Disentangle to Decay: Linear Attention with Trainable Decay Factor

Anonymous ACL submission

Abstract

Linear attention enhances inference efficiency of Transformer and has attracted research interests as an efficient backbone of language models. Existing linear attention based models usually exploit decay factor based positional encoding (PE), where attention scores decay exponentially with increasing relative distance. However, most work manually designs a non-trainable base of exponential calculation (decay factor), which limits further optimization. Our analysis reveals that direct training decay factor is unstable. To address this problem, we propose a novel PE for linear attention named Disentangle to Decay (D2D). D2D disentangles decay factor into two parts to achieve further optimization and stable training. Moreover, D2D can be transformed into recurrent form for efficient inference. Experiments demonstrate that D2D achieves stable training of decay factor, and enhances performance of linear attention in both normal context length and length extrapolation scenarios [1].

1 Introduction

Linear attention, by substituting the softmax function in vanilla Transformers with a dot-product of kernel feature maps, achieves linear complexity during inference and is particularly advantageous for processing long sequences (Katharopoulos et al., 2020). However, challenges such as cumulative regularity errors over long sequences necessitate specialized mechanisms for effective information filtering (Qin et al., 2022a). For existing language models based on linear attention, such as RetNet (Sun et al., 2023) and TransNormer-LLM (Qin et al., 2024), their PEs include decay terms $\gamma^{(i-j)}$, where γ is the **decay factor** and $i-j$ represents the relative position between two tokens. Decay factor provides a mechanism for information forgetting, which can alleviate the aforementioned issue and enhance the capability of linear attention in handling long sequences.

However, decay factors used in these models are manually designed and non-trainable, as a limit to further optimization with model and dataset (Moreno-Cartagena et al., 2023). We reveal that directly training decay factor might generate significantly large gradients, due to exponential calculation with a trainable base. Consequently, large gradients integrate numeric instability and leads trainings to collapse. Models fail to yield satisfactory outcomes from a trainable decay factor.

To enhance stability of training and performance of models, our work proposes an innovative trainable decay factor based PE named **D**isentangle to **D**ecay (**D2D**). D2D disentangles decay factor into two parts. **Global decay factor** is fixed and provides base value. With an effective initialization, it provides numeric foundation for trainable decay factor. Moreover, it is initialized to certain range to generate an item mitigate large gradients into an acceptable range. **Local tuning factor** is trainable for further optimization of performance, which is stable with integration of fixed global decay factor. In implementation, we separate two parts of decay factor in calculation. Consequently, D2D is represented as a combination of absolute positional encoding (APE) and relative positional encoding (RPE). This form can also avoid unnecessary calculation and address overflow problem for large positional indices.

We pretrain language models using D2D and other types of decay factors, with a similar scale to GPT-2 (Radford et al., 2019). Then we conduct various experiments on language modeling, length extrapolation and downstream tasks. The results show that D2D enables linear attention to achieve better performance compared with both directly trained decay factors and fixed decay factors. Additionally, D2D shows greater numerical stability

[1] Our code implementation is available at: https://anonymous.4open.science/r/D2D-0CF7/

during training. D2D outperforms existing PE, including RoPE (Su et al., 2024) and ALiBi (Press et al., 2022). We also provide an implementation of the transformation for recurrent inference and conduct experiments on inference speed, indicating that D2D is both spatially and temporally efficient.

Our main contributions are as follows:

1. We analyze existing decay based PE methods for linear attention from the perspective of gradients, investigating how numerical instability during the training process leads to either training failure or suboptimal results.
2. We propose D2D, a novel trainable PE method for linear attention. D2D maintains stability in training and enhances representational capability of decay based PE. Moreover, we provide implementation of D2D for efficient training and inference.
3. We conduct experiments of language modeling, length extrapolation and downstream tasks for D2D based language models. Results show that D2D is more stable and outperforms other types of decay factors as well as existing PE on the aforementioned tasks.

2 Preliminary

2.1 Computational Form of Linear Attention

For two tokens with positions i and j, let Q_i and K_j represent query and key respectively. According to Katharopoulos et al. (2020), unified formulation of linear and vanilla attention is given by Eq. 1, where the similarity calculation $Sim(Q_i, K_j)$ quantifies relationship between the query of the i-th token and the key of the j-th token:

$$Att_{i,j} = \frac{Sim(Q_i, K_j)}{\sum_{k=1}^{i} Sim(Q_i, K_k)} \quad (1)$$

In vanilla attention, the similarity is calculated using the exponent of the dot product of query and key, expressed as $Sim(Q_i, K_j) = \exp(Q_i K_j^{\intercal})$. Conversely, in linear attention, the similarity is computed directly through a kernel function ϕ, leading to a similarity measure $Sim(Q_i, K_j) = \phi(Q_i)\phi(K_j)^{\intercal}$.

2.2 Constraints of PE in Linear Attention

Compared with vanilla attention, PE used in linear attention must satisfy certain constraints. To enhance computational efficiency during inference, it is necessary to transform linear attention into RNN (Katharopoulos et al., 2020). This transformation is contingent upon a specific positional encoding format, as detailed in Eq. 2:

$$Sim(Q_i, K_j) = f_q(Q_i, i) \cdot f_k(K_j, j) \quad (2)$$

where f_q and f_k are functions applied to Q_i and K_j, respectively, to incorporate absolute positional information. To be more detailed, by this equation, the similarity calculation between queries and keys is decomposed into independent functions that are completely dependent on the queries and keys. The detailed proof process for this constraint is provided in Appendix. A.4.

3 Instability of Training Decay Factor

Numerical Instability For most decay factor based PEs, decay factors are set as fixed number instead of a trainable parameter, since they do not achieve better performance (Press et al., 2022; Sun et al., 2023). In this section, we analyze training of decay factor exhibits **numerical instability**, leading to training collapse and limited optimization.

More specifically, the value of decay factor exhibits significant fluctuations throughout the training process and fails to converge rapidly to a stable value. When the decay factor reaches a certain threshold, it tends to trigger gradient explosion, causing the training to collapse.

Large Gradients Brought By Exponential Calculation The attention calculation involves higher-order terms of decay factor, which can generate large gradients and lead to unstable gradient descent. For two tokens separated by a relative distance of δ, a higher-order term γ^{δ} is adopted in calculation (Qin et al., 2024; Sun et al., 2023), where γ is the decay factor. When γ becomes trainable, it generates gradient of $\frac{d(\gamma^{\delta})}{d\delta} = \delta\gamma^{\delta-1}$. This will cause instability while training as analyzed below.

When the range of δ increases, the gradient produced by the global decay factor can potentially reach a very large value. For instance, GPT-2 [2] has a context length of 1024. Taking $\gamma = 0.999$ as an example, the gradient of decay factor reaches a summit value of 376. This gradient acts as a coefficient while calculating attention score, which enlarges the overall training gradients significantly. Large gradients result in instability during training.

[2]https://huggingface.co/openai-community/gpt2/blob/main/config.json

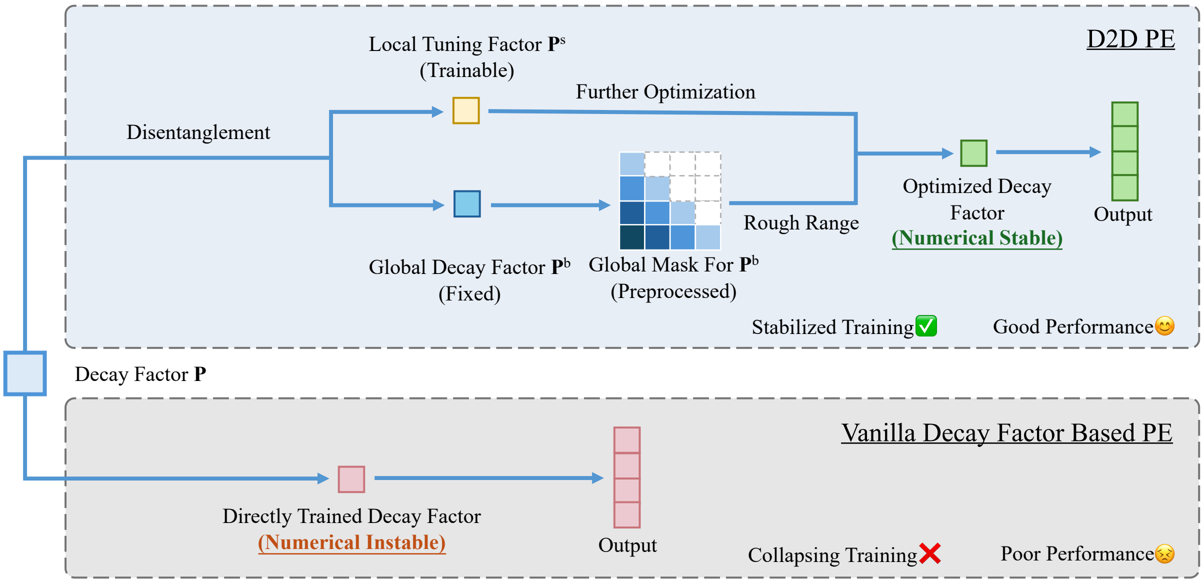

Figure 1: An overview of D2D and vanilla decay factor based PE during training. Firstly, D2D disentangles decay factor to global decay factor and local tuning factor to implement PE design. Then, D2D provides fixed global decay factor for rough range, and trains local tuning factor for detailed optimization. By exploiting sum of them, a well-optimized decay factor can be exploited for stable training and good performance. Additionally, we preprocess value of global decay factor for every position in training length, in order to enhance training efficiency.

Moreover, linear attention has limited performance compared with vanilla attention due to large gradients (Qin et al., 2022a). Large gradients produced by decay factor will further amplify unstable gradients produced by linear attention. In subsequent experiments, we observe collapse of training and poor performance via directly trained decay factor, which valids our analysis.

In summary, such unexpected gradients emphasize the sensitivity of the attention mechanism. It is necessary to stabilize training of decay factor and develop a more efficient decay factor based PE.

4 Method

Here, we propose D2D, an effective solution to address the instability of decay factor during training. Main process of D2D compared with vanilla decay factor based PE is shown in Fig. 1.

4.1 Disentanglement based Positional Encoding

Disentanglement of Decay Factor Firstly, we provide **detailed assignment within attention head** for decay factor. For l-th attention head, decay factor is described as a vector $\boldsymbol{P_l} \in \mathbb{R}^{1\times d_h}$, where d_h is dimension of each attention head. For comparison, existing method exploit a constant scalar γ_l within the attention head.

Secondly, we disentangle value of decay factor into two parts, **global decay factor** and **local tuning factor** to achieve value of decay factor more detailed. Global decay factor $\boldsymbol{P^b}$ is applied to each attention head, providing a rough range for decay factor. For the l-th attention head, global decay factor has a value of $\boldsymbol{P_l^b} \in \mathbb{R}^{1\times d_h}$, where $\boldsymbol{P_l^b} = (p_l^b, \dots, p_l^b)$ is composed of a series of fixed scalars p_l^b. Local tuning factor $\boldsymbol{P_l^s} \in \mathbb{R}^{1\times d_h}$ is applied to each dimension of the attention head to achieve fine-grained optimization of decay factor. For the l-th attention head, vector $\boldsymbol{P_l}$ is disentangled, that is $\boldsymbol{P_l} = \boldsymbol{P_l^b} + \boldsymbol{P_l^s}$. As shown in Fig. 2, possible sum of two factors takes up a wider range of distribution, which is benefit for optimization.

Positional Encoding Design On the basis of aforementioned disentanglement of decay factor $\boldsymbol{P_l}$, we rewrite the form of decay factor as Eq. 3, where $Sim(Q_i, K_j)[l]$ represents the similarity calculation for the l-th attention head, with all divisions performed element-wise. In Eq. 3, undivided calculation (first line of Eq. 3) and Θ_s use APE form, while Θ_b uses the RPE form.

For training, calculation of D2D is transformed into $\Theta_b \cdot \Theta_s$. This transformation is key to improving training stability and provides foundation for the discussion on stable training methods in Sec. 4.2. We implement an efficient training approach in Sec. 4.3 for detailed discussion.

$$
\begin{aligned}
Sim(Q_i, K_j)[l] &= \frac{\phi(Q_i)}{\exp(i\boldsymbol{P_l})}(\frac{\phi(K_j)}{\exp(-j\boldsymbol{P_l})})^\intercal \\
&= \Theta_b \cdot \Theta_s \\
\Theta_b &= \exp{(-p_l^b)}^{i-j} \\
\Theta_s &= \frac{\phi(Q_i)}{\exp(i\boldsymbol{P_l^s})}(\frac{\phi(K_j)}{\exp(-j\boldsymbol{P_l^s})})^\intercal
\end{aligned} \tag{3}
$$

For inference, numeric instability does not need to be concerned since all parameters of D2D are fixed during inference. We exploit first line of Eq. 3 for effective recurrent inference, where value of $\boldsymbol{P_l}$ is derived from $\boldsymbol{P_l^b}+\boldsymbol{P_l^s}$. It satisfies the constraints of converting linear attention into RNN as described in Sec. 2.2, so D2D is available for recurrent inference. More specifically, we can transform the linear attention calculation using D2D into the following expression as described in Katharopoulos et al. (2020):

$$
\begin{aligned}
V_i' &= \frac{\sum_{j=1}^{i}(\phi(Q_i)\exp(-i\boldsymbol{P_l}))(\phi(K_j)\exp(j\boldsymbol{P_l}))^\intercal V_j}{\sum_{j=1}^{i}(\phi(Q_i)\exp(-i\boldsymbol{P_l}))(\phi(K_j)\exp(j\boldsymbol{P_l}))^\intercal} \\
&= \frac{\phi(Q_i)(S_{i-1}\exp(-\boldsymbol{P_l}) + \phi(K_i)^\intercal V_i)}{\phi(Q_i)(Z_{i-1}\exp(-\boldsymbol{P_l}) + \phi(K_i)^\intercal)} \\
S_i &= S_{i-1}\exp(-\boldsymbol{P_l}) + \phi(K_i)^\intercal V_i \\
Z_i &= Z_{i-1}\exp(-\boldsymbol{P_l}) + \phi(K_j)^\intercal
\end{aligned} \tag{4}
$$

Eq. 4 is derived from Eq. 3 and is used in the inference process. In Eq. 4, V_i' is the output of the attention, $S_0 \in \mathbb{R}^{d_h \times d_h}$, $Z_0 \in \mathbb{R}^{1 \times d_h}$. All elements in S_0 and Z_0 are zero. More details of converting linear attention into RNN are shown in Appendix A.3.

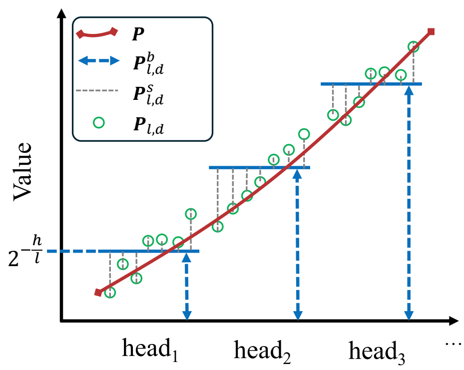

Figure 2: Illustration of disentanglement. Green circle stands for each index of $\boldsymbol{P}$ is sum of fixed $\boldsymbol{P^b}$ and trainable $\boldsymbol{P^s}$. To visualize the value of $\boldsymbol{P}$, we approximate it with a smooth red curve on the Figure. Possible sum of them could cover a wide range during optimization. In the legend, $\boldsymbol{P_{l,d}}$ represents the value of $\boldsymbol{P}$ at dimension d in the l-th head.

4.2 Stabilizing Training

Effective Initialization for D2D An effective initialization strategy can provide optimal foundation for PE (Press et al., 2022). Compared with random initialization or zero initialization, it provides a more structured initialization, facilitating faster convergence and better overall model performance.

Following (Press et al., 2022), we initialize global decay factor p_l^b as $2^{-\frac{h}{l}}$ for l-th attention head, where h is the total amount of attention head. This ensures an increasing density of values as they approach zero, facilitating a more nuanced representation of positional information. And we apply zero initialization for local tuning factor, aims at optimizing global decay factor in fine-granularity.

In D2D, global decay factor provides a foundation for the training of local tuning factor. Once global decay factor in each attention head is preset to an appropriate value, range of $\boldsymbol{P_l^s}$ during gradient descent is narrowed and simultaneously enhances stability during training. Subsequent experiments in Sec. 5.4 validate the above analysis.

Stabilizing Gradients of Decay Factor As shown in Sec. 3, training of decay factor generates large gradients, which is the main reason for training instability. The global decay factor in D2D can reduce the gradients during training to an acceptable range.

After adding the global decay factor, the absolute value of gradient produced by the local tuning factor is $\delta\exp(-p_l^b)^\delta \exp(-\boldsymbol{P_l^s})^\delta$. Compared with the gradient produced by directly trained decay factor, gradient of D2D has an extra coefficient $\exp(-p_l^b)^\delta$, where $\exp(-p_l^b) < 1$. This item decreases with the growth of δ, mitigating large gradients brought by δ. In practical training in first attention head of 8, global decay factor can generate a coefficient of 0.018 and reduce the gradient in Sec. 3 from 376 to 6.87.

4.3 Mask-based Efficient Training Implementation

Extra Time Cost on Calculating Θ_b As shown in Eq. 3, Θ_b needs to be calculated every time in similarity calculation of $Sim(Q_i, K_j)$. But Θ_b is only determined by positional indices i, j, resulting unnecessary exponential calculations. This problem also exists when directly training decay factor.

Precision Problems In Calculation During training phrase, calculation in the first line of Eq. 3

encounters preicision problems of floating numbers. For decay factor based PE, $\exp(\gamma i) \cdot \exp(-\gamma j) = \exp(\gamma(i-j))$ should hold for all position indices. However, the exponential calculation overflows when i is very large and approaches zero when j is very large. As a result, the product does not match the theoretical value during training. This causes the value of D2D, which is only related to relative positions, to be affected by absolute positions of tokens. Consequently, optimization of decay factor might be truncated to certain value. Moreover, precision problems limit application for longer sequences as a result of larger positional indices. Therefore, it is necessary to avoid direct computation of $\exp(\gamma i)$ and $\exp(-\gamma j)$ in APE form. Instead, computing $\exp(\gamma(i-j))$ in RPE form can help mitigate the impact of precision issues.

Masked-based Transformation For Eq. 3, $\boldsymbol{P}_l^b$ consists of identical scalars p_l^b. Therefore, the global decay factor can be factored out as Θ_b. It is constant across all computations within a head. Consequently, when context length is given, all possible results of relative positions can be preprocessed before training. We implement this by presetting a mask M as shown in Fig. 3. The element in the i-th row and j-th column of the matrix corresponds $M_{i,j}$. The part where $j > i$ is assigned a value of 0 to ensure attention is unidirectional in autoregressive language modeling. During training, positional information for i-th query and j-th key should multiply $M_{i,j}$ to integrate global decay factor. For different attention heads, we preprocess matrices respectively, since number of attention heads is usually limited. Regarding the precision problem, we convert the global decay factor with larger values into a preprocessed mask, and the calculation of this mask only involves RPE. The remaining local tuning factor has smaller values and does not cause significant precision problems.

To integrate this mask, we apply element-wise product of mask and attention scores. In implementation, we replace *causal mask* [3] with M to save time and space cost.

$$\begin{matrix} 1 & 0 & 0 & 0 \\ p_b & 1 & 0 & 0 \\ {p_b}^2 & p_b & 1 & 0 \\ {p_b}^3 & {p_b}^2 & p_b & 1 \end{matrix}$$

Figure 3: An instance of decay mask (length $n = 4$).

Overall Training And Inference Implementation During the training and inference phases, the key difference lies in the introduction of the computation process for the D2D attention output, while the remaining steps follow those described in Katharopoulos et al. (2020). Algorithm 1 and Algorithm 2 respectively illustrate the process of incorporating the D2D attention output in training and inference. $\div$ stands for element-wise division, and $\odot$ stands for element-wise multiply. In the algorithm, the operations *splithead* and *mergehead* refer to the processes used in the multi-head attention mechanism (Vaswani et al., 2017).

Algorithm 1 Attention Output During Training

procedure ATTN($Q, K, V, M, \boldsymbol{P}^s, n$)
 $K \leftarrow K^\intercal$
 $Q, K \leftarrow \phi(Q), \phi(K)$
 $\boldsymbol{a} \leftarrow (0, 1, \ldots, n-1)$
 $C \leftarrow \exp(\boldsymbol{a} \cdot \boldsymbol{P}^s)$
 $Q \leftarrow Q \div C$
 $K \leftarrow K \odot C$
 $Q, K, V \leftarrow \text{splithead}(Q, K, V)$
 $Att \leftarrow Q \cdot K \odot M$
 for $i \leftarrow 0$, **to** $n-1$ **do**
  $Att_i \leftarrow Att_i / \sum_{j=0}^{n-1}(Att_{i,j})$
 end for
 $O \leftarrow Att \cdot V$
 $O \leftarrow \text{mergehead}(O)$
 return O
end procedure

5 Experiments

In this section, we apply D2D and linear attention into vanilla Transformer. We conduct experiments on **language modeling**, **length extrapolation** and several **downstream tasks** after finetuning. Experiment result validates effectiveness of D2D for encoding positional information. Moreover, we provide an implementation to transform linear attention based on D2D to RNN in Appendix. A.7. Result shows that D2D is efficient during inference.

[3] For autoregressive language models, causal mask is a lower triangular matrix to ensure attention is unidirectional.

	Language Modeling					Length Extrapolation				Downstream Tasks	
Datasets	enwiki8 (PPL↓)	LAMBADA (PPL↓)		WikiText2 (PPL↓)		GovReport (PPL↓)		PG19 (PPL↓)		ARC-e (ACC↑)	ARC-c (ACC↑)
Finetune	w/o	w/o	w/	w/o	w/	w/o	w/	w/o	w/	w/	w/
Methods											
Fixed.	94.91	95.06	31.53	96.29	18.57	24.14	16.78	198.53	40.52	0.250	0.218
Trained.	92.27	**89.01**	29.65	85.07	18.40	22.77	16.69	174.81	34.38	0.251	0.234
D2D	**86.36**	90.63	**25.83**	**72.48**	**18.29**	**21.25**	**15.97**	**169.99**	**29.76**	**0.262**	**0.256**

Table 1: The results of testing D2D, fixed decay factor, and directly factor on various tasks. *Fixed.* represents linear attention using fixed decay factor, *Trained.* represents linear attention using directly trained decay factor and *D2D* represents linear attention using D2D. *w/o* represents direct testing on the dataset, while *w* indicates testing after finetuning on the corresponding training set. The best results for each task are bold.

Algorithm 2 Attention Output During Inference

procedure ATTN($Q, K, V, \boldsymbol{P}^b, \boldsymbol{P}^s, n$)
  $K \leftarrow K^\intercal$
  $\boldsymbol{P} \leftarrow \boldsymbol{P}^b + \boldsymbol{P}^s$
  $\boldsymbol{P} \leftarrow \exp(\boldsymbol{P})$
  $\boldsymbol{S}, \boldsymbol{Z} \leftarrow \mathbf{0}_{d_h \times d_h}, \mathbf{0}_{d_h \times 1}$
  $Q, K, V \leftarrow \text{splithead}\,(Q, K, V)$
  for $i \leftarrow 0$ **to** $n-1$ **do**
    $Q_i, K_i \leftarrow \phi(Q_i), \phi(K_i)$
    $\boldsymbol{S} \leftarrow \boldsymbol{S} \odot \boldsymbol{P} + K_i \cdot V_i$
    $\boldsymbol{Z} \leftarrow \boldsymbol{Z} \odot \boldsymbol{P} + K_i$
    $O_i \leftarrow (Q_i \cdot \boldsymbol{S})/(Q_i \cdot \boldsymbol{Z})$
  end for
  $O \leftarrow concat(O_1, \ldots, O_n)$
  $O \leftarrow \text{mergehead}\,(O)$
  return O
end procedure

5.1 Experiment Settings

We select GPT-2 (Brown et al., 2020) as backbone of autoregressive language models. We select $elu(x) + 1$ (Clevert et al., 2016) as kernel function and pretrain models on OpenWebText (Gokaslan and Cohen, 2019) datasets. Likewise, we use a dataset and number of training steps similar to (Radford et al., 2019). Dataset statistics and more details can be found in Appendix A.5.

For comparison, we pretrain two models with fixed decay factor and directly trained decay factor respectively. For downstream tasks, we involve 1 epochs of finetuning after pretrain.

5.2 Experiment Results

5.2.1 Language Modelling

Following (Radford et al., 2019), we evaluate the capabilities of D2D in language modeling on enwiki8 [4], LAMBADA (Paperno et al., 2016) and WikiText2 (Merity et al., 2016). As shown in Table 1, the model exhibits good language modeling performance with D2D, resulting from improved positional information. On LAMBADA dataset, D2D and the directly trained decay factor yield similar results before finetuning. After finetuning, D2D achieves better performance.

5.2.2 Length Extrapolation

It is crucial that the PE we design has sufficient length extrapolation capability to fully leverage the benefits of linear attention. Following (Press et al., 2022), we conduct experiments in pretrained domain and outside pretrained domain.

In-domain Length Extrapolation We conduct language modeling task in training set of OpenWebText for longer context length than trained. D2D achieves better results compared with both the fixed decay factor and the directly trained decay factor.

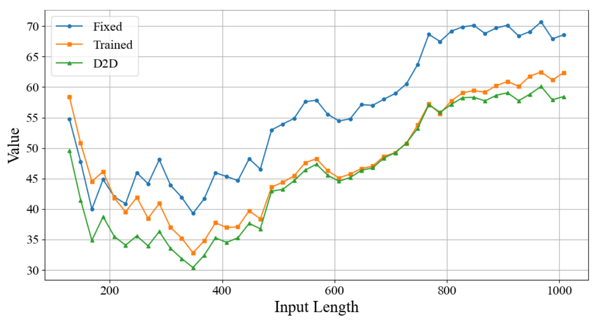

Figure 4: The figure illustrates the model's ability to extrapolate the length within the domain. As the length increases, the model using the decay factor initially shows a decreasing trend in PPL, followed by an increase, and eventually stabilizes.

[4] http://mattmahoney.net/dc/text.html

PE	Vanilla APE Linear Attention	Vanilla APE and Attention	RoPE	ALiBi	D2D
PPL(Train)	49.40	45.74	44.59	44.88	**43.82**
PPL(Valid)	50.86	47.66	47.8	47.85	**46.9**

Table 2: PPL on training and validation dataset, lower PPL shows better performance. Values bold are denoted as optimal results.

Out-of-domain Length Extrapolation Following (Rae et al., 2020; Dong et al., 2024), we test length extrapolation on GovReport (Huang et al., 2021) and PG19 (Rae et al., 2019). Firstly, we fine-tune models in normal context length and then conduct language modeling on longer context length. Results are shown in Table. 1, D2D performs better compared to both the fixed decay factor and the directly trained decay factor.

5.2.3 Downstream Task

To verify the impact of D2D on linear attention in terms of reasoning capabilities and language understanding capabilities, we conduct downstream task tests on ARC-e and ARC-c (Clark et al., 2018). As shown in Table. 1, linear attention using D2D outperforms those that use a fixed decay factor or a directly trained decay factor.

5.3 Comparing with Other Positional Encoding

To compare the performance of D2D with other commonly used PE in linear attention, we train combinations of various PEs with linear attention as well as vanilla Transformers on the first 10% of OpenWebtext dataset. We selected RoPE (Su et al., 2024), ALiBi (Press et al., 2022), Vanilla APE (Vaswani et al., 2017), and vanilla attention as baselines to compare with D2D. Detailed information of other PE is shown in Appendix. A.6. As shown in Table. 2, D2D achieves the best results on both the training and validation datasets.

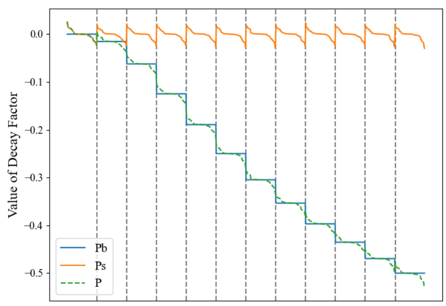

Figure 5: The value of decay factor in the first layer of D2D based linear attention model. To enhance image clarity, we use vertical gray dashed lines to split heads and sort P^s within each head.

5.4 Numerical Stability During Training

To verify the numerical stability of D2D, we train both a linear attention with D2D and directly trained decay factor. We compare their stability by observing the numerical changes in the trainable parts of the PE and the final training outcomes.

As shown in Fig. 5, the values of $\boldsymbol{P^s}$ in D2D are smaller compared to $\boldsymbol{P^b}$, primarily serving to adjust the decay factor within each head. Fig. 6 provides a more intuitive illustration of the value changes in the decay factor for both D2D and directly trained methods during the training process. Compared with directly trained decay factor, the stability of D2D during training is significantly higher.

In Sec. 4.3, we discuss the issue of not converting Θ_b into a mask. To address this, we directly

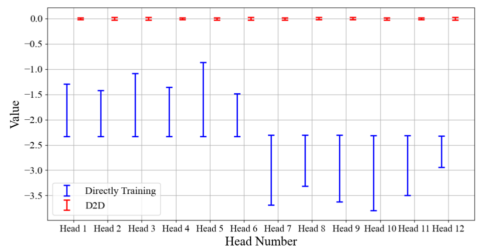

Figure 6: The numerical fluctuations of the D2D and directly trained decay factor from the first layer of linear attention model during training process.

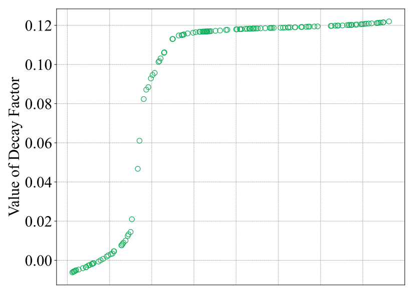

Figure 7: The results of directly training D2D without converting $\boldsymbol{P^b}$ into mask. The image displays the values of $\boldsymbol{P}$ in the first layer of the model.

train a linear attention model using D2D without any transformations. As shown in Fig. 7, the value of $\boldsymbol{P}$ gets truncated near a certain threshold, making it difficult for the D2D to further change after reaching this value. This indicates that the problem mentioned in Sec. 4.3 significantly impacts training, limiting the range of values for the D2D.

6 Analysis

In this section, we analyze improvements of D2D in three aspects of experiments. Improvements in **language modeling** can be attributed to the stable training of decay factor and appropriate range of global decay factor. They provide a more reasonable decay factor to represent more information. Regarding **length extrapolation**, we believe that the decay factor inherently possesses significant length extrapolation capabilities (Press et al., 2022). D2D enlarges such advantages with its stronger representation capabilities. For **downstream tasks**, the primary advantage of D2D lies in the optimization of local tuning factor. Fig. 5 illustrates that in the first head, the $\boldsymbol{P}^{\boldsymbol{b}} + \boldsymbol{P}^{\boldsymbol{s}}$ values are negative in certain dimensions, indicating these dimensions focus on tokens that are farther apart. This capability is not present in models with fixed decay factors or models with directly trained decay.

7 Related Work

7.1 Linear Attention

Linear attention enhances computational efficiency by reducing the space-time complexity from quadratic to linear. It can be roughly categorized into kernel-based methods and random-based methods. Kernel-based linear attentions (Qin et al., 2022b; Katharopoulos et al., 2020; Qin et al., 2022a) process query and key with kernel functions. Random-based linear attentions (Peng et al., 2021; Choromanski et al., 2021) fit expected value through random sampling methods.

A notable advancement is the transformation of linear attention into a recurrent neural network form, as explored by Katharopoulos et al. (2020) and further applied in large-scale models by (Yang et al., 2023; Sun et al., 2023). These approaches allow for both parallel and serial processing, improving scalability and efficiency.

7.2 Positional Encoding

Positional encoding integrates positional information into the Transformer model, which is essential for sequence recognition and computational efficiency, especially with long sequences and large models (Kazemnejad et al., 2023).

Types of Positional Encoding PE can be categorized into APE, RPE, and convertible positional encoding, each serving distinct roles within model's architecture. APE uses absolute positions, utilizing trigonometric functions or trainable parameters (Vaswani et al., 2017; Brown et al., 2020; Zhang et al., 2022). RPE accounts for relative distances between tokens with approaches like RoPE or ALiBi (Su et al., 2024; Press et al., 2022), which are common in large language models (Raffel et al., 2020; Chowdhery et al., 2023; Scao et al., 2022). Convertible Positional Encoding allows switching between APE and RPE, facilitating flexible computational strategies (Su et al., 2024).

Decay Factor Commonly used RPEs such as RoPE (Su et al., 2024), ALiBi (Press et al., 2022), and XPos (Sun et al., 2022) exhibit certain decay properties. Specifically, during the computation of attention scores, these PEs cause the model to focus more on tokens that are closer in proximity. This enhances the model's focus during the calculation of attention scores, thereby improving its language modeling capabilities (Han et al., 2023).

For linear attention models (Sun et al., 2023; Qin et al., 2024). They incorporate decay terms in form of $\gamma^{(i-j)}$, where γ is the **decay factor** and $i - j$ denotes the relative positions.

8 Conclusion

In this paper, we design a positional encoding method, D2D, for models based on linear attention. By analyzing the conditions under which linear attention can be transformed into RNN, we ascertain that D2D needs to facilitate the conversion between absolute and relative positional encoding. Leveraging this characteristic, we disentangle D2D during the training process, transforming it into a combination of APE and RPE to enhance training stability. In the inference process, we fully convert D2D into APE, enabling the transformation of linear attention into an RNN form. This fully leverages the advantages of linear attention in terms of time complexity and space complexity during the inference process. Models utilizing D2D linear attention have demonstrated commendable performance in language modeling and length extrapolation.

9 Limitation

Our positional encoding demonstrates effectiveness across various kernel functions, though the extent of the effect is somewhat dependent on the choice of kernel function. Based on our experiments, we find that $elu(x) + 1$ is a good choice for the kernel function, but we cannot provide a very systematic theoretical explanation for this choice. Additionally, although we have conducted some analysis on the instability of the decay factor both experimentally and theoretically, we have not provided a comprehensive mathematical proof. Moreover, application of D2D has not been extended to large language models.

References

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853.

Galen Andrew and Jianfeng Gao. 2007. Scalable training of L1-regularized log-linear models. In *Proceedings of the 24th International Conference on Machine Learning*, pages 33–40.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamás Sarlós, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J. Colwell, and Adrian Weller. 2021. Rethinking attention with performers. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

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

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2016. Fast and accurate deep network learning by exponential linear units (elus). In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Zican Dong, Junyi Li, Xin Men, Wayne Xin Zhao, Bingbing Wang, Zhen Tian, Weipeng Chen, and Ji-Rong Wen. 2024. Exploring context window of large language models via decomposed positional vectors.

Aaron Gokaslan and Vanya Cohen. 2019. Openwebtext corpus. `http://Skylion007.github.io/OpenWebTextCorpus`.

Dongchen Han, Xuran Pan, Yizeng Han, Shiji Song, and Gao Huang. 2023. Flatten transformer: Vision transformer using focused linear attention. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5961–5971.

Luyang Huang, Shuyang Cao, Nikolaus Parulian, Heng Ji, and Lu Wang. 2021. Efficient attentions for long document summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1419–1436, Online. Association for Computational Linguistics.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. 2020. Transformers are rnns: Fast autoregressive transformers with linear attention. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org.

Amirhossein Kazemnejad, Inkit Padhi, Karthikeyan Natesan, Payel Das, and Siva Reddy. 2023. The impact of positional encoding on length generalization in transformers. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models.

Daniel Moreno-Cartagena, Guillermo Cabrera-Vives, Pavlos Protopapas, Cristobal Donoso-Oliva, Manuel Pérez-Carrasco, and Martina Cádiz-Leyton. 2023. Positional encodings for light curve transformers: Playing with positions and attention.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The lambada dataset: Word prediction requiring a broad discourse context.

Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah A. Smith, and Lingpeng Kong. 2021. Random feature attention. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Ofir Press, Noah A. Smith, and Mike Lewis. 2022. Train short, test long: Attention with linear biases enables input length extrapolation. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Zhen Qin, Xiaodong Han, Weixuan Sun, Dongxu Li, Lingpeng Kong, Nick Barnes, and Yiran Zhong. 2022a. The devil in linear transformer. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 7025–7041, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Zhen Qin, Dong Li, Weigao Sun, Weixuan Sun, Xuyang Shen, Xiaodong Han, Yunshen Wei, Baohong Lv, Xiao Luo, Yu Qiao, and Yiran Zhong. 2024. Transnormerllm: A faster and better large language model with improved transnormer.

Zhen Qin, Weixuan Sun, Hui Deng, Dongxu Li, Yunshen Wei, Baohong Lv, Junjie Yan, Lingpeng Kong, and Yiran Zhong. 2022b. cosformer: Rethinking softmax in attention. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

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

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, and Timothy P. Lillicrap. 2019. Compressive transformers for long-range sequence modelling.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(1).

Mohammad Sadegh Rasooli and Joel R. Tetreault. 2015. Yara parser: A fast and accurate dependency parser. *Computing Research Repository*, arXiv:1503.06733. Version 2.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilic, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, Jonathan Tow, Alexander M. Rush, Stella Biderman, Albert Webson, Pawan Sasanka Ammanamanchi, Thomas Wang, Benoît Sagot, Niklas Muennighoff, Albert Villanova del Moral, Olatunji Ruwase, Rachel Bawden, Stas Bekman, Angelina McMillan-Major, Iz Beltagy, Huu Nguyen, Lucile Saulnier, Samson Tan, Pedro Ortiz Suarez, Victor Sanh, Hugo Laurençon, Yacine Jernite, Julien Launay, Margaret Mitchell, Colin Raffel, Aaron Gokaslan, Adi Simhi, Aitor Soroa, Alham Fikri Aji, Amit Alfassy, Anna Rogers, Ariel Kreisberg Nitzav, Canwen Xu, Chenghao Mou, Chris Emezue, Christopher Klamm, Colin Leong, Daniel van Strien, David Ifeoluwa Adelani, and et al. 2022. BLOOM: A 176b-parameter open-access multilingual language model. *CoRR*, abs/2211.05100.

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

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

Yutao Sun, Li Dong, Barun Patra, Shuming Ma, Shaohan Huang, Alon Benhaim, Vishrav Chaudhary, Xia Song, and Furu Wei. 2022. A length-extrapolatable transformer.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA. Curran Associates Inc.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. 2023. Gated linear attention transformers with hardware-efficient training. *CoRR*, abs/2312.06635.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher

Dewan, Mona T. Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. OPT: open pre-trained transformer language models. *CoRR*, abs/2205.01068.

A Appendix

A.1 Notations Of Vanilla Transformer

In the transformer architecture, X is transformed into three distinct sequences, namely **query** (Q), **key** (K), and **value** (V), through separate linear projections. This projection is split into h attention heads, known as **Multi Head Attention**. As shown in Eq. 5, l-th head transform Q, K, V into d_h dimension, obtaining Q_l, K_l, V_l.

$$\begin{aligned} Q_l &= QW_l^Q \\ K_l &= KW_l^K \\ V_l &= VW_l^V \\ W_l^Q, W_l^K&, W_l^V \in \mathbb{R}^{d_{model} \times d_h} \end{aligned} \tag{5}$$

Attention calculation is defined as Eq. 6, where Att is known as attention score.

$$\begin{aligned} Att &= \text{softmax}\left(\frac{QK^\intercal}{\sqrt{d_h}}\right) \\ Attention(Q, K, V) &= Att \cdot V \end{aligned} \tag{6}$$

And final output of attention needs to concatenate (notated as $concat$ in equations) each head and apply a linear projection.

$$\begin{aligned} &MultiHead(Q, K, V) \\ &= concat(head_1, \dots, head_h)W_O, \\ &head_l = Attention(QW_l^Q, KW_l^K, VW_l^V) \\ &W_O \in \mathbb{R}^{d_{model} \times d_{model}} \end{aligned} \tag{7}$$

A.2 Classification of Positional Encodings

Absolute Positional Encoding For queries Q and keys K with positional information $\boldsymbol{a} = [1, 2, \dots, n]$. APE can be represented as functions to add positional information to input sequences, notated as Eq. 8.

$$\tilde{Q} = APE(Q, \boldsymbol{a}), \tilde{K} = APE(K, \boldsymbol{a}) \tag{8}$$

Relative Positional Encoding RPE leverages the positional difference, $i - j$, between the i-th token in the query and the j-th token in the key. Consequently, the similarity calculation as depicted in Eq. 1 incorporates additional relative information, denoted as $g(i - j)$, in Eq. 9. Here, f signifies a novel function designed to integrate relative positional information into the similarity calculation, where common approaches typically involve either adding or multiplying $g(i - j)$ to incorporate RPE, as discussed in (Raffel et al., 2020; Press et al., 2022).

$$Sim(Q_i, K_j) = f(Q_i, K_j, g(i - j)) \tag{9}$$

PE Convertible Between RPE and APE Some positional encodings can freely convert between RPE and APE. These encodings must satisfy Eq. 10 holds for $\forall 1 \leq i, j \leq n$ (Su et al., 2024).

$$Sim(APE(Q, i), APE(K, j)) = h(Q, K, i - j) \tag{10}$$

In Eq. 10, on the left side, this type of PE is applied to query and key, fulfilling the requirements for APE as specified in Eq. 8. On the right side of Eq. 10, this PE is related to the difference $(i - j)$ and affects both query and key, meeting the criteria for RPE outlined in Eq. 9.

A.3 Conversion of Kernel-Based Linear Attention to RNN

The process of converting kernel-based linear attention to an RNN framework hinges on the ability to decompose the similarity calculation into independent functions of queries and keys. Here, we delve into the mathematical underpinnings of this conversion, starting with the general form of linear attention:

$$Att_{i,j} = \frac{\phi(Q_i)\phi(K_j)^\intercal}{\sum_{j=1}^{i} \phi(Q_i)\phi(K_j)^\intercal}$$

The computation of the updated representation V_i' involves weighting by the attention scores:

$$V_i' = \frac{\sum_{j=1}^{i} \phi(Q_i)\phi(K_j)^\intercal V_j}{\sum_{j=1}^{i} \phi(Q_i)\phi(K_j)^\intercal}$$

This equation can be simplified by recognizing that $\phi(Q_i)$ can be factored out, leading to a recursive form that mirrors RNN computations:

$$V_i' = \frac{\phi(Q_i)(S_{i-1} + \phi(K_i)^\intercal V_i)}{\phi(Q_i)(Z_{i-1} + \phi(K_i)^\intercal)}$$

with S_{i-1} and Z_{i-1} representing cumulative sums over j up to $i - 1$, allowing for an RNN-like iterative update mechanism.

A.4 Proof of Constraints on Converting Linear Attention to RNN

The core operation of Linear Attention can be expressed as follows:

$$\text{Att}_{i,j} = \frac{\phi(Q_i)\phi(K_j)^\intercal}{\sum_{j=1}^{i} \phi(Q_i)\phi(K_j)^\intercal} \tag{11}$$

This formulation necessitates updating the representation V_i' using attention scores weighted by the respective values:

$$V_i' = \frac{\sum_{j=1}^{i} \phi(Q_i)\phi(K_j)^\intercal V_j}{\sum_{j=1}^{i} \phi(Q_i)\phi(K_j)^\intercal} \tag{12}$$

The potential for simplification arises from the ability to factor out $\phi(Q_i)$, thereby converting the attention computation into a recursive form reminiscent of RNN computations:

$$V_i' = \frac{\phi(Q_i)(S_{i-1} + \phi(K_i)^\intercal V_i)}{\phi(Q_i)(Z_{i-1} + \phi(K_i)^\intercal)} \tag{13}$$

where S_{i-1} and Z_{i-1} represent the cumulative sums over j up to $i-1$, facilitating an RNN-like iterative update mechanism.

For the transformation into RNN be viable, the positional encoding introduced must independently influence Q and K without involving cross terms of i and j. If such independence is not maintained, $\phi(Q_i)$ cannot be isolated from the summation expression, ultimately impeding the transformation of linear attention into RNN. This requirement underscores the necessity of adhering to the specified positional encoding format, ensuring that linear attention remains computationally efficient and theoretically sound.

A.5 Implementation Details of Experiments

The specific model parameters and training settings are presented in Table. 3.

A.6 Calculation and Initialization of Other Positional Encoding

RoPE (Su et al., 2024) exploits APE to catch relative Positional information. We select implementation for linear attention as Eq. 14, where R_i stands for RoPE positional encoding for position i. RoPE cancels applications of APE in normalization of similarity calculation.

$$\begin{aligned} Sim(Q_i, K_j) &= (R_i\phi(Q_i))(R_j\phi(K_j)^\intercal) \\ Att_{i,j} &= \frac{Sim(Q_i, K_j)}{\sum_{j=1}^{i} \phi(Q_i)\phi(K_j)^\intercal} \end{aligned} \tag{14}$$

Parameter	Value
Number of Layers	12
Attention Heads	12 per layer
Hidden Dimension	64 per attention head
Batch Size	640
Training Text Length	512 tokens
Learning Rate	5e-4
Learning Rate Schedule	Cosine
Warmup Rounds	3000
Epochs	1
Gradient Optimizer	Adam (Kingma and Ba, 2015)
Total Parameters	137M

Table 3: Training Configuration and Model Parameters

Vanilla APE of Transformer (Vaswani et al., 2017) applies a trainable embedding [5] for absolute positional information $E(\boldsymbol{a}), \boldsymbol{a} = [1, 2, \ldots, n]$. The embedding is initialized randomly.

$$Sim(Q_i, K_j) = \phi(Q_i + E(\boldsymbol{a})_i)\phi(K_j + E(\boldsymbol{a})_j)^\intercal \tag{15}$$

For D2D, we initialize $\boldsymbol{P_l^s}$ for each head l with a zero vector $\mathbf{0} \in \mathbb{R}^{1\times d_h}$. $\boldsymbol{P_l^b}$ is initialized with scalar P_l^b in Eq. 16, where h indicates the number of heads, and then fill the vector $\boldsymbol{P_l^b}$ with the scalar.

$$P_l^b = 2^{-\frac{h}{l}} \tag{16}$$

A.7 Experiments For Effective Inference

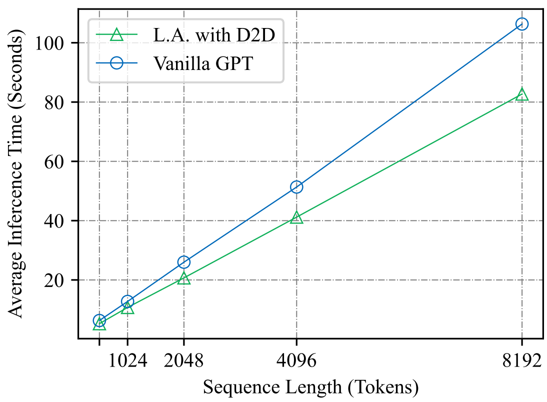

Figure 8: Average infercene time for sequence with different length. L.A. with D2D stands for linear attention with D2D.

To ensure that D2D exhibits superiority in terms of inference speed compared to the vanilla model,

[5]Trainable embedding is only added in the first layer of GPT-2 in vanilla implementation.

we conduct speed tests for language generation at the inference stage. We transform our method into RNN-form to achieve $O(n)$ time complexity. We eliminate the "End of Sequence" (EOS) token from the vocabulary to guarantee the production of texts that conform to specified length criteria. We conducte ten experiments for each model at each length and took the average as the generation time. The weights of the model are subjected to random initialization, given that this has no impact on the assessment of generation speed.

Results indicate that inference time complexity of our method is lower than that of the vanilla GPT, and as the inference length increases, the advantages of our method become increasingly pronounced. When the sequence length is relatively short, the improvement in time is not very pronounced, as the fundamental computations and data copying still require a certain amount of time.