Disentangle to Decay: Linear Attention with Trainable Decay Factor

Haibo Tong [*], **Chenyang Zhang** [*], **Jiayi Lin, Bingxuan Hou,**
Qingqing Hong, Junli Wang [†]
Key Laboratory of Embedded System and Service Computing (Tongji University),
Ministry of Education, Shanghai 201804, China.
National (Province-Ministry Joint) Collaborative Innovation Center
for Financial Network Security, Tongji University, Shanghai 201804, China.

{2151130,inkzhangcy,2331908,2052643,2332012,junliwang}@tongji.edu.cn

Abstract

Linear attention enhances the inference efficiency of the 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 decay factor of exponential calculation, which limits further optimization. Our analysis reveals directly training decay factor is unstable because of large gradients. To address this, we propose a novel PE for linear attention named Disentangle to Decay (D2D). D2D disentangles the decay factor into two parts to achieve further optimization and stable training. Moreover, D2D can be transformed into a recurrent form for efficient inference. Experiments demonstrate that D2D achieves stable training of decay factor, and enhances the performance of linear attention in both normal context length and length extrapolation scenarios [1].

1 Introduction

Linear attention (Katharopoulos et al., 2020) substitutes the softmax calculation in vanilla Transformers (Vaswani et al., 2017) with a dot-product of kernel feature maps. It achieves linear complexity during inference, which is particularly advantageous for processing long sequences (Katharopoulos et al., 2020). However, some challenges such as cumulative regularity errors over long sequences necessitate specialized mechanisms for effective information filtering (Qin et al., 2022a). PE contributes to effective information filtering since it integrates positional information for language models. For existing linear attention language models (Sun et al., 2023; Qin et al., 2024), they introduce a PE structure including decay terms $\gamma^{(i-j)}$, where γ is the **decay factor** and $i-j$ represents the relative position between the query and the key tokens. The decay factor provides an information-forgetting mechanism, which alleviates the aforementioned issue and enhances the capability of processing longer sequences.

However, decay factors used in these models are manually designed and non-trainable, limiting further optimization (Moreno-Cartagena et al., 2023). We reveal that directly training decay factor might generate enormous gradients, due to exponential calculation with a trainable base. Consequently, large gradients integrate numeric instability, leading training to collapse. Models fail to yield expected outcome from the trainable decay factor.

To enhance the stability of training and the performance of models, our work proposes an innovative trainable decay factor based PE named **D**isentangle to **D**ecay (**D2D**). D2D disentangles the decay factor into two parts: **Global decay factor** is fixed with an effective initialization, providing the numeric foundation; **local tuning factor** is trainable for further optimization. With the initial value of the global decay factor, the local tuning factor exhibits mild numeric fluctuation and bounded gradients for stable training. We use a mixed form of absolute positional encoding (APE) and relative positional encoding (RPE) in our D2D implementation. Consequently, D2D avoids unnecessary calculations and addresses the precision problems of floating-point. We offer the whole training process and the recurrent inference for D2D.

In experiments, we construct language models using D2D and other baselines, with a pre-training scale similar to GPT-2 (Radford et al., 2019). Then we conduct various experiments on language modeling, length extrapolation, and downstream tasks. D2D exhibits training stability and facilitates enhanced performance for language models than di-

[*]Equally Contribution.
[†]Corresponding author.
[1]Our code implementation is available at https://github.com/TongjiNLP/Disentangle-to-Decay-Linear-Attention-with-Trainable-Decay-Factor

Proceedings of the 31st International Conference on Computational Linguistics, pages 9877–9890
January 19–24, 2025.

rectly trained decay factors and untrained decay factors. Moreover, D2D outperforms existing PE in linear attention, including RoPE (Su et al., 2024) and ALiBi (Press et al., 2022). D2D fulfills the recurrent inference of linear attention and attains the expected generation efficiency.

Our main contributions are listed below:

(1) We analyze the stability of the training decay factor in linear attention PE. Training of the decay factor exhibits a large and unbounded gradient, leading to training collapse or sub-optimal results.

(2) We propose D2D, a novel trainable PE for linear attention, which maintains stability during training and enhances representational capability. We provide an optimized implementation for training and efficient inference, addressing intrinsic precision and space issues in intuitive implementation.

(3) We construct language models for D2D and other baselines and then conduct various experiments. Results show that D2D demonstrates training stability and superior ability.

2 Preliminary

2.1 Computational Form of Linear Attention

For query token Q_i in position i and key token K_j with position j, unified formulation of linear and vanilla attention is given in Eq. 1 (Katharopoulos et al., 2020), where similarity calculation $Sim(Q_i, K_j)$ quantifies relationship between query of the i-th token and 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 (Vaswani et al., 2017), similarity is calculated using the exponent of 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, as $Sim(Q_i, K_j) = \phi(Q_i)\phi(K_j)^\intercal$.

2.2 Constraints of PE in Linear Attention

For efficient inference, PE used in linear attention must satisfy certain constraints. Katharopoulos et al. (2020) proposes the approach of converting linear attention to recurrent form for inference. This transformation is contingent upon a specific positional encoding format, as detailed in Eq. 2, where f_q and f_k are functions applied to Q_i and K_j, respectively, to incorporate absolute positional information. Through this equation, 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.2.

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

3 Instability of Training Decay Factor

For most decay factor based PEs, decay factors are fixed number rather than trainable, since they do not achieve better performance (Press et al., 2022; Sun et al., 2023). We analyze that training of decay factor exhibits **numerical instability**, leading to training collapse and limited optimization.

Numerical Instability The value of decay factor exhibits significant fluctuations throughout the training process and fails to converge rapidly. 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 When the decay factor becomes trainable, the exponential calculation involves higher-order terms of decay factor, which can generate large gradients.

For two tokens separated by a relative distance of δ, a higher-order term γ^δ is adopted in the calculation (Qin et al., 2024; Sun et al., 2023), where γ is the decay factor. When γ becomes trainable, it generates gradient of $\frac{d(\gamma^\delta)}{d\delta} = \delta\gamma^{\delta-1}$.

Theoretically, directly training decay factor leads to larger unbound gradient in linear attention. When the range of δ increases, the gradient produced by the global decay factor can potentially reach a very large value. Previous work (Qin et al., 2022a) has demonstrated linear attention suffers from large gradient compared with vanilla attention. Directly training decay factor will further amplify unstable gradients produced by linear attention.

Practically, gradient is not acceptable during directly training decay factor. Taking settings in Sun et al. (2023) [2] as an instance, training decay factor will integrates gradient with a maximum value of 376. The large gradient is integrated in overall gradient by multiplication. Consequently, the entire training will suffer from gradient explosion. In subsequent experiments, we observe a collapse of training when directly training decay factor. This phenomenon is consistent with the analysis above.

[2] It exploits a context length of 1024 for language model, and set decay factor close to 0.999 for some attention heads.

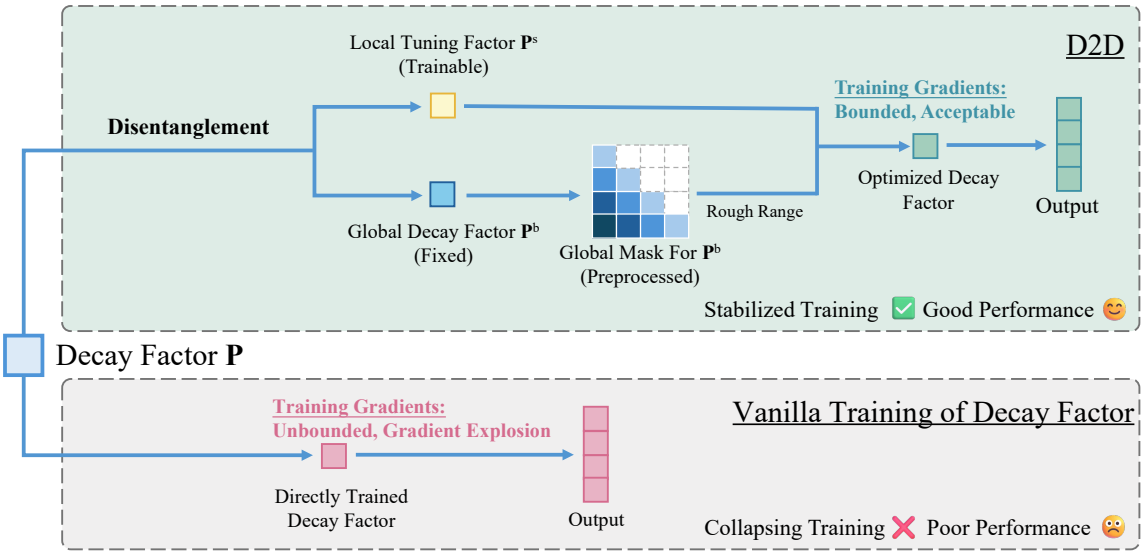

Figure 1: A training process overview of D2D and vanilla of decay factor. Firstly, D2D disentangles the decay factor into the global decay factor and local tuning factor. With the initialized global decay factor, training of the local tuning factor exhibits bounded gradients. By summing them, a well-optimized decay factor is obtained. Global decay factor is integrated with mask matrices, to avoid precision problems and improve efficiency. Conversely, directly training decay factor exhibits unbounded large gradients, leading to training collapse and poor performance.

4 Method

We propose D2D, an effective solution to the instability of the decay factor during training. The main structure of D2D compared with vanilla decay factor based PE is demonstrated in Figure 1.

4.1 Disentanglement based PE

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

Then, we disentangle the decay factor of D2D into **global decay factor** and **local tuning factor**.

Global decay factor $\boldsymbol{P^b}$ is a vector assigned to each attention head. Global decay factor for l-th head is represented as $\boldsymbol{P_l^b} \in \mathbb{R}^{1\times d_h}$, where $\boldsymbol{P_l^b} = (p_l^b, \ldots, p_l^b)$ consists of same fixed scalars p_l^b. It provides a rough range for the decay factor.

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 the 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 Figure 2, the sum of two factors takes up a wider range of distribution, which is beneficial for optimization.

Positional Encoding Design Based on the aforementioned disentanglement, we proposed calculation of D2D in Eq. 3, where $Sim(Q_i, K_j)[l]$ represents similarity calculation for l-th attention head. All divisions are performed element-wisely.

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

D2D exploits a mixed form of APE and RPE [3] (Wang et al., 2021). **APE form** calculates of query and key indices i, j separately, like first line of Eq. 3. And **RPE form** calculates a function of i, j together, like calculation of Θ_b.

During training, PE is calculated with $\Theta_b \cdot \Theta_s$. Θ_s is trainable vector, it is calculated in APE form because of essential time and space complexity concerns. Θ_b is calculated in RPE form, it provides foundation for mask implementation in Section 4.3. Mask implementation is necessary for efficiency and precision of floating-point numbers.

During inference, PE is represented using APE form, as the first line of Eq. 3. In calculation, value

[3] In Appendix C, we explore the necessity of APE and RPE for corresponding calculations.

of $\boldsymbol{P}_l$ is derived from $\boldsymbol{P}_l^b + \boldsymbol{P}_l^s$. APE form is necessary owing to constraints of converting linear attention into RNN in Section 2.2. Consequently, D2D is available for recurrent inference using approaches described in Eq. 4 (Katharopoulos et al., 2020). 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.1.

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

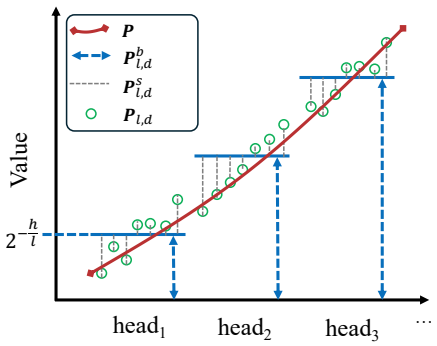

Figure 2: Illustration of disentanglement. Green circle stands for each index of $\boldsymbol{P} = \boldsymbol{P^b} + \boldsymbol{P^s}$. To visualize the value of $\boldsymbol{P}$, we approximate it with a smooth red curve on the Figure. Possible sum of them could cover a wide range during optimization. $\boldsymbol{P_{l,d}}$ 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 an optimal foundation for PE (Press et al., 2022). Compared with other initializations, we provide a more structured initialization in D2D, facilitating faster convergence and better overall model performance.

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. As described in Press et al. (2022), this initialization provides a more concentrated distribution of decay factor near zero, and it is effective for the representation of positional information. For the local tuning factor, we apply zero initialization since we expect a small range of local tuning factors for training stability concerns.

In D2D, the global decay factor provides a foundation for the training of the local tuning factor. Once the global decay factor in each attention head is initialized to an appropriate value, the range of $\boldsymbol{P_l^s}$ is narrowed during gradient descent. D2D avoids rapid numerical fluctuations, enhancing training stability. Experiments in Section 6 validate the above analysis.

Stabilizing Gradients of Decay Factor D2D brings a bounded gradient for the training decay factor. For l-th attention head, gradient g_l satisfies $|g_l| \leq \frac{1}{e \cdot p_l^b}$, as proved in Appendix B. p_l^b is a fixed value manually initialized, which can control the maximum gradient. Compared to unbounded gradients of directly training decay factor in Section 3, D2D mitigates gradients to an acceptable range for stable training. Taking the first attention head as a practical instance, D2D reduces the gradient of the decay factor from 376 to 6.87. In practical training, the gradient is acceptable.

4.3 Mask-based Efficient Training Implementation

4.3.1 Limitation Of Original Implementation

D2D stabilizes training of the decay factor according to the aforementioned analysis. But original implementation of D2D encounters some problems in practical training, as analyzed in the following paragraphs:

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 in unnecessary exponential calculations. This problem also exists when directly training the decay factor.

Precision In Floating-Point Arithmetic During the training phrase, calculation in the first line of Eq. 3 encounters precision problems of floating-point. For exponential calculation $\exp(x)$, the result is not reliable when $|x|$ becomes very large in training.

For decay factor based PE, $\exp(\gamma i) \cdot \exp(-\gamma j) = \exp(\gamma(i-j))$ should hold for all position indices. But $|-\gamma j|$ and $|\gamma i|$ could be very large in practical training, causing precision problems. Consequently, the product does not match the theoretical value. This causes the value of D2D, which is only related to relative positions, to be affected by absolute positions of tokens.

Precision problems disrupt the theoretical property of D2D. This concern emerges more frequently for longer sequences due to larger position indices. Moreover, we observe limited optimization of decay factor in training, due to precision problems [4].

4.3.2 Masked-based Transformation

Preprocessing of global decay factor is available, because $\boldsymbol{P}_l^b$ in Eq. 3 consists of identical scalars p_l^b. Therefore, the global decay factor can be factored out as Θ_b, which 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 Figure 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 auto-regressive language modeling. For different attention heads, we preprocess matrices respectively, since number of attention heads is usually limited.

To integrate this mask, we apply the element-wise product of mask and attention scores. To save time and space cost, we replace *causal mask* [5] with M.

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

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

Effectiveness of Mask For extra time cost, our mask is integrated into a causal mask of language model and avoids extra calculation.

Regarding the precision problem, the mask applies RPE calculation $\exp(\gamma(i-j))$ for the global decay factor, to avoid precision calculation. And only the local tuning factor is calculated in APE form $\exp(\gamma i) \cdot \exp(-\gamma j)$. The local tuning factor is significantly smaller than the whole decay factor, and it requires a larger exponent to trigger precision problems. In our practical experiments, the precision problem is solved through mask mechanisms.

[4] We record the distribution of decay factor in practical training in Appendix D.4, which demonstrates a truncated optimization range.

[5] For auto-regressive language models, the causal mask is a lower triangular matrix to ensure attention is unidirectional.

4.4 Overall Training And Inference Implementation

Algorithm 1 and 2 respectively illustrate the whole process training and inference for D2D-based linear attention. $\div$ 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).

The difference between training and inference lies in how to introduce D2D into attention output, while the remaining steps both follow procedure in Katharopoulos et al. (2020).

Algorithm 1 Attention Output During Training

procedure ATTN($Q, K, V, M, \boldsymbol{P}^s, n$)
  $K \leftarrow K^\intercal$
  $Q, K \leftarrow \phi(Q), \phi(K)$
  $\boldsymbol{a} \leftarrow (0, 1, \ldots, n-1)$
  $C \leftarrow \exp(\boldsymbol{a} \cdot \boldsymbol{P}^s)$
  $Q \leftarrow Q \div C$
  $K \leftarrow K \odot C$
  $Q, K, V \leftarrow$ 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
  $O \leftarrow Att \cdot V$
  $O \leftarrow$ mergehead (O)
  return O
end procedure

Algorithm 2 Attention Output During Inference

procedure ATTN($Q, K, V, \boldsymbol{P}^b, \boldsymbol{P}^s, n$)
  $K \leftarrow K^\intercal$
  $\boldsymbol{P} \leftarrow \boldsymbol{P}^b + \boldsymbol{P}^s$
  $\boldsymbol{P} \leftarrow \exp(\boldsymbol{P})$
  $\boldsymbol{S}, \boldsymbol{Z} \leftarrow \mathbf{0}_{d_h \times d_h}, \mathbf{0}_{d_h \times 1}$
  $Q, K, V \leftarrow$ 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
  $O \leftarrow concat(O_1, \ldots, O_n)$
  $O \leftarrow$ mergehead (O)
  return O
end procedure

5 Experiments

5.1 Experiment Settings

5.1.1 Language Model Construction

Previous work (Press et al., 2022; Su et al., 2024) constructs language model with designed PE to demonstrate their performance. In this step, a pure

Datasets	Language Modeling					Length Extrapolation					Downstream Tasks		
	enwiki8 (PPL↓)	LAMBADA (PPL↓)		WikiText2 (PPL↓)		Open WebText (PPL↓)	GovReport (PPL↓)		PG19 (PPL↓)		ARC-e (ACC↑)	ARC-c (ACC↑)	SQuAD (F1↑)
Finetune	w/o	w/o	w/	w/o	w/	w/o	w/o	w/	w/o	w/	w/	w/	w/
Methods													
Fixed.	94.91	95.06	31.53	96.29	18.57	67.64	24.14	16.78	198.53	40.52	0.250	0.218	0.504
D.T.	92.27	**89.01**	29.65	85.07	18.40	62.53	22.77	16.69	174.81	34.38	0.251	0.234	0.518
D2D	**86.36**	90.63	**25.83**	**72.48**	**18.29**	**57.40**	**21.25**	**15.97**	**169.99**	**29.76**	**0.262**	**0.256**	**0.536**

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

attention based language model backbone is required for experiments. Attention should be integrated simply without additional architectural modification. Consequently, recent work with linear attention like RetNet (Sun et al., 2023), Mamba (Gu and Dao, 2024) is not available for comparison.

In our experiments, we adopt GPT-2 (Radford et al., 2019) as the language model backbone and replace vanilla position embedding with D2D in attention calculation. We reconstruct the language model through a whole pre-training epoch on OpenWebText dataset (Gokaslan and Cohen, 2019), with a similar size and training steps to Radford et al. (2019). The training scale is also comparable with previous PE research like RoPE (Su et al., 2024) and ALiBi (Press et al., 2022) [6].

5.1.2 Baselines Design

Aside from D2D based language model, we design two additional baselines for comparison, aiming to probe the effectiveness and training stability of D2D. Overall baselines are mentioned as follows.

D2D: This baseline exploits D2D implementation in training as mentioned before.

Fixed decay factor (notated as *Fixed.*): This baseline has fixed decay factors during training, initialization is the same with Section 4.2.

Directly trained decay factor (notated as *D.T.*): The decay factor is randomly initialized and trained directly alongside the model. In practical training, we discover that training frequently collapses due to a large gradient and that the language model parameters are not available. To observe the performance before collapse, we make manipulation to narrow gradients in an acceptable range [7].

[6] In Appendix D, we provide detailed experiment settings and supplementary experimental observations (i.e. generation efficiency of recurrent form of D2D).

[7] We truncate the value of exponential calculation, to avoid large gradients. Range of truncation is determined by repetitive experiments, to avoid generating fatal gradient explosion.

5.2 Basic Performance

5.2.1 Language Modeling

Language modeling is a basic ability in experiments. We test language modeling performance of different baselines in various text datasets, including enwiki8 [8], LAMBADA (Paperno et al., 2016) and WikiText2 (Merity et al., 2016). Following Radford et al. (2019), we calculate the perplexity of language models. As shown in Table 1, the model exhibits good language modeling performance with D2D, with the benefit of better positional information.

5.2.2 Downstream Task

We evaluated the capability of D2D on downstream tasks through instruction tuning, as a supplement to language modeling experiments. We exploit multiple-choice task ARC-e and ARC-c (Clark et al., 2018) and question-answering task SQuAD (Rajpurkar et al., 2016) as downstream tasks, with 1 epoch fine-tuning on corresponding training dataset. We calculate Accuracy for ARC and F1-Score for SQuAD as experiment results. Experiment results demonstrates that D2D achieves better instruction ability.

5.3 Length Extrapolation

Following Press et al. (2022), we test language modeling performance on longer sequences than training, to demonstrate superiority of D2D in length extrapolation. We provide three series of experiments, settings are listed below.

In-domain Length Extrapolation We observe the perplexity of these baselines in the validation set of OpenWebText, which shares the same distribution with the pretrained corpus. We exploit a test

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

length of 1024, which is longer than the training length of 512.

Out-Domain Length Extrapolation Corpus out of pretrained domain is integrated into length extrapolation experiments. Following Rae et al. (2020); Dong et al. (2024), we integrate GovReport (Huang et al., 2021) and PG19 (Rae et al., 2019) as a dataset, with a test length of 1024. Additionally, we fine-tune each baseline with a corresponding training corpus with a sequence length of 512 and record their length extrapolation results as supplementary.

Extrapolation On Varying Sequence Lengths Following Press et al. (2022), we conduct length extrapolation on different sequence lengths, with the same setting as In-domain Length Extrapolation. We demonstrate perplexity with different sequence lengths of these three baselines.

Experiment Results In-domain and out-domain results are shown in Table 1. D2D outperforms other baselines, especially in complex text datasets (PG19), indicating that D2D can capture more positional information in long sequences.

We demonstrate extrapolation on varying sequence lengths in Figure 4, D2D exhibits better perplexity than other baselines, especially for sequences very short (200) and long (1000). The result indicates necessity and effectiveness in D2D, since *D.T.* and *Fixed.* exhibit drawback in various lengths, D2D outperforms them in all of the lengths.

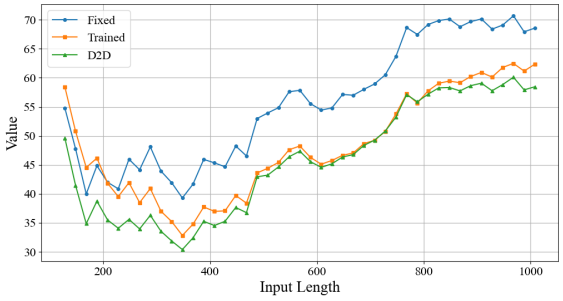

Figure 4: The figure illustrates the language model's ability to extrapolate the length within the domain. As the length increases, the model using the decay factor initially shows a decreasing trend in PPL, followed by an increase, and eventually becomes stable.

5.4 Comparing with Other Positional Encoding

We compare D2D with existing PE for linear attention in a smaller pretrained corpus (%10 of aforementioned settings). Other experiments' settings are the same with Section 5.1.1.

We select RoPE (Su et al., 2024), ALiBi (Press et al., 2022), Vanilla APE (Vaswani et al., 2017; Radford et al., 2019) and implement them on linear attention language models. RoPE and ALiBi exhibit decay properties and are frequently used as position-encoding methods in linear attention. Additionally, we integrate a language model with full attention into vanilla APE. Details are shown in Appendix D.3.

As results shown in Table 2, D2D demonstrates a lower loss during training compared to full attention, exhibiting good performance, even surpassing full attention, which indicates the effectiveness of linear attention models. However, linear attention itself has certain limitations, showing weaker generalization ability compared to full attention, as detailed in Appendix D.6.

6 Discussions

6.1 Training Stability

We record numerical fluctuations of the trainable part of the decay factor in different baselines, to demonstrate the numerical stability of *D.T.* and D2D in Section 5.1.2. For *D.T.*, we record the range of the decay factor. For D2D, we record range of $\boldsymbol{P^s}$ during training since $\boldsymbol{P^b}$ is not trainable. As shown in Figure 5, the trainable part of D2D exhibits slighter fluctuation. With the assistance of global decay factor, local tuning factor $\boldsymbol{P^s}$ does not require large-scale training compared to *D.T.*.

6.2 Representation Ability of Decay Factor

To test the representation ability of D2D, we illustrate the value of $\boldsymbol{P^s}$, $\boldsymbol{P^b}$, and $\boldsymbol{P^s} + \boldsymbol{P^b}$ in a trained D2D model. Figure 6 illustrates values in the first head. Trainable local tuning factor remains

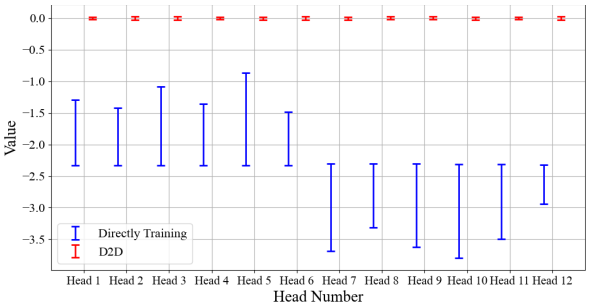

Figure 5: The numerical fluctuations of the D2D and directly trained decay factor from the first layer of linear attention model during the training process.

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	47.80	47.85	**46.90**

Table 2: Language modeling performance of different PE. Values bold are denoted as optimal results.

a stable range in different heads. However, with the assistance of $\boldsymbol{P^b}$, final outcome of the decay factor achieves a wide coverage.

6.3 Discussion of Scaling

We have exploited a reasonable experiment scale in the aforementioned experiments, which is comparable with recent PE works on linear attention. Theoretically, stability and length extrapolation performance of D2D is expected to hold for a larger scale of training, The reasons are as follows:

Scaling of Parameters The main reason for unstable training is gradient. However, the gradient in training is irrelevant to training steps and parameters, according to Section 3 and Section 4.2.

Training collapse is inclined to occur in the early steps of training, and more training steps would not affect training stability significantly, because the decay factor needs more optimization in early steps. In our experiments of direct training decay factor, the collapse occurs in the first 5% of training steps.

Scaling of Sequence Length Larger-scale length extrapolation experiments require larger pre-trained models and fine-tuning on ultra-long text corpora, which is not the core goal of the D2D design. To demonstrate the extrapolation ability of D2D with longer corpora, we present the trend of PPL for different position encodings as the input corpus length changes in Figure 4. It can be inferred that D2D will also have advantages with longer inputs.

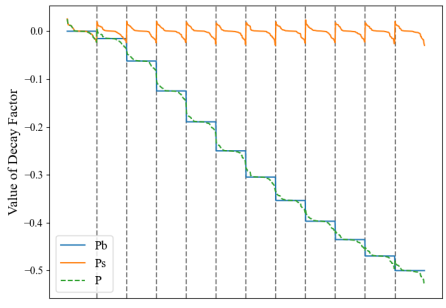

Figure 6: The value of decay factor in the first layer of D2D based linear attention model. To enhance image clarity, we use vertical gray dashed lines to split heads and sort $\boldsymbol{P^s}$ within each head.

6.4 Discussion MSA Employed in D2D

When using the multi-head self-attention (MSA) mechanism (Vaswani et al., 2017), D2D can maximize its advantages. As shown in Section 4.2, MSA allows D2D to assign different global decay factors to each head, which enhances its representational capability. Moreover, MSA generally provides stronger representational power than single-head attention.

Additionally, when using single-head attention, it is necessary to partition the dimensions of the head and apply different initializations to achieve the effect of the global decay factor. This leads to the failure of the training acceleration method described in Section 4.3.2, reducing the training efficiency of the model. Therefore, we mainly discuss the performance of D2D when using MSA.

6.5 Discussion of Other Gradient-Control Approaches

Regarding issues of large gradients, common gradient-control approaches don't work properly.

(1) Large gradient varies for experiment circumstances, such as attention layers and dataset distributions. And gradient may be very large and unbounded. Hence, manually clipping gradients into certain values is ineffective, the clipping value is hard to decide.

(2) In training, large gradients occur sporadically, while other gradients have acceptable values. Therefore, normalizing gradients using overall distribution is ineffective, as it excessively weakens other gradients but not fully address the large ones.

(3) As shown in Section 4.2, D2D solves this issue by generating coefficients for higher-order terms through the global decay factor, avoiding large gradients generation and obtaining expected performance.

6.6 Advantages of D2D

In this section, we analyze improvements of D2D in the following three aspects of experiments:

Improvements in **language modeling** can be attributed to the stable training 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 the local tuning factor. Taking Figure 6 as an instance, $\boldsymbol{P^b} + \boldsymbol{P^s}$ is 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. Kernel-based linear attentions (Qin et al., 2022b; Katharopoulos et al., 2020; Qin et al., 2022a) process query and key with kernel functions [9]. Transformation of linear attention into a recurrent form enables efficient inference, as explored by Katharopoulos et al. (2020) and further applied in large-scale models (Yang et al., 2023; Sun et al., 2023).

7.2 Positional Encoding

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

PE can be categorized into APE, RPE and convertible PE. APE uses the function of absolute positions (Vaswani et al., 2017; Brown et al., 2020; Zhang et al., 2022). RPE accounts for relative distances between tokens (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 for switching between APE and RPE, facilitating flexible computational strategies (Su et al., 2024).

Commonly used RPEs such as Su et al. (2024); Press et al. (2022); Sun et al. (2022) exhibit certain decay properties, which cause the model to focus more on closer tokens. This leads to a more focused attention structure, thereby improving its language modeling capabilities (Han et al., 2023).

8 Conclusion

In this paper, we design a positional encoding method, D2D, for models based on linear attention. By analyzing the gradients of the training decay factor, we reveal the numeric instability of direct training decay factor. To stabilize training, we disentangle D2D during the training process, transforming it into a combination of APE and RPE. Disentanglement of decay factors and proper initialization contribute to the bounding gradient of training into an acceptable range. 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. We conduct various experiments on D2D based linear attention models and other baselines. Results demonstrate the effectiveness and training stability of D2D.

9 Limitation

Our positional encoding demonstrates effectiveness across various kernel functions. However, the performance may differ depending on the specific kernel function used. 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, D2D has not been tested on structures other than the linear attention framework. We conducted discussions on an acceptable scale and provided a theoretical analysis of the potential effects on larger models.

We welcome training on larger models and corpora, as well as the use of D2D to replace existing methods in frameworks that adopt decay factors, in order to enhance model performance.

Acknowledgments

This work is supported by the National Key Research and Development Program of China under Grant 2022YFB4501704.

[9] Random-based linear attention (Peng et al., 2021; Choromanski et al., 2021) exploits other manners, which is beyond discussion range of this paper.

References

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. *Preprint*, arXiv:1803.05457.

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. *Preprint*, arXiv:2405.18009.

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

Albert Gu and Tri Dao. 2024. Mamba: Linear-time sequence modeling with selective state spaces. *Preprint*, arXiv:2312.00752.

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. *Preprint*, arXiv:1609.07843.

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. *Preprint*, arXiv:2308.06404.

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. *Preprint*, arXiv:1606.06031.

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. *Preprint*, arXiv:2307.14995.

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. *Preprint*, arXiv:1911.05507.

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).

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

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. *Preprint*, arXiv:2212.10554.

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.

Benyou Wang, Lifeng Shang, Christina Lioma, Xin Jiang, Hao Yang, Qun Liu, and Jakob Grue Simonsen. 2021. On position embeddings in {bert}. In *International Conference on Learning Representations*.

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 Conversion of Kernel-Based Linear Attention to RNN

A.1 Conversion Details

The process of converting kernel-based linear attention to an RNN framework (Katharopoulos et al., 2020) 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.2 Proof of Constraints on Converting Linear Attention to RNN

The core operation of Linear Attention can be expressed as follows, where $Att_{i,j}$ stands for corresponding attention score:

$$\text{Att}_{i,j} = \frac{\phi(Q_i)\phi(K_j)^\intercal}{\sum_{j=1}^{i}\phi(Q_i)\phi(K_j)^\intercal} \tag{5}$$

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

$$V_i' = \frac{\sum_{j=1}^{i}\phi(Q_i)\phi(K_j)^\intercal V_j}{\sum_{j=1}^{i}\phi(Q_i)\phi(K_j)^\intercal} \tag{6}$$

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:

$$V_i' = \frac{\phi(Q_i)(S_{i-1} + \phi(K_i)^\intercal V_i)}{\phi(Q_i)(Z_{i-1} + \phi(K_i)^\intercal)} \tag{7}$$

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 to 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.

B Bound of Gradients in Training of D2D

After adding the global decay factor, the absolute value of the gradient produced by the local tuning factor is $\delta \exp(-p_l^b)^\delta \exp(-\boldsymbol{P_l^s})^\delta$. Compared directly training in Section 3, D2D has an extra coefficient $\exp(-p_l^b)^\delta$, where $\exp(-p_l^b) < 1$. This term decreases with the growth of δ, mitigating large gradients. Since $\boldsymbol{P_l^s}$ is much smaller than p_l^b during training and can be approximated as zero, the gradient simplifies to $\delta \exp(-p_l^b)^\delta$. By taking the partial derivative with respect to δ, we obtain the maximum gradient at $\delta = \frac{1}{p_l^b}$, leading to a maximum absolute gradient value of $\frac{1}{p_l^b e}$. This value can be controlled by adjusting p_l^b. In contrast, as shown in Section 3, when γ approaches 1, the gradient generated by directly training decay factor is only controlled by the δ and has no theoretical upper bound.

C Space Complexity Discussion for D2D

For D2D, the calculation of global decay factor $\boldsymbol{P^b}$ is in RPE form, while the calculation of local tuning factor $\boldsymbol{P^s}$ is in APE form. This is motivated by space and time complexity considerations.

Calculation of Global Decay Factor For the RPE form of calculation $\boldsymbol{P_l^b}$, the values within each head for $\boldsymbol{P^b}$ are identical. Consequently, the element-wise product is convenient for integration $\boldsymbol{P_l^b}$. As mentioned in Section 4.3, RPE calculation can be transformed into several mask matrices, and integrated to attention score with element-wise product. Since the value of $\boldsymbol{P^b}$ varies for different attention heads, h [10] matrices should be calculated. So the total size of the RPE calculation is $h \times len \times len$ [11]. Space complexity is acceptable since head numbers h are usually small in settings. And mask-based RPE calculation employs element-wise product, which is efficient.

[10] h is the number of attention heads in the language model, which is set to 12 in our experiments.

[11] len is context length of the language model, known as a maximum positional index.

Calculation of Local Tuning Factor Local tuning factor $\boldsymbol{P^s}$ varies within the head, and it has a total dimension of $d = d_h \times h$ [12]. If we were to apply a transformation to $\boldsymbol{P^s}$ similar to that of $\boldsymbol{P^b}$, we need to decompose the computation of Θ_s:

$$\Theta_s = \Sigma\phi(Q_{ik}) \cdot \phi(K_{j_k}) \exp((i-j)P^s_{l\,k}) \quad (8)$$

Term $\exp((i-j)P^s_{l\,k})$ can also be transformed into a relative position encoding mask. Since $\boldsymbol{P^s_l}$ varies across each dimension, a separate mask must be created for each one. $d \times len \times len$ matrices should be integrated. In practical settings, d is usually significantly larger than h, and space cost is not acceptable.

Serial mask calculation from each model dimension can address space issues, but it exploits d times of additional multiplication, which is not acceptable in time cost. In our experiments, we find it is 5 times slower in practical implementation.

Consequently, APE is necessary in the calculation of Θ_s. APE form of calculation conserves the use of additional masks with very few calculations. We narrow the range of local tuning factors in Section 4.2 and Section 4.3 to avoid latent issues like numerical instability and precision in floating-point.

D Experiment Details

D.1 Implementation Details of Experiments

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

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 Scheduler
Warmup Steps	3000
Epochs	1
Gradient Optimizer	Adam (Kingma and Ba, 2015)
Total Parameters	137M

Table 3: Training Configuration and Model Parameters

[12] d is known as model dimension in Vaswani et al. (2017). And d is set to 768 in our experiments.

D.2 Kernel Selection

For linear attention, there is no one-size-fits-all criterion for selecting the kernel function; thus, it is essential to choose the most appropriate kernel function based on the position encoding we use. For the position encoding we employ, using $elu(x) + 1$ as the kernel function achieves a lower perplexity (ppl) compared to using $\exp(x)$.

D.3 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. 9, where R_i stands for RoPE positional encoding for position i. RoPE cancels applications of APE in the normalization of similarity calculation.

$$Sim(Q_i, K_j) = (R_i\phi(Q_i))(R_j\phi(K_j)^\intercal)$$
$$Att_{i,j} = \frac{Sim(Q_i, K_j)}{\sum_{j=1}^{i} \phi(Q_i)\phi(K_j)^\intercal} \quad (9)$$

Vanilla APE of Transformer (Vaswani et al., 2017) applies a trainable embedding [13] for absolute positional information $E(\boldsymbol{a}), \boldsymbol{a} = [1, 2, \ldots, n]$. The embedding is initialized randomly.

$$Sim(Q_i, K_j) = \phi(Q_i + E(\boldsymbol{a})_i)\phi(K_j + E(\boldsymbol{a})_j)^\intercal \quad (10)$$

For D2D, we initialize $\boldsymbol{P^s_l}$ for each head l with a zero vector $\mathbf{0} \in \mathbb{R}^{1\times d_h}$. $\boldsymbol{P^b_l}$ is initialized with scalar P^b_l in Eq. 11, where h indicates the number of heads, and then fill the vector $\boldsymbol{P^b_l}$ with the scalar.

$$P^b_l = 2^{-\frac{h}{l}} \quad (11)$$

D.4 Training Outcome of Precision Problem

In Section 4.3, we discuss the issue of not converting Θ_b into a mask. To address this, we directly train a linear attention model using D2D without any transformations. As shown in Figure 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 Section 4.3 significantly impacts training, limiting the range of values for the D2D.

D.5 Experiments For Effective Inference

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

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

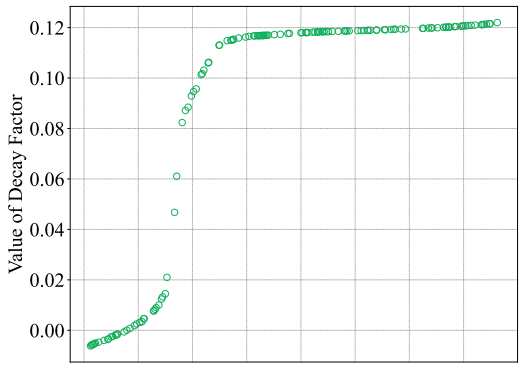

Figure 7: The results of directly training D2D without converting $\boldsymbol{P}^{\boldsymbol{b}}$ into mask. The image displays the values of $\boldsymbol{P}$ in the first layer of the model.

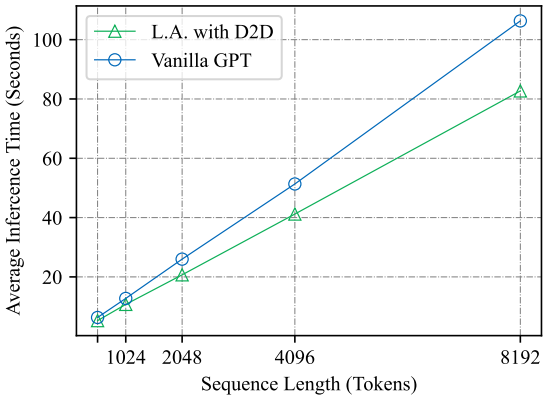

Figure 8: Average inference time for the sequence with different lengths. L.A. with D2D stands for linear attention with D2D.

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 conduct 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.

As shown in Figure 8, the inference time complexity of our method is lower than that of the vanilla GPT. Moreover, as the inference length increases, the advantage of D2D becomes 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.

D.6 Comparison of Pre-training Processes

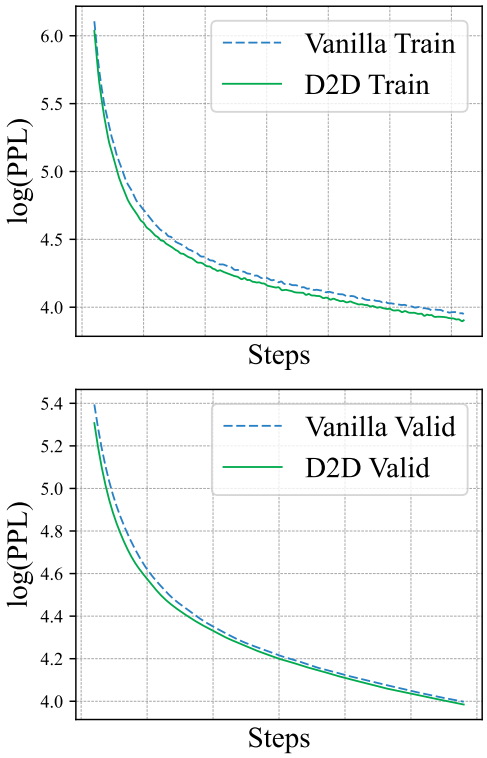

Figure 9: The figure shows the loss curves for the D2D and Vanilla GPT during certain part of pre-training process on both the training and validation datasets.

As shown in Figure 9, D2D, due to its stronger prior knowledge, exhibits a much faster convergence in loss during the early steps of training compared to Vanilla GPT. However, vanilla attention eventually converges to a stage of lower PPL. This trend is more pronounced on the validation set, where the linear attention in D2D shows certain limitations in generalization ability compared to Vanilla GPT.