 47, 61] can overcome the complexity limitation of  $TC^0$  where Transformer is located, resulting in higher expressivity and achieving better performance in tasks such as state tracking[22, 61, 38].

**Model Architecture** Over the past many years, people have attempted various structures to enhance model’s abilities from early RNN [41] and LSTM [64] to the dominant Transformer [66] today. Transformer has quadratic complexity, and many efforts have been made to improve them [46, 23, 63, 4, 49]. Nevertheless, Transformers still has many excellent properties that cannot be replaced temporarily, especially their ability to retrieve information and adapt to a form of dynamic sparsity [26, 3, 42]. Currently, popular language models [1, 16, 70] still use Transformer as the main

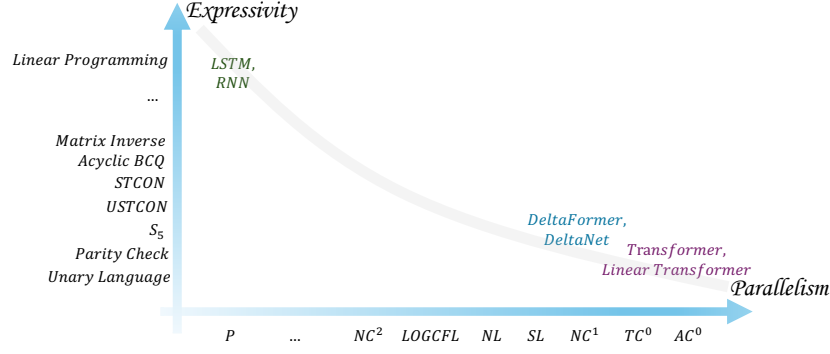


Figure 2: Parallelism and expressivity between models. The higher the parallelism of a model, the more restricted it can be. Previous work has shown that the Log-Precision Transformer is in  $TC^0$  [39] and constant precision Transformer in  $AC^0$  [32]. The model based on DeltaNet can perform tasks that exceed the expressivity of  $TC^0$ , if we accept that  $TC^0 \neq NC^1$  [47]. In addition, even for models with the same degree of parallelism, their performance is affected by the model capacity. One of the most intuitive examples is that Transformer can perform better than Linear Transformer in retrieval related tasks [26].

73 architecture and some works use a hybrid architecture [53, 33, 30]. Our paper aims to make  
 74 improvements to break through the expressive limitation of Transformer.

75 **Understanding Transformer** Since the popularity of Transformer [67], extensive research has  
 76 also been conducted on their underlying machines. Some works analyze its powerful approximation  
 77 ability [74, 14, 29]. There are also some tasks aimed at understanding the dynamics of model  
 78 training [37, 6, 65] and interpretability [9, 35, 36, 2]. With the development of large models, more  
 79 and more understanding works focus on analyzing the contextual learning ability of Transformer  
 80 [21, 31, 44, 19, 17, 18, 69]. An important aspect for understanding the behavior of large models is  
 81 the associative memory [43, 28, 5, 68]. And Delta rule can be also treated as an update rule of the  
 82 associative memory. We have re-examined the delta rule from the perspective of kernel functions and  
 83 proposed a new model.

### 84 3 Method

#### 85 3.1 Re-examining the Delta Rule from the Perspective of Kernel Functions

86 The research on delta rule has a long history [62, 48, 54] and has recently attracted the interest of  
 87 model architecture researchers [58, 72, 47]. The method derives its name from updating weights  
 88 based on the difference between the prediction and the target. The mathematical expression is that  
 89 there is a state space  $S_t$  at time  $t$ , and at this time, the input is  $k_t$ , and the target is  $v_t$ . The model  
 90 needs to minimize  $\|S_t k_t - v_t\|_2$ . Then we use one step SGD with learning rate  $\beta_t$ :

$$S_{t+1} = S_t - \beta_t(S_t k_t - v_t)k_t^\top = S_t(I - \beta_t k_t k_t^\top) + \beta_t v_t k_t^\top. \quad (1)$$

91 and we use the following rule to read information out of  $S_t$ :

$$o_t = S_t q_t. \quad (2)$$

92 We generalize the above process and use kernel functions to measure similarity  $\kappa(x, y) = \psi(x)^\top \psi(y)$ ,  
 93 where  $x, y \in R^d$ ,  $\psi(\cdot)$  is a function that maps to an infinite dimensional space. Then we generalize  
 94 the delta rule as:

$$S_{t+1} = S_t(I - \beta_t \psi(k_t)\psi(k_t)^\top) + \beta_t v_t \psi(k_t)^\top, \quad (3)$$

95 and we use the following rule to read information out of  $S_t$ :

$$o_t = S_t \psi(q_t). \quad (4)$$

We can rewrite Eq. 3 into a simpler equivalent form:  $S_{t+1} = S_t + \beta_t u_t k_t^\top$ , where  $u_t$  is determined by  $\{(k_i, v_i)\}_{i=1}^{t-1}$ . According to the derivation in Appendix A, we get:

$$u_t = \beta_t v_t - \beta_t \sum_{i=1}^{t-1} \kappa(k_i, k_t) u_i, \quad (5)$$

$$o_t = \sum_{i=1}^t \kappa(k_i, q_t) u_i. \quad (6)$$

Although  $\psi(\cdot)$  is a function that may map to an infinite dimensional space, there is no  $\psi(\cdot)$  in Eq. 5 and Eq. 6, and all terms in Eq. 5 and Eq. 6 can be finite. If we set  $\kappa(x, y)$  as  $x^\top y$ , we can find that this is exactly the DeltaNet [58, 72].

**Generalized design.** We can actually perform some more general operations, such as we can also apply separate kernel functions ( $\kappa_1, \kappa_2$ ) in Eq. 5 and Eq. 6, use different gates ( $\alpha_t, \beta_t$ ) before  $v_t$  and  $\sum_{i=1}^{t-1} \kappa(k_i, k_t) u_i$ , use  $w_t$  in Eq. 5 to control the write and delete instead of  $k_t$ . Then we can get:

$$u_t = \alpha_t v_t - \beta_t \sum_{i=1}^{t-1} \kappa_1(k_i, w_t) u_i, \quad (7)$$

$$o_t = \sum_{i=1}^t \kappa_2(k_i, q_t) u_i. \quad (8)$$

If we set  $\beta = 0$ ,  $\alpha = 1$ , and use  $o_t = \frac{\sum_{i=1}^t \exp(k_i^\top q_t / \sqrt{d}) u_i}{\sum_{i=1}^t \exp(k_i^\top q_t / \sqrt{d})}$ , we can find this is the standard Transformer. Of course, we can also try introducing decay factors like Fox [34] to replace the role of positional encoding, but this is not within the scope of our discussion.

We call Eq. 7 and Eq. 8 as DeltaFormer. So what are the advantages of this model compared to previous models, is there any fundamental difference in performance between this model and Transformer and DeltaNet?

### 3.2 Beyond the Expressivity of Transformer

As previous literature have shown, the performance of Transformer is limited to  $TC^0$  if chain of thought is not performed [39, 32, 20]. Based on a hypothesis that is considered correct,  $TC^0 \neq NC^1$ , We will prove that DeltaFormer can solve a problem which is  $NC^1$ -complete under  $AC^0$  reduction. Specifically, we provide a constructive proof as shown in Theorem 1, and the detailed proof is in Appendix B.

**Assumption 1** *There exist  $n$  state points on a  $d$ -dimensional unit sphere, and the absolute value of the inner product of any two distinct state points is less than or equal to  $\epsilon(d, n)$ , which means:*

$$\exists x_1, x_2, \dots, x_n \in \mathbb{R}^d \text{ s.t. } \|x_i\|_2 = 1 (\forall i), \quad \max_{i \neq j} |x_i^\top x_j| \leq \epsilon(d, n) < \frac{1}{8}.$$

**Assumption 2** *There is a function  $f$  satisfies:*

$$\forall x \in \{-1, 0, 1, 2\}, \forall \tilde{x} \in U(x, 4\epsilon(d, n)) : f(\tilde{x}) = x.$$

**Theorem 1** *Based on the above assumptions, we can consider initializing  $n$  key-value pairs as  $\{(k_1, v_1), \dots, (k_n, v_n)\}$ . The keys  $\{k_1, \dots, k_n\}$  lie on a  $d$ -dimensional unit sphere and satisfies Assumption 1, which means:*

$$\forall i, j \in \{1, \dots, n\}, i \neq j : \|o_i\|_2 = 1, \quad |k_i^\top k_j| \leq \epsilon(n, d).$$

*and define an attention mechanism as follows:*

$$u_t = o_t - \sum_{i=1}^{t-1} f(k_i^\top k_t) u_i, \quad o_t = \sum_{i=1}^t f(q_t^\top k_i) u_i,$$

where  $f(\cdot)$  satisfies Assumption 2 and it is noted that  $\forall i \in \{1, \dots, n\}$ , since  $f(k_i^\top k_i) = 0$ , we have  $u_i = o_i$ .

At the current step  $t$ ,  $t > n$ , the value corresponding to  $k_i$  is denoted by  $\tilde{v}_i$ ,  $i \in \{1, \dots, n\}$ . Note that, after  $t - 1 - n$  exchanges,  $\tilde{v}_i$  is not necessarily equal to the initially assigned  $v_i$ .  $\forall 1 \leq t_2 < t_1 \leq n$ , to exchange the stored values  $\tilde{v}_{t_1}$  and  $\tilde{v}_{t_2}$  corresponding to  $k_{t_1}$  and  $k_{t_2}$ , it suffices to construct:

$$k_t = k_{t_1} - k_{t_2}, \quad v_t = 0$$

When retrieving the values:

Query  $q_t = k_{t_1}$ , then  $o_t = \tilde{v}_{t_2}$ ;

Query  $q_t = k_{t_2}$ , then  $o_t = \tilde{v}_{t_1}$ ;

Query  $q_t = k_{t_3}$ ,  $1 \leq t_3 \leq n$ ,  $t_3 \neq t_1, t_2$ , then  $o_t = \tilde{v}_{t_3}$ .

This implies the exchange of values corresponding to  $k_{t_1}$  and  $k_{t_2}$  is completed.

### 3.2.1 Re-exam Assumptions

Now we re-exam the two assumptions in Theorem 1.

What relationship between  $d$  and  $n$  must hold for Assumption 1 to be satisfied? According to Theorem 5.2.1 in [76],

For every  $\alpha \in (0, 1)$  and  $\varepsilon > 0$ , there exists  $c > 0$  such that for every  $d$ , one can find at least  $2^{cd}$  unit vectors in  $\mathbb{R}^d$  whose pairwise inner products all lie in  $[\alpha - \varepsilon, \alpha + \varepsilon]$ .

Setting  $\alpha = 0.01$  and  $\varepsilon = 0.1$ , we have  $\epsilon(d, n) \leq 0.11 < \frac{1}{8}$ . This implies that for Assumption 1 to hold, it is required that  $d = O(\log n)$ .

Regarding the choice of  $f(\cdot)$  in Assumption 2, intuitively, we can use the rounding function  $\text{round}(\cdot)$ , i.e., setting the input to the nearest integer. Under appropriate rounding precision, this function can satisfy the assumption. If we prefer the common exponential kernel  $\exp(\cdot)$ , theoretically, a multi-query attention with four shared heads can express an  $f(\cdot)$  that fulfills Assumption 2.

In summary, according to Theorem 1, delta rule-based softmax attention can achieve state exchange between historical time steps  $t_1$  and  $t_2$ , provided that  $d = O(\log n)$  and the projection matrices  $W_k$  and  $W_v$  learn mappings such that  $k_t = k_{t_1} - k_{t_2}$  and  $v_t = 0$ .

### 3.2.2 Native State Space Compression

What is the space cost in Theorem 1, if we want to track  $T$  exchanges of  $n$  states? Obviously, we need to set  $d = O(\log n)$  and use KU cache with length  $T$ . So it needs  $O(T \log n)$  spaces.

If we read out  $\{\tilde{v}_1, \dots, \tilde{v}_n\}$  based on  $\{k_1, \dots, k_n\}$  every  $O(n)$  steps, then re-write  $\{(k_i, \tilde{v}_i)\}_{i=1}^n$  into KU cache. We only use the KU cache with length  $O(n)$ , so the total space cost is  $O(n \log n)$ , which is much smaller than the  $O(n^2)$  space when  $f(\cdot)$  is an identity mapping, in which RWKV7 [47] use a  $5 \times 5$  matrix to track the exchange of 5 elements.

For comparison, employing non-linear kernels unlocks the powerful potential of the delta rule: we can track the exchange of exponentially many states instead of merely the previous  $n = O(d)$  states.

In summary, as a general form of transformer, DeltaFormer not only does it surpass the inherent  $TC^0$  expressivity of the Transformer, but it can also track the exchange of  $n$  objects using much smaller state space than models such as RWKV7.

## 3.3 Efficient Chunk-wise Algorithm on GPUs

High parallelism on GPUs is an important reason why Transformers have been able to outperform non parallelizable models such as LSTM during the scaling process of large models in the past period of time. But from a high-level perspective, there is an irreconcilable contradiction between parallelism and expressivity. Obviously, the parallelizability of DeltaNet has decreased, and if it is reduced to the

level of LSTM, it will lose its practical significance. Therefore, in order to make the more expressive model more practical, we must find an efficient implementation method on GPUs.

Consider  $q, k, v, u \in \mathbb{R}^d$ , sequence length  $T$ , and assume  $T \gg d$ . For the sake of simplicity and without loss of generality, we only consider the fast calculation of the loop part  $u_t = v_t - \sum_{i=1}^{t-1} \kappa(k_i, k_t) u_i$  here, while the calculation of  $o$  is similar to Flash Attention [13].

If we compute  $\{u_1, \dots, u_T\}$  according to  $u_t = v_t - \sum_{i=1}^{t-1} \kappa(k_i, k_t) u_i$ , the computational complexity is  $O(T^2 d)$ . Note that, since each  $u_t$  depends on all previous  $u_{<t}$ , this algorithm cannot be parallelized, resulting in a  $O(T)$  iteration. Consequently, it fails to utilize GPUs' parallel computation capabilities.

In fact, the computations of  $\{u_1, \dots, u_T\}$  can be parallelized at the cost of increased computes. Rewriting  $u_t = v_t - \sum_{i=1}^{t-1} \kappa(k_i, k_t) u_i$  in matrix form, we have:

$$U = V - AU, \quad (9)$$

where the  $t$ -th rows of matrices  $U$  and  $V$  are  $U_{t,:} = u_t$  and  $V_{t,:} = v_t$ , respectively. The similarity matrix  $A$  is defined as  $A_{t,i} = \kappa(k_t, k_i)$ , (e.g.  $\frac{\exp(k_i^\top k_t)}{Z_t^{(1)}}$  or  $k_t^\top k_i$ ) and is lower-triangular. Thus,  $U$  can be solved via matrix operations:

$$U = (I + A)^{-1} V. \quad (10)$$

This approach requires a matrix inversion operation, increasing computational complexity to  $O(T^3)$ , but can be parallelized in GPUs.

Intuitively, we can seek a trade-off between the two approaches described above. By dividing the sequence of length  $T$  into  $N$  equally sized chunks, each having length  $C = \lfloor \frac{T}{N} \rfloor$ , we can sequentially compute the sequence of  $u$  for each chunk using parallelized matrix operations. As a result, the recurrent steps from  $O(T)$  to  $O(N)$ , while the computational complexity transitions from  $O(T^2 d)$  to  $O(T^2 d + TCd + TC^2)$ . Essentially, this approach trades computational resources for reduced runtime. For the pseudo-code of chunk wise implementation, please refer to the Appendix C.

## 4 Experiment

### 4.1 Track the Exchange of Elements

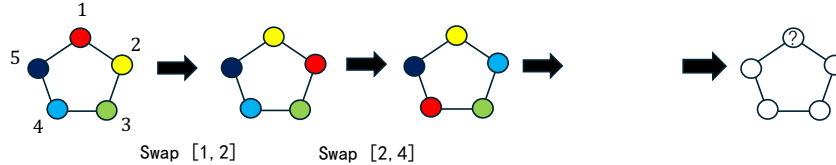


Figure 3: Swap task diagram. At the beginning, tokens of different colors are placed at positions 1 to 5, and the tokens of two positions are exchanged at each step. We expect the model to query what the token for each position is at each step. Simply but without loss of generality, we default to outputting the token at the first position to avoid introducing a "query token". This task can also be tokenized into a task with an input vocabulary size of  $C_5^2 = 10$  and an output vocabulary size of 5.

Although our theorem proves that we can track the exchange of  $n$  objects, it still needs to be proven through experiments. By using gradient descent, can DeltaFormer learn to track the exchange of  $n$  objects from the data. For this purpose, we conduct an experiment in this section to verify this. The setting is shown in Figure 3, and the default context length is 16.

**DeltaFormer can track the exchange of elements.** We compared DeltaFormer and Transformer under different similarity function designs, as shown in Figure 4. Under almost imaginable simple  $\kappa_1(\cdot)$  designs, DeltaFormer has achieved better results than Transformer models. And the 1-layer DeltaFormer can execute and track the exchange operations of 5 elements. But increasing the number of layers in the transformer did not improve either. We speculate that there may be optimization issues involved, even with DeltaFormer. We will explore this hypothesis in the following sections.

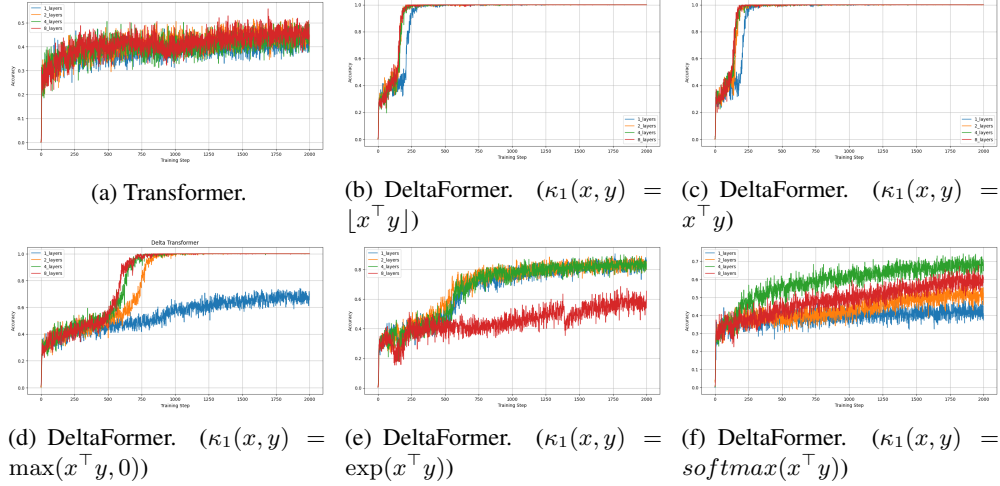


Figure 4: Comparison of Transformer and DeltaFormer using different similarity functions  $\kappa_1(\cdot)$  for performing swapping tasks. For  $\kappa_2(\cdot)$ , we use the softmax function to maintain consistency with Transformer. Pay attention to the scale of the y-axis. To ensure convergence,  $\lfloor \cdot \rfloor$  means round to two decimal, such as  $\lfloor 1.236 \rfloor = 1.24$ .

197 **The similarity function used in Eq. 7 is important.** Another noteworthy point is that the choice  
 198 of different similarity functions also has a significant impact on the final effect. According to our  
 199 construction proof in Theorem 1, the similarity function can effectively track 5 elements. The closer  
 200 the similarity function is to the constructive method, the better the performance. The normalization  
 201 term of *softmax* has a negative impact on the similarity calculation of *exp*. It is worth noting that  
 202 the similarity calculation function  $k_2$  used in our reading method is not based on the constructed  
 203 similarity to ensure that it is the same as the standard attention, but instead uses softmax. Even so, a  
 204 suitable  $k_1$  can achieve 100% effectiveness. Because our Theorem 1 actually proves that there is a  
 205 way to read the elements at each position in a certain form of  $u$ . This means that the exchange of  
 206 elements is implicitly included in the update of  $u$ . Intuitively speaking, if the similarity selection  
 207 of  $\kappa_1$  is not appropriate, it will cause more cumulative errors in the update of  $u$ . Mathematically  
 208 speaking, it actually reflects the perturbation of a inverse matrix is likely to be ill-conditioned. For  
 209 detailed instructions, please refer to Appendix D. We also conducted stress tests on the Round and  
 210 Linear kernels in Appendix B.1, with  $d = 128$ , but track the exchange of  $n \geq 128$  elements.

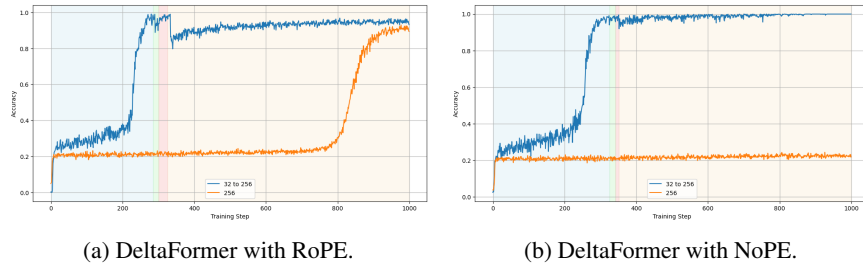


Figure 5: Comparison of DeltaFormer using different learning strategy and position embedding. Each use  $\kappa_1(x, y) = \lfloor x^\top y \rfloor$ . "32 to 256" means that the initial training length is 32, which means the number of swaps is 32. When the accuracy reaches 0.99, the training length will be doubled until it reaches 256. Each color gradient in the image represents a doubling of the training length. And "256" means that the model is trained on a training length of 256 from the beginning. The y-axis reflects the accuracy at the current training length.

211 **Curriculum learning is important.** As shown in the figure, we initially trained at a length of 256,  
 212 and the convergence speed of the model was very slow. So we decided to gradually lengthen the  
 213 window from 32, that is, gradually increase the difficulty. We find that on the basis of such curriculum  
 214 learning, the model can achieve better performance with fewer computation.

**The role of rotary embeddings.** Because our proof of Theorem 1 does not require positional embeddings. Therefore, another interesting experiment is that we removed the default rotary position embeddings. We find that after removing the positional embedding, the convergence of the model slowed down, and even when trained directly at a length of 256, the model get a random score. On the other hand, after removing RoPE, "32 to 256" can achieve 100% accuracy. And during the extension process, the performance degradation of NoPE at jump points decreases, indicating that NoPE has better length generalization. Therefore, we speculate that RoPE may have damaged the expression of model and extrapolation ability, but it is more conducive to optimization.

## 4.2 Reachability of directed acyclic graphs

Furthermore, we have a simple graph connectivity task, which involves determining reachability on a directed acyclic graph. For simplicity, we only consider whether other nodes can be reached by the first node in a certain topology ranking. And each node encodes at most its neighboring node information at the beginning. Due to the final output being true and false, in order to avoid class imbalance, we construct the input data by dividing  $n$  points into 2 classes on average, with one tree for each class. So we only judge the reachability of other nodes starting from a certain root node. Then we only encode the parent node information for each node.

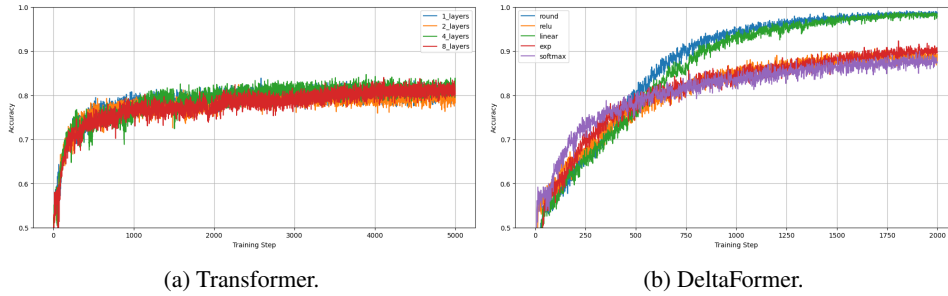


Figure 6: Comparison of Transformer and DeltaFormer using different similarity functions  $\kappa_1(\cdot)$  for performing swapping tasks. For  $\kappa_2(\cdot)$ , we use the softmax function to maintain consistency with Transformer. Pay attention to the scale of the y-axis.

**DeltaFormer performer better than Transformer.** We conducted experiments on 32 nodes, as shown in Figure 6. A multi-layer transformer model is also difficult to achieve 100% accuracy, but a single-layer DeltaFormer can do very well. In theory, a transformer requires  $O(\log n)$  layers to perform connectivity checks with  $n$  nodes [56]. However, based on Figure 6a, we speculate that the optimization problem also constrains the effectiveness of the Transformer in this task.

**The power of matrix inverse.** As shown in Eq. 10, the calculation of  $u$  can be rewrite by matrix inverse. If the adjacency matrix  $A$  is known, then to determine whether nodes  $i$  and  $j$  are connected, simply calculate  $A, A^2, A^3, \dots, A^n$  in sequence, and then observe whether the  $(i, j)$  elements of this matrix are greater than 0. And  $(I - A)^{-1}$  can be seen as an approximation of  $I + A + \dots + A^n$ . Therefore, the operation of matrix inversion greatly improves the expressivity of the model. We can expect a model with matrix inverse to achieve better performance in graph related tasks.

## 4.3 Language modeling

To verify that DeltaFormer does not affect language modeling capabilities, we conducted experiments on a small scale. Following prior work [71], we use open-source code of them and open-source dataset Fineweb-edu for training and the open-source evaluation tool lm-evaluation-harness for benchmark evaluation. The benchmarks that include LAMBADA [LMB.;[45]], PiQA[7], HellaSwag [Hella.;[75]], WinoGrande [Wino.;[55]], ARC-easy (ARC-e) and ARC-challenge (Arc-c)[11], Boolq [10], OpenbookQA [OBQA.;[40]], SIQA [57] and Copa [52]. We train on a 340M parameter scale with 15B tokens using a warm up decoy learning rate scheduling with a peak learning rate of 2e-3. The context length is 2,048 and the global batch size is 0.5M tokens. The experimental results are shown in Table 1.

Due to the fact that at this scale, the fluctuations of one or two points in these benchmark indicators are considered random. Therefore, we can only slightly argue that, with the same number of parameters,



Model	ARC-c acc_n ↑	ARC-e acc ↑	Boolq acc ↑	Copa acc ↑	Hella. acc_n ↑	LMB. acc ↑	OBQA. acc_n ↑	PIQA acc ↑	SCIQ. acc ↑	Wino. acc ↑
Transformer	28.58	59.61	60.00	68.00	40.11	34.50	38.40	67.25	81.60	52.01
DeltaFormer										
$\kappa_1(x, y) = x^\top y$	29.01	59.09	60.52	69.00	40.43	34.17	38.20	67.90	80.60	50.04
$\kappa_1(x, y) = \text{Relu}(x^\top y)$	28.41	57.62	59.88	68.00	40.07	32.76	37.00	65.83	80.10	51.22
$\kappa_1(x, y) = \lfloor x^\top y \rfloor$	28.50	58.33	60.09	70.00	40.29	33.03	35.40	67.03	81.50	51.92
$\kappa_1(x, y) = \text{softmax}(x^\top y)$	28.92	57.89	61.80	69.00	40.21	34.05	37.40	67.21	81.40	52.41

Table 1: Comparison of DeltaFormer and its variants on various language modeling benchmarks.

DeltaFormer is not weaker than standard Transformers in language modeling tasks. Even with different similarity functions, the differences are very small, which is also different from the findings of Section 4.1. With sufficient resources, we would like to train a large-scale model and compare its performance on complex inference benchmarks with baseline.

## 5 Discussion

**Expression.** Due to the fact that matrix inversion is a task within  $NC^2$ , it is easy to observe that the upper bound of DeltaFormer’s performance is within  $NC^2$ . We can also design more expressive models at the cost of constantly sacrificing parallelizability, and even more extreme, it becomes an inherently non parallelizable model like LSTM. Ultimately, we need to find a trade-off between parallelism and expressivity. Hardware or other environmental factors can affect the balance point of this trade off. On this trade-off path, perhaps scaling a model with slightly lower parallelism and higher expression than Transformer is the beginning.

**Optimization.** In order to ultimately obtain a satisfactory model, the optimization process should also be considered. During our exploration, we discovered some interesting phenomena, such as gradually extending the context length, which can be beneficial for DeltaFormer to learn element tracking tasks. In addition, the differences between DeltaFormer and the standard Transformer may lead to significant variations in the optimization process. In addition, as shown in Section 4.1, rotary embeddings seem to have a negative effect on performance, but beneficial for optimization. Studying the optimization process of these models may be interesting, and we will leave it for future work.

**Scaling.** Scaling this model to a larger scale and observing its effects would be an interesting thing, and we speculate that there will also be some interesting phenomena during the scaling process. For example, we speculate that the transformer needs to stack layers to achieve fitting at a certain scale in order to solve tasks beyond its expressive power, while the DeltaFormer may require shallower layers to learn, so the optimal aspect ratio may be different. At the same time, in order to adjust DeltaFormer to its optimal setting, we may need to rethink the original components in Transformer. It will also be interesting to think about the parameter extension of DeltaFormer like Transformer.

## 6 Limitation

Firstly, although we have proposed an algorithm that can be efficiently implemented on GPUs, the current efficiency is not sufficient, and we need to seek algorithms that can be efficiently implemented on GPUs. Secondly, although we have proposed a more expressive model, our evaluation is limited to toy tasks and small-scale language modeling tasks, and there is a lack of industrial scale language model training to demonstrate its ability to achieve higher performance in complex tasks.

## 7 Conclusion

In this work, we extended the delta rule based on the kernel function and proposed the DeltaFormer. We proved theoretically and empirically that the DeltaFormer broke through the  $TC^0$  expression of Transformer. Especially, the introduction of nonlinear functions allows DeltaFormer to track exponential element exchanges in a same dimension compared to linear kernel. At the same time, our experiments show that the new structure is not inferior to the standard Transformer in language modeling. We will extend this model to industrial scale for future work. We hope this work can shed light on the design of transformers from its expression.

## References

- [1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.
- [2] Zeyuan Allen-Zhu. ICML 2024 Tutorial: Physics of Language Models, July 2024. Project page: <https://physics.allen-zhu.com/>.
- [3] Josh Alman and Hantao Yu. Fundamental limitations on subquadratic alternatives to transformers. *arXiv preprint arXiv:2410.04271*, 2024.
- [4] Maximilian Beck, Korbinian Pöppel, Markus Spanring, Andreas Auer, Oleksandra Prudnikova, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. xlstm: Extended long short-term memory. *arXiv preprint arXiv:2405.04517*, 2024.
- [5] Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. *arXiv preprint arXiv:2501.00663*, 2024.
- [6] Alberto Bietti, Vivien Cabannes, Diane Bouchacourt, Herve Jegou, and Leon Bottou. Birth of a transformer: A memory viewpoint. *Advances in Neural Information Processing Systems*, 36:1560–1588, 2023.
- [7] Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin 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.
- [8] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.
- [9] Hila Chefer, Shir Gur, and Lior Wolf. Transformer interpretability beyond attention visualization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 782–791, 2021.
- [10] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*, 2019.
- [11] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.
- [12] Vanya Cohen and Raymond Mooney. Met-bench: Multimodal entity tracking for evaluating the limitations of vision-language and reasoning models. *arXiv preprint arXiv:2502.10886*, 2025.
- [13] Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023.
- [14] Yihe Dong, Jean-Baptiste Cordonnier, and Andreas Loukas. Attention is not all you need: Pure attention loses rank doubly exponentially with depth. In *International conference on machine learning*, pages 2793–2803. PMLR, 2021.
- [15] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.
- [16] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

- [17] Nelson Elhage, Tristan Hume, Catherine Olsson, Neel Nanda, Tom Henighan, Scott Johnston, Sheer ElShowk, Nicholas Joseph, Nova DasSarma, Ben Mann, Danny Hernandez, Amanda Askell, Kamal Ndousse, Andy Jones, Dawn Drain, Anna Chen, Yuntao Bai, Deep Ganguli, Liane Lovitt, Zac Hatfield-Dodds, Jackson Kernion, Tom Conerly, Shauna Kravec, Stanislav Fort, Saurav Kadavath, Josh Jacobson, Eli Tran-Johnson, Jared Kaplan, Jack Clark, Tom Brown, Sam McCandlish, Dario Amodei, and Christopher Olah. Softmax linear units. *Transformer Circuits Thread*, 2022. <https://transformer-circuits.pub/2022/solu/index.html>.
- [18] Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. Toy models of superposition. *Transformer Circuits Thread*, 2022. [https://transformer-circuits.pub/2022/toy\\_model/index.html](https://transformer-circuits.pub/2022/toy_model/index.html).
- [19] Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, et al. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 1(1):12, 2021.
- [20] Guhao Feng, Bohang Zhang, Yuntian Gu, Haotian Ye, Di He, and Liwei Wang. Towards revealing the mystery behind chain of thought: a theoretical perspective. *Advances in Neural Information Processing Systems*, 36:70757–70798, 2023.
- [21] Shivam Garg, Dimitris Tsipras, Percy S Liang, and Gregory Valiant. What can transformers learn in-context? a case study of simple function classes. *Advances in Neural Information Processing Systems*, 35:30583–30598, 2022.
- [22] Riccardo Grazi, Julien Siems, Arber Zela, Jörg KH Franke, Frank Hutter, and Massimiliano Pontil. Unlocking state-tracking in linear rnns through negative eigenvalues. *arXiv preprint arXiv:2411.12537*, 2024.
- [23] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.
- [24] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.
- [25] Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.
- [26] Samy Jelassi, David Brandfonbrener, Sham M Kakade, et al. Repeat after me: Transformers are better than state space models at copying. In *Forty-first International Conference on Machine Learning*, 2024.
- [27] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Židek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *nature*, 596(7873):583–589, 2021.
- [28] Jikun Kang, Wenqi Wu, Filippos Christianos, Alex J Chan, Fraser Greenlee, George Thomas, Marvin Purtorab, and Andy Toulis. Lm2: Large memory models. *arXiv preprint arXiv:2502.06049*, 2025.
- [29] Jinwoo Kim, Dat Nguyen, Seonwoo Min, Sungjun Cho, Moontae Lee, Honglak Lee, and Seunghoon Hong. Pure transformers are powerful graph learners. *Advances in Neural Information Processing Systems*, 35:14582–14595, 2022.
- [30] Aonian Li, Bangwei Gong, Bo Yang, Boji Shan, Chang Liu, Cheng Zhu, Chunhao Zhang, Congchao Guo, Da Chen, Dong Li, et al. Minimax-01: Scaling foundation models with lightning attention. *arXiv preprint arXiv:2501.08313*, 2025.
- [31] Yingcong Li, Muhammed Emrullah Ildiz, Dimitris Papailiopoulos, and Samet Oymak. Transformers as algorithms: Generalization and stability in in-context learning. In *International conference on machine learning*, pages 19565–19594. PMLR, 2023.

- [32] Zhiyuan Li, Hong Liu, Denny Zhou, and Tengyu Ma. Chain of thought empowers transformers to solve inherently serial problems. *arXiv preprint arXiv:2402.12875*, 1, 2024.
- [33] Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, Itay Dalmedigos, Erez Safahi, Shaked Meirom, Yonatan Belinkov, Shai Shalev-Shwartz, et al. Jamba: A hybrid transformer-mamba language model. *arXiv preprint arXiv:2403.19887*, 2024.
- [34] Zhixuan Lin, Evgenii Nikishin, Xu Owen He, and Aaron Courville. Forgetting transformer: Softmax attention with a forget gate. *arXiv preprint arXiv:2503.02130*, 2025.
- [35] David Lindner, János Kramár, Sebastian Farquhar, Matthew Rahtz, Tom McGrath, and Vladimir Mikulik. Tracr: Compiled transformers as a laboratory for interpretability. *Advances in Neural Information Processing Systems*, 36:37876–37899, 2023.
- [36] Jack Lindsey, Wes Gurnee, Emmanuel Ameisen, Brian Chen, Adam Pearce, Nicholas L. Turner, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, Jonathan Marcus, Michael Sklar, Adly Templeton, Trenton Bricken, Callum McDougall, Hoagy Cunningham, Thomas Henighan, Adam Jermyn, Andy Jones, Andrew Persic, Zhenyi Qi, T. Ben Thompson, Sam Zimmerman, Kelley Rivoire, Thomas Conerly, Chris Olah, and Joshua Batson. On the biology of a large language model. *Transformer Circuits Thread*, 2025.
- [37] Ashok Vardhan Makkuva, Marco Bondaschi, Adway Girish, Alliot Nagle, Hyeji Kim, Michael Gastpar, and Chanakya Ekbote. Local to global: Learning dynamics and effect of initialization for transformers. *Advances in Neural Information Processing Systems*, 37:86243–86308, 2024.
- [38] William Merrill, Jackson Petty, and Ashish Sabharwal. The illusion of state in state-space models. In *International Conference on Machine Learning*, pages 35492–35506. PMLR, 2024.
- [39] William Merrill and Ashish Sabharwal. The parallelism tradeoff: Limitations of log-precision transformers. *Transactions of the Association for Computational Linguistics*, 11:531–545, 2023.
- [40] Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*, 2018.
- [41] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernocký, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Interspeech*, volume 2, pages 1045–1048. Makuhari, 2010.
- [42] Alireza Mousavi-Hosseini, Clayton Sanford, Denny Wu, and Murat A Erdogdu. When do transformers outperform feedforward and recurrent networks? a statistical perspective. *arXiv preprint arXiv:2503.11272*, 2025.
- [43] Xueyan Niu, Bo Bai, Lei Deng, and Wei Han. Beyond scaling laws: Understanding transformer performance with associative memory. *arXiv preprint arXiv:2405.08707*, 2024.
- [44] Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, et al. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*, 2022.
- [45] Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The lambada dataset: Word prediction requiring a broad discourse context. *arXiv preprint arXiv:1606.06031*, 2016.
- [46] Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, et al. Rwkv: Reinventing rnns for the transformer era. *arXiv preprint arXiv:2305.13048*, 2023.
- [47] Bo Peng, Ruichong Zhang, Daniel Goldstein, Eric Alcaide, Haowen Hou, Janna Lu, William Merrill, Guangyu Song, Kaifeng Tan, Saiteja Utpala, et al. Rwkv-7" goose" with expressive dynamic state evolution. *arXiv preprint arXiv:2503.14456*, 2025.
- [48] DL Prados and SC Kak. Neural network capacity using delta rule. *Electronics Letters*, 25(3):197–199, 1989.

- [49] Zhen Qin, Weixuan Sun, Hui Deng, Dongxu Li, Yunshen Wei, Baohong Lv, Junjie Yan, Lingpeng Kong, and Yiran Zhong. cosformer: Rethinking softmax in attention. *arXiv preprint arXiv:2202.08791*, 2022.
- [50] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.
- [51] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. In *International conference on machine learning*, pages 28492–28518. PMLR, 2023.
- [52] Siva Reddy, Danqi Chen, and Christopher D Manning. Coqa: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266, 2019.
- [53] Liliang Ren, Yang Liu, Yadong Lu, Yelong Shen, Chen Liang, and Weizhu Chen. Samba: Simple hybrid state space models for efficient unlimited context language modeling. *arXiv preprint arXiv:2406.07522*, 2024.
- [54] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning internal representations by error propagation, 1985.
- [55] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.
- [56] Clayton Sanford, Bahare Fatemi, Ethan Hall, Anton Tsitsulin, Mehran Kazemi, Jonathan Halcrow, Bryan Perozzi, and Vahab Mirrokni. Understanding transformer reasoning capabilities via graph algorithms. *Advances in Neural Information Processing Systems*, 37:78320–78370, 2024.
- [57] Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. Socialliqa: Commonsense reasoning about social interactions. *arXiv preprint arXiv:1904.09728*, 2019.
- [58] Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *International conference on machine learning*, pages 9355–9366. PMLR, 2021.
- [59] ByteDance Seed, Yufeng Yuan, Yu Yue, Mingxuan Wang, Xiaochen Zuo, Jiaze Chen, Lin Yan, Wenyan Xu, Chi Zhang, Xin Liu, et al. Seed-thinking-v1. 5: Advancing superb reasoning models with reinforcement learning. *arXiv preprint arXiv:2504.13914*, 2025.
- [60] Darsh J Shah, Peter Rushton, Somanshu Singla, Mohit Parmar, Kurt Smith, Yash Vanjani, Ashish Vaswani, Adarsh Chaluvaram, Andrew Hojel, Andrew Ma, et al. Rethinking reflection in pre-training. *arXiv preprint arXiv:2504.04022*, 2025.
- [61] Julien Siems, Timur Carstensen, Arber Zela, Frank Hutter, Massimiliano Pontil, and Riccardo Grazi. Deltaproduct: Improving state-tracking in linear rnns via householder products. *arXiv preprint arXiv:2502.10297*, 2025.
- [62] Gregory O Stone et al. An analysis of the delta rule and the learning of statistical associations. *Parallel distributed processing: Explorations in the microstructure of cognition*, 1:444–459, 1986.
- [63] Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. Retentive network: A successor to transformer for large language models. *arXiv preprint arXiv:2307.08621*, 2023.
- [64] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. Lstm neural networks for language modeling. In *Interspeech*, volume 2012, pages 194–197, 2012.

- 484 [65] Yuandong Tian, Yiping Wang, Zhenyu Zhang, Beidi Chen, and Simon Shaolei Du. Joma:  
485 Demystifying multilayer transformers via joint dynamics of mlp and attention. In *The Twelfth*  
486 *International Conference on Learning Representations*, 2024.
- 487 [66] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*,  
488 2017.
- 489 [67] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez,  
490 Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information*  
491 *processing systems*, 30, 2017.
- 492 [68] Ke Alexander Wang, Jiaxin Shi, and Emily B Fox. Test-time regression: a unifying framework  
493 for designing sequence models with associative memory. *arXiv preprint arXiv:2501.12352*,  
494 2025.
- 495 [69] Xiaoqiang Wang, Suyuchen Wang, Yun Zhu, and Bang Liu. R<sup>3</sup>mem: Bridging memory retention  
496 and retrieval via reversible compression. *arXiv preprint arXiv:2502.15957*, 2025.
- 497 [70] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li,  
498 Chengyuan Li, Dayiheng Liu, Fei Huang, et al. Qwen2 technical report. *arXiv preprint*  
499 *arXiv:2407.10671*, 2024.
- 500 [71] Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2  
501 with delta rule. *arXiv preprint arXiv:2412.06464*, 2024.
- 502 [72] Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear  
503 transformers with the delta rule over sequence length. *arXiv preprint arXiv:2406.06484*, 2024.
- 504 [73] Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Shiji Song, and Gao Huang. Does  
505 reinforcement learning really incentivize reasoning capacity in llms beyond the base model?  
506 *arXiv preprint arXiv:2504.13837*, 2025.
- 507 [74] Chulhee Yun, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank J Reddi, and Sanjiv Kumar.  
508 Are transformers universal approximators of sequence-to-sequence functions? *arXiv preprint*  
509 *arXiv:1912.10077*, 2019.
- 510 [75] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a  
511 machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.
- 512 [76] Yufei Zhao. Probabilistic methods in combinatorics. *Draft available at <https://yufeizhao.com/pm>*, 2022.  
513

## 514 A Delta Rule with Kernel Function

515 We consider the kernel function  $\kappa(x, y) = \psi(x)^\top \psi(y)$ , where  $\psi(x)$  is a mapping from  $d$  to infinite  
 516 dimensions. Then the delta-rule-based update form and the corresponding read-out equation can be  
 517 re-write as:

$$S_t = S_{t-1}(I - \psi(k_t)\psi(k_t)^\top) + v_t\psi(k_t)^\top, \quad (11)$$

$$o_t = S_t\psi(q_t). \quad (12)$$

518 Hypothesis:

$$S_t = \sum_{i=1}^t u_i w_i^\top, \quad (13)$$

519 where  $u_i$  and  $w_i$  is pending. Then we have:

$$\sum_{i=1}^t u_i w_i^\top = \sum_{i=1}^{t-1} u_i w_i^\top (I - \psi(k_t)\psi(k_t)^\top) + v_t\psi(k_t)^\top, \quad (14)$$

$$u_t w_t^\top = \sum_{i=1}^{t-1} u_i w_i^\top (-\psi(k_t)\psi(k_t)^\top) + v_t\psi(k_t)^\top, \quad (15)$$

520 take the pending  $w_i = \psi(k_i)$ :

$$\begin{aligned} u_t \psi(k_t)^\top &= \sum_{i=1}^{t-1} u_i \psi(k_i)^\top (-\psi(k_t)\psi(k_t)^\top) + v_t\psi(k_t)^\top \\ &= -\sum_{i=1}^{t-1} \psi(k_i)^\top \psi(k_t) u_i \psi(k_t)^\top + v_t\psi(k_t)^\top \\ &= \left( -\sum_{i=1}^{t-1} \psi(k_i)^\top \psi(k_t) u_i + v_t \right) \psi(k_t)^\top. \end{aligned} \quad (16)$$

521 Thus, we get the pending  $u_t$ :

$$\begin{aligned} \cancel{u_t \psi(k_t)^\top} &= \left( -\sum_{i=1}^{t-1} \psi(k_i)^\top \psi(k_t) u_i + v_t \right) \cancel{\psi(k_t)^\top} \\ &= -\sum_{i=1}^{t-1} \kappa(k_i, k_t) u_i + v_t. \end{aligned} \quad (17)$$

522 Then we have the final update form and the corresponding read-out equation:

$$S_t = \sum_{i=1}^t u_i \psi(k_i)^\top = S_{t-1} + u_t \phi(k_t)^\top, \quad (18)$$

$$o_t = \sum_{i=1}^t \kappa(k_i, q_t) u_i. \quad (19)$$

## 523 B Proof of Theorem 1

524 Before proving Theorem 1, we introduce an auxiliary lemma for facilitating the proof.

**Lemma 1** Consider Theorem 1, the set of keys  $\{k_i\}_{i=1}^n$  satisfies Assumption 1, and  $k_{>n}$  is the difference between two keys chosen from  $\{k_i\}_{i=1}^n$ . If the function  $f(\cdot)$  satisfies Assumption 2, then the following identity holds:

$$\forall 1 \leq j < i \leq n, \forall l \geq 1: \quad f((k_i - k_j)^\top k_l) = f(k_i^\top k_l) - f(k_j^\top k_l).$$

525

526 **B.0.1 Proof of Lemma 1**

527 We distinguish two separate cases according to the value of the index  $l$ :

528 **Case 1:**  $1 \leq l \leq n$ . Consider the following subcases:

529 i. If  $k_i = k_l$ , then we obtain

$$f((k_i - k_j)^\top k_l) = f(1 - k_j^\top k_l) = f(U(1, \epsilon)) = 1$$

$$f(k_i^\top k_l) - f(k_j^\top k_l) = 1 - f(U(0, \epsilon)) = 1.$$

531 ii. If  $k_j = k_l$ , then we have

$$f((k_i - k_j)^\top k_l) = f(k_i^\top k_l - 1) = f(U(-1, \epsilon)) = -1$$

$$f(k_i^\top k_l) - f(k_j^\top k_l) = f(U(0, \epsilon)) - 1 = -1.$$

533 iii. If  $k_i \neq k_l, k_j \neq k_l$ , then

$$f((k_i - k_j)^\top k_l) = f(k_i^\top k_l - k_j^\top k_l) = f(U(0, 2\epsilon)) = 0$$

$$f(k_i^\top k_l) - f(k_j^\top k_l) = f(U(0, \epsilon)) - f(U(0, \epsilon)) = 0.$$

535 **Case 2:**  $l > n$ . In this case, denote  $k_l = k_{l_1} - k_{l_2}$ , where  $1 \leq l_2 < l_1 \leq n$ . Consider the following  
536 possibilities regarding the number of equalities among indices  $i, j$  and  $l_1, l_2$ :

537 i. If no pair among  $(i, j)$  and  $(l_1, l_2)$  is equal, then we have

$$f((k_i - k_j)^\top k_l) = f(U(0, 4\epsilon)) = 0$$

$$f(k_i^\top k_l) - f(k_j^\top k_l) = f(U(0, 2\epsilon)) - f(U(0, 2\epsilon)) = 0.$$

539 ii. If exactly one pair is equal, we analyze further:

540 1. If  $i = l_1$ , then we have

$$f((k_i - k_j)^\top k_l) = f(U(1, 3\epsilon)) = 1$$

$$f(k_i^\top k_l) - f(k_j^\top k_l) = f(U(1, \epsilon)) - f(U(0, 2\epsilon)) = 1.$$

542 2. If  $i = l_2$ , then we have

$$f((k_i - k_j)^\top k_l) = f(U(-1, 3\epsilon)) = -1$$

$$f(k_i^\top k_l) - f(k_j^\top k_l) = f(U(-1, \epsilon)) - f(U(0, 2\epsilon)) = -1.$$

544 3. If  $j = l_1$ , then similarly

$$f((k_i - k_j)^\top k_l) = f(U(-1, 3\epsilon)) = -1$$

$$f(k_i^\top k_l) - f(k_j^\top k_l) = f(U(0, 2\epsilon)) - f(U(1, \epsilon)) = -1.$$

546 4. If  $j = l_2$ , then similarly

$$f((k_i - k_j)^\top k_l) = f(U(1, 3\epsilon)) = 1$$

$$f(k_i^\top k_l) - f(k_j^\top k_l) = f(U(0, 2\epsilon)) - f(U(-1, \epsilon)) = 1.$$

548 iii. If two pairs are equal simultaneously:

549 1. If  $i = l_1, j = l_2$ , we have

$$f((k_i - k_j)^\top k_l) = f(U(2, 2\epsilon)) = 2$$

$$f(k_i^\top k_l) - f(k_j^\top k_l) = f(U(1, \epsilon)) - f(U(-1, \epsilon)) = 2.$$

551 2. If  $i = l_2, j = l_1$ , this contradicts the ordering condition  $j < i, l_2 < l_1$  and thus cannot  
552 occur.

553 Combining all the above cases, we have completed the proof.



554 **B.0.2 Formally Prove Theorem 1**

555 We use mathematical induction to prove Theorem 1.

556 When  $t = n + 1$ :

$$k_t = k_{t_1} - k_{t_2}, \quad (20)$$

$$u_t = -\sum_{i=1}^{t-1} f(k_i^\top k_t) u_i = -u_{t_1} + u_{t_2}. \quad (21)$$

557 If we read the state at  $t_1$ , i.e.,  $q_t = k_{t_1}$ ,

$$\sum_{i=1}^t f(q_t^\top k_i) u_i = \sum_{i=1}^t f(k_{t_1}^\top k_i) u_i = u_{t_1} + (-u_{t_1} + u_{t_2}) = u_{t_2}. \quad (22)$$

558 If we read the state at  $t_2$ , i.e.,  $q_t = k_{t_2}$ ,

$$\sum_{i=1}^t f(q_t^\top k_i) u_i = \sum_{i=1}^t f(k_{t_2}^\top k_i) u_i = u_{t_2} + (u_{t_1} - u_{t_2}) = u_{t_1}. \quad (23)$$

559 If we read other states, i.e., the state at  $j$ , where  $j \neq t_1, t_2$ ,

$$\sum_{i=1}^t f(q_t^\top k_i) u_i = \sum_{i=1}^t f(k_j^\top k_i) u_i = u_j. \quad (24)$$

560 In summary, at step  $t = n + 1$ , according to our rules, it is possible to trace the states exchanged  
561 between  $t_1$  and  $t_2$ .

562 Assuming the proposition holds for  $t - 1$ , we consider the case for  $t$  ( $t > n + 1$ ).

563 At the  $t$ -th step,

$$k_t = k_{t_1} - k_{t_2}. \quad (25)$$

564 According to Lemma 1, we have

$$\begin{aligned} u_t &= -\sum_{i=1}^{t-1} f(k_t^\top k_i) u_i \\ &= -\sum_{i=1}^{t-1} f(k_{t_1}^\top k_i) u_i + \sum_{i=1}^{t-1} f(k_{t_2}^\top k_i) u_i \\ &= -\tilde{v}_{t_1} + \tilde{v}_{t_2}. \end{aligned} \quad (26)$$

565 If we read the state at  $t_1$ , i.e.,  $q_t = k_{t_1}$ ,

$$\begin{aligned} \sum_{i=1}^t f(q_t^\top k_i) u_i &= \sum_{i=1}^t f(k_{t_1}^\top k_i) u_i \\ &= \sum_{i=1}^{t-1} f(k_{t_1}^\top k_i) u_i + f(k_{t_1}^\top k_t) u_t \\ &= \tilde{v}_{t_1} + (-\tilde{v}_{t_1} + \tilde{v}_{t_2}) \\ &= \tilde{v}_{t_2}. \end{aligned} \quad (27)$$

566 If we read the state at  $t_2$ , i.e.,  $q_t = k_{t_2}$ ,

$$\begin{aligned}
\sum_{i=1}^t f(q_t^\top k_i) u_i &= \sum_{i=1}^t f(k_{t_2}^\top k_i) u_i \\
&= \sum_{i=1}^{t-1} f(k_{t_2}^\top k_i) u_i + f(k_{t_2}^\top k_t) u_t \\
&= \tilde{v}_{t_2} + (\tilde{v}_{t_1} - \tilde{v}_{t_2}) \\
&= \tilde{v}_{t_1}.
\end{aligned} \tag{28}$$

567 If we read other states, i.e., the state at  $j$ , where  $j \neq t_1, t_2$ ,

$$\begin{aligned}
\sum_{i=1}^t f(q_t^\top k_i) u_i &= \sum_{i=1}^t f(k_j^\top k_i) u_i \\
&= \sum_{i=1}^{t-1} f(k_j^\top k_i) u_i + f(k_j^\top k_t) u_t \\
&= \tilde{v}_j.
\end{aligned} \tag{29}$$

568 In summary, at step  $t$ , according to our rules, the retrieved states corresponding to  $\{k_1, \dots, k_n\}$  is  
569 correct.

570 By mathematical induction, regardless of how large the exchange step  $t$  is, the model can always  
571 trace the exchange of  $n$  states.

## 572 B.1 The expression of nonlinear vs linear

573 We also conducted stress tests on the round and linear function in this Section, with  $d = 128$ . The  
574 setting is similar to Section 4.1, but with experiments where  $n$  is greater than or equal to 128. We use  
575  $n \in \{128, 256, 512\}$ , and the corresponding training length is  $\{256, 512, 1024\}$  to ensure as much as  
576 possible that most elements participate in the exchange. In addition, to avoid optimization issues, we  
577 adopted the almost orthogonal vectors used in our Theorem 1 to set key and value of the model and  
578 the model only needs to learn to read information from the state space. The results is shown in Figure  
579 7.

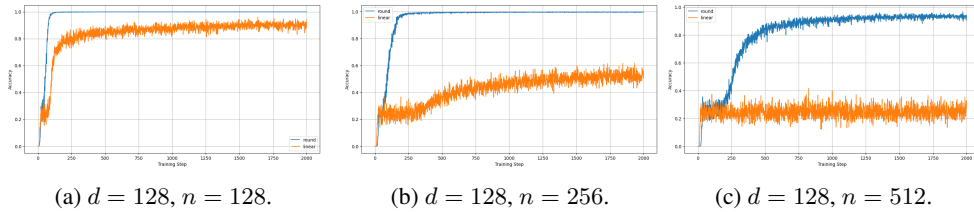


Figure 7: Comparison of DeltaFormer with  $\kappa(x, y) = x^\top y$  and  $\kappa(x, y) = \lfloor x^\top y \rfloor$ .

580 We can observe that when  $d$  is fixed, as  $n \geq d$  increases, the performance of the linear kernel is  
581 severely degraded. This essentially involves the famous Thompson problem, which is how to place  
582 as many orthogonal vectors as possible on the  $d$ -dimensional unit sphere. However linear functions  
583 cannot have superposition, and nonlinear functions can store a large amount of information through  
584 superposition [18].

## 585 C Efficient Chunk-wise Implementation

586 Below is a simple PyTorch implementation, serving as pseudo-code. We can easily modify the  
587 selection of the kernel function or remove the normalization term.

```
588 import torch
```

```

589 import torch.nn.functional as F
590 import math
591
592 def flash_attn(K_chunk, K_prev, V_prev):
593     attn = K_chunk @ K_prev.transpose(-1, -2)/math.sqrt(K_chunk.shape
594 [-1])
595     z_intra = torch.logsumexp(attn, dim=-1)
596     return torch.softmax(attn,dim=-1)@V_prev, z_intra
597
598 def naive_implementation(k, n, d_model): """n is the previous v, v is
599 actually new v. """
600     B, H, T, D = k.shape
601     v = torch.zeros_like(n)
602     for t in range(T):
603         if t == 0:
604             v[:, :, 0] = n[:, :, 0]
605         else:
606             scores = torch.matmul(k[:, :, :t], k[:, :, t].unsqueeze
607 (-1)).squeeze(-1) / math.sqrt(d_model)
608             attn_probs = F.softmax(scores, dim=-1)
609             v[:, :, t] = n[:, :, t] - torch.sum(attn_probs.unsqueeze
610 (-1) * v[:, :, :t], dim=-2)
611     return v
612
613 def optimized_chunked_implementation(K, N, d_model, C):
614     B, H, T, D = K.shape
615     V = torch.zeros(B, H, T, D)
616     chunk_nums = T // C
617     mask = torch.tril(torch.ones(C, C),diagonal=-1).unsqueeze(0).
618     unsqueeze(0).to(K.device)
619     for chunk_num in range(chunk_nums):
620         start = chunk_num * C
621         end = (chunk_num + 1) * C
622         K_chunk = K[:, :, start:end]
623         N_chunk = N[:, :, start:end]
624         if chunk_num > 0:
625             intra_output, Z_intra = flash_attn(K_chunk, K[:, :, :start
626 ], V[:, :, :, :start])#0(TCD)
627             A = (K_chunk @ K_chunk.transpose(-2, -1)).masked_fill(mask
628[:, :, :, :C, :C] == 0, float("-inf")) / math.sqrt(d_model)#0(C^2D)
629             Z_inter = torch.logsumexp(A, dim=-1)
630             P = N_chunk - intra_output * (1/(1 + torch.exp((Z_inter -
631 Z_intra).unsqueeze(-1))))
632             A = F.softmax(A, dim=-1) * (1/(1 + torch.exp((Z_intra -
633 Z_inter).unsqueeze(-1))))
634             A[:, :, 0, :] = 0
635         else:
636             A = (K_chunk @ K_chunk.transpose(-2, -1)).masked_fill(mask
637[:, :, :, :C, :C] == 0, float("-inf")) / math.sqrt(d_model)
638             A = F.softmax(A, dim=-1)
639             A[:, :, 0, :] = 0
640             P = N_chunk
641             Ti = torch.eye(C).unsqueeze(0).unsqueeze(0).unsqueeze(0).to(K.
642 device) + A
643             Ti_inverse = torch.inverse(Ti) ## Forward substitution method
644 0(C^3) Each block can be solved in parallel if we don't use the
645 normalization of softmax.
646             V[:, :, start:end] = Ti_inverse @ P # 0(C^2D)
647     return V #0(T/C * (TCD + C^2D)) = 0(T^2D + TCD + TC^2)
648
649 def verify_equivalence():
650     B = 2
651     H = 2
652     T = 1024
653     D = 64

```

```

654 C = 32
655 K = torch.randn(B, H, T, D)
656 N = torch.randn(B, H, T, D)
657 naive_output = naive_implementation(K, N, D)
658 optimized_output = optimized_chunked_implementation(K, N, D, C)
659 equivalence = torch.allclose(naive_output, optimized_output, atol
660 =1e-5)
661 print(f"{equivalence}")

```

Listing 1: PyTorch-style pseudo-code.

### 662 C.0.1 Efficient matrix inversion on GPUs.

663 In our method, the matrix inversion of a block is an important part. Previous work [72] often  
664 used a hybrid approach of forward substitution methods and block matrix inversion, but failed to  
665 fundamentally address the issue of forward substitution method not efficiently utilizing GPU parallel  
666 capabilities. To solve this problem, we studied the efficient implementation of inverse of a lower  
667 triangular matrix with diagonal 1 on GPU.

668 Assuming the matrix we are studying is  $I + A$ , where  $A$  is a strict lower triangular matrix with a  
669 diagonal of 0 and a size of  $C \times C$ . Under the setting of DeltaNet [72], this  $C$  is generally 64.

670 Then we can find that  $A^C = 0$ .

671 Then we can get:

$$(I + A)(I - A) \prod_{i=1}^{\log_2 C - 1} (I + A^{2^i}) = I - A^C = I, \quad (30)$$

672 which means

$$(I + A)^{-1} = (I - A) \prod_{i=1}^{\log_2 C - 1} (I + A^{2^i}) \quad (31)$$

673 Then we can use the following iteration to calculate  $(I + A)^{-1}$ :

$$\begin{aligned} [B_0, C_0] &= [I - A, A^2] \\ [B_i, C_i] &= [B_{i-1}, O] + [B_{i-1}, C_{i-1}]C_{i-1}, i = 1, \dots, \log_2 C - 1, \end{aligned} \quad (32)$$

674 and  $(I + A)^{-1} = B_{\log_2 C - 1}$ . Our preliminary experiments find that using this strategy can accelerate  
675 the training of DeltaNet by accelerating the calculation of DeltaNet. This method can be seen as a  
676 special case of Newton-Schulz iteration. But compared to forward substitution method, this method  
677 is not numerically stable enough, when  $C$  is large.

## 678 D The stability of the calculation of $u$ and $o$

679 We rewrite the calculations for  $u$  and  $o$  as follows:

$$\begin{aligned} u &= A_1^{-1}v \\ o &= A_2u, \end{aligned} \quad (33)$$

680 where  $A_1(i, j) = \kappa_1(k_i, k_j)$ ,  $A_2(i, j) = \kappa_2(k_i, k_j)$ .

681 Then we will have:

$$\|(A_1 + \Delta A)^{-1}V - A_1^{-1}V\| \approx \|A_1^{-1}(\Delta A)A_1^{-1}V\| \leq \|A_1^{-1}\| \|\Delta A\| \|A_1^{-1}\| \|V\| = \|A_1^{-1}\|^2 \|\Delta A\| \|V\|, \quad (34)$$

682 and

$$\|(A_2 + \Delta A_2)U - A_2U\| = \|(\Delta A_2)U\| \leq \|\Delta A_2\| \|U\|. \quad (35)$$

683 The stability of the calculation for  $u$  is weaker than that for  $o$ , so the selection of the  $\kappa_1$  need to  
684 balance stability and expressivity.

## E Code for synthetic data.

Here we provide a code for synthesizing data and the encoding of input information.

### E.1 Track the Exchange of Elements.

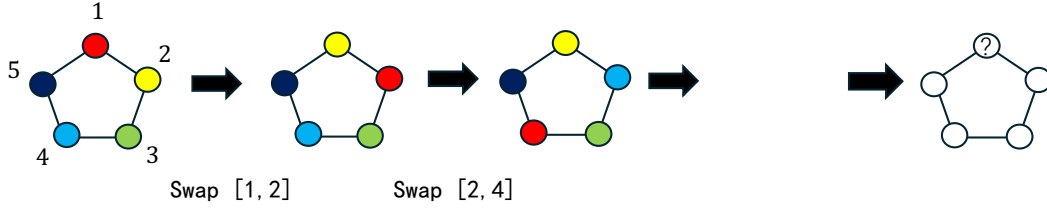


Figure 8: Swap task diagram. At the beginning, tokens of different colors are placed at positions 1 to 5, and the tokens of two positions are exchanged at each step. We expect the model to query what the token for each position is at each step. Simply but without loss of generality, we default to outputting the token at the first position to avoid introducing a "query token". This task can also be tokenized into a task with an input vocabulary size of  $C_5^2 = 10$  and an output vocabulary size of 5.

```

688 import numpy
689 import random
690 n_elements = 5
691 swap_pairs = [(i, j) for i in range(n_elements) for j in range(i+1,
692     n_elements)]
693
694 def apply_swap(perm, swap_idx):
695     i, j = swap_pairs[swap_idx]
696     perm = list(perm)
697     perm[i], perm[j] = perm[j], perm[i]
698     return tuple(perm)
699
700 def generate_data(k, num_samples):
701     data = []
702
703     for _ in range(num_samples):
704         swap_sequence = [random.randint(0, len(swap_pairs)-1) for _ in
705             range(k)]
706         current_perm = tuple(range(n_elements))
707         first_elements = []
708
709         for swap_idx in swap_sequence:
710             current_perm = apply_swap(current_perm, swap_idx)
711             first_elements.append(current_perm[0])
712
713         input_ids = torch.tensor(swap_sequence, dtype=torch.long)
714         labels = torch.tensor(first_elements, dtype=torch.long)
715         data.append((input_ids, labels))
716
717     return data

```

### E.2 Reachability of directed acyclic graphs.

```

719 import numpy
720 import random
721 import torch.nn as nn
722 def create_graph(n):
723     if n % 2 != 0:
724         raise ValueError("n should be an even number")
725
726     # Step 1: Randomly divide the points into two sets S_1 and S_2

```

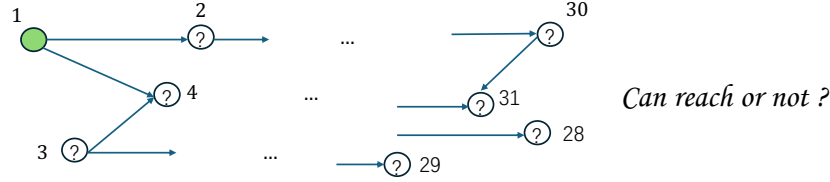


Figure 9: Reachability of directed acyclic graphs. Each node encodes at most its neighboring node information at the beginning. Then the model need to determine whether a node can reach from a starting point.

```

727 points = list(range(1, n + 1))
728 random.shuffle(points)
729 mid = n // 2
730 S_1, S_2 = sorted(points[:mid]), sorted(points[mid:])
731
732 def assign_parents(S):
733     parents = {}
734     for i in range(1, len(S)):
735         possible_parents = S[:i]
736         parents[S[i]] = random.choice(possible_parents)
737     return parents
738
739 # Step 2: Assign parent nodes within each set
740 parents_S1 = assign_parents(S_1)
741 parents_S2 = assign_parents(S_2)
742
743 # Step 3: Build adjacency matrix
744 adjacency_matrix = np.eye(n)
745 def fill_adjacency_matrix(parents):
746     for child, parent in parents.items():
747         if parent is not None:
748             adjacency_matrix[child - 1][parent - 1] = 1
749
750 fill_adjacency_matrix(parents_S1)
751 fill_adjacency_matrix(parents_S2)
752 labels = [0 for i in range(n)]
753 if 1 in S_1:
754     for i in S_1:
755         labels[i-1] = 1
756 else:
757     for i in S_2:
758         labels[i-1] = 1
759 return labels, adjacency_matrix
760
761 def generate_graph_data(num_samples=100, n=32):
762     """
763     Generates graph data samples with reachability information.
764
765     :param num_samples: Number of samples to generate.
766     :param n: Number of nodes in the graph.
767     :return: A list of tuples. Each tuple contains an adjacency matrix
768             and a list of labels indicating reachability from node 1 to each
769             node.
770     """
771     data = []
772     for _ in range(num_samples):
773         labels, A = create_graph(n)
774         adj_matrix = torch.tensor(A, dtype=torch.float)
775         # adj_matrix = adj_matrix.transpose(0,1)
776         labels = torch.tensor(labels, dtype=torch.long)
777         data.append((adj_matrix, labels))

```

```

778     return data
779 class Emb(nn.Module): #Encode the neighbor node information of each
780     node and mark the starting point
781     def __init__(self, config):
782         super().__init__()
783         self.hidden_size = config.hidden_size
784     def forward(self, x):
785         # x shape: (batch_size, seq_len, input_dim)
786         batch_size, seq_len, input_dim = x.shape
787         pos_onehot = torch.zeros(seq_len, seq_len, device=x.device)
788         pos_onehot[0, 0] = 1 # Mark the starting point
789         pos_emb = pos_onehot.unsqueeze(0).expand(batch_size, -1, -1)
790         # (batch_size, seq_len, seq_len)
791         current_dim = x.size(-1)
792         if current_dim < self.hidden_size:
793             pad_size = list(x.shape)
794             pad_size[-1] = self.hidden_size - current_dim
795             padding = torch.zeros(*pad_size, device=x.device)
796             x = torch.cat([x, padding], dim=-1) # (batch_size,
797             seq_len, hidden_size)
798         return x.to(dtype)

```

## NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes], [No], or [NA].
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

**The checklist answers are an integral part of your paper submission.** They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes]" is generally preferable to "[No]", it is perfectly acceptable to answer "[No]" provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No]" or "[NA]" is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist",
- Keep the checklist subsection headings, questions/answers and guidelines below.
- Do not modify the questions and only use the provided macros for your answers.

### 1. Claims

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

Answer: [Yes]

Justification: Yes. At the last of introduction.

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: Yes. There is a limitation section.



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: [\[Yes\]](#)

Justification: There is two assumption in main text. And the complete proof is in Appendix.

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: We use open-source code for the training of language model and offer code for toy tasks we design.

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 use open-source data. The synthetic data for the toy task can be found in the supplementary materials.

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).

954 • Providing as much information as possible in supplemental material (appended to the  
955 paper) is recommended, but including URLs to data and code is permitted.

956 **6. Experimental setting/details**

957 Question: Does the paper specify all the training and test details (e.g., data splits, hyper-  
958 parameters, how they were chosen, type of optimizer, etc.) necessary to understand the  
959 results?

960 Answer: [\[Yes\]](#)

961 Justification: We offer it in a jupyterbook in supplementary materials.

962 Guidelines:

963 • The answer NA means that the paper does not include experiments.

964 • The experimental setting should be presented in the core of the paper to a level of detail  
965 that is necessary to appreciate the results and make sense of them.

966 • The full details can be provided either with the code, in appendix, or as supplemental  
967 material.

968 **7. Experiment statistical significance**

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

971 Answer: [\[Yes\]](#)

972 Justification: : We just average over 3 runs for language modeling.

973 Guidelines:

974 • The answer NA means that the paper does not include experiments.

975 • The authors should answer "Yes" if the results are accompanied by error bars, confi-  
976 dence intervals, or statistical significance tests, at least for the experiments that support  
977 the main claims of the paper.

978 • The factors of variability that the error bars are capturing should be clearly stated (for  
979 example, train/test split, initialization, random drawing of some parameter, or overall  
980 run with given experimental conditions).

981 • The method for calculating the error bars should be explained (closed form formula,  
982 call to a library function, bootstrap, etc.)

983 • The assumptions made should be given (e.g., Normally distributed errors).

984 • It should be clear whether the error bar is the standard deviation or the standard error  
985 of the mean.

986 • It is OK to report 1-sigma error bars, but one should state it. The authors should  
987 preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis  
988 of Normality of errors is not verified.

989 • For asymmetric distributions, the authors should be careful not to show in tables or  
990 figures symmetric error bars that would yield results that are out of range (e.g. negative  
991 error rates).

992 • If error bars are reported in tables or plots, The authors should explain in the text how  
993 they were calculated and reference the corresponding figures or tables in the text.

994 **8. Experiments compute resources**

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

998 Answer: [\[Yes\]](#)

999 Justification: We offer in main text.

1000 Guidelines:

1001 • The answer NA means that the paper does not include experiments.

1002 • The paper should indicate the type of compute workers CPU or GPU, internal cluster,  
1003 or cloud provider, including relevant memory and storage.

- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

## 9. Code of ethics

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

Answer: [Yes]

Justification: We made sure to preserve anonymity.

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: We study the structure of models, which has no societal impact.

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: [No]

Justification: We use public datasets from the Internet.

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: We use CC-BY 4.0 in our code.

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](https://paperswithcode.com/datasets) has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

## 13. New assets

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

Answer: [\[Yes\]](#)

Justification: We communicate the details of the dataset/code/model at the supplementary materials.

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

1108 Guidelines:

1109 • The answer NA means that the paper does not involve crowdsourcing nor research with

1110 human subjects.

1111 • Including this information in the supplemental material is fine, but if the main contribu-

1112 tion of the paper involves human subjects, then as much detail as possible should be

1113 included in the main paper.

1114 • According to the NeurIPS Code of Ethics, workers involved in data collection, curation,

1115 or other labor should be paid at least the minimum wage in the country of the data

1116 collector.

1117 **15. Institutional review board (IRB) approvals or equivalent for research with human**

1118 **subjects**

1119 Question: Does the paper describe potential risks incurred by study participants, whether

1120 such risks were disclosed to the subjects, and whether Institutional Review Board (IRB)

1121 approvals (or an equivalent approval/review based on the requirements of your country or

1122 institution) were obtained?

1123 Answer: [NA]

1124 Justification: the paper does not involve crowdsourcing nor research with 780 human

1125 subjects.

1126 Guidelines:

1127 • The answer NA means that the paper does not involve crowdsourcing nor research with

1128 human subjects.

1129 • Depending on the country in which research is conducted, IRB approval (or equivalent)

1130 may be required for any human subjects research. If you obtained IRB approval, you

1131 should clearly state this in the paper.

1132 • We recognize that the procedures for this may vary significantly between institutions

1133 and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the

1134 guidelines for their institution.

1135 • For initial submissions, do not include any information that would break anonymity (if

1136 applicable), such as the institution conducting the review.

1137 **16. Declaration of LLM usage**

1138 Question: Does the paper describe the usage of LLMs if it is an important, original, or

1139 non-standard component of the core methods in this research? Note that if the LLM is used

1140 only for writing, editing, or formatting purposes and does not impact the core methodology,

1141 scientific rigorousness, or originality of the research, declaration is not required.

1142 Answer: [Yes]

1143 Justification: LLM is used only for writing, editing.

1144 Guidelines:

1145 • The answer NA means that the core method development in this research does not

1146 involve LLMs as any important, original, or non-standard components.

1147 • Please refer to our LLM policy (<https://neurips.cc/Conferences/2025/LLM>)

1148 for what should or should not be described.