The Devil in Linear Transformer

⋆Zhen Qin[1], ⋆Xiaodong Han[1], Weixuan Sun[2,3], Dongxu Li[2],
Lingpeng Kong[4,5], Nick Barnes[2], ✉Yiran Zhong[4]
[1]SenseTime Research, [2]Australian National University,
[3]OPPO Research Institute, [4]Shanghai AI Laboratory, [5]The University of Hong Kong
https://github.com/OpenNLPLab/Transnormer

Abstract

Linear transformers aim to reduce the quadratic space-time complexity of vanilla transformers. However, they usually suffer from degraded performances on various tasks and corpora. In this paper, we examine existing kernel-based linear transformers and identify two key issues that lead to such performance gaps: 1) *unbounded gradients* in the attention computation adversely impact the convergence of linear transformer models; 2) *attention dilution* which trivially distributes attention scores over long sequences while neglecting neighbouring structures. To address these issues, we first identify that the *scaling* of attention matrices is the devil in unbounded gradients, which turns out unnecessary in linear attention as we show theoretically and empirically. To this end, we propose a new linear attention that replaces the scaling operation with a normalization to stabilize gradients. For the issue of attention dilution, we leverage a *diagonal attention* to confine attention to only neighbouring tokens in early layers. Benefiting from the stable gradients and improved attention, our new linear transformer model, TRANSNORMER, demonstrates superior performance on text classification and language modeling tasks, as well as on the challenging Long-Range Arena benchmark, surpassing vanilla transformer and existing linear variants by a clear margin while being significantly more space-time efficient. The code is available at TRANSNORMER.

1 Introduction

Transformer models show great performance on a wide range of natural language processing and computer vision tasks (Qin et al., 2022; Sun et al., 2022b; Cheng et al., 2022a,b; Zhou et al., 2022). One issue of the vanilla transformer model lies in its quadratic space-time complexity with respect to the input length. Various prior works attempt to alleviate this inefficiency (Zaheer et al., 2020; Beltagy et al., 2020; Tay et al., 2020a; Kitaev et al., 2020; Child et al., 2019; Liu et al., 2022; Sun et al., 2022b). In this work, we focus on a particular subset of these methods, known as *kernel-based linear transformers* (Choromanski et al., 2020; Wang et al., 2020; Katharopoulos et al., 2020; Peng et al., 2020; Qin et al., 2022) considering their desirable linear space-time complexity.

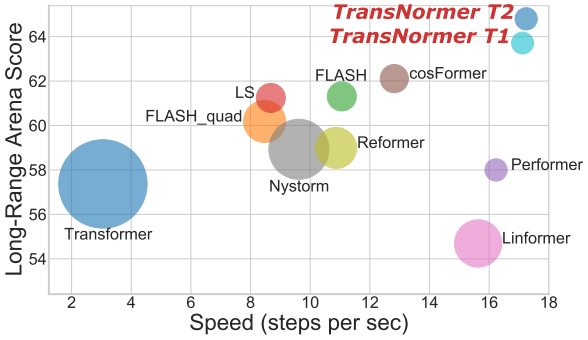

Figure 1: TRANSNORMER has smaller memory footprints (circle sizes) and produces clearly favorable speed (x-axis) and overall scores (y-axis), when evaluated on the challenging Long-Range Arena benchmark than the vanilla transformer and other competing methods.

Despite their space-time efficiency, linear transformers are not always in favor for practical adoption, largely due to the degraded performance than the vanilla model. To address this issue, we take a close look at existing kernel-based linear transformers and identify ***two*** deficiencies that lead to such a performance gap.

Unbounded gradients. Most existing linear transformers inherit attention formulation from the vanilla transformer, which scales attention scores to ensure they are bounded within $[0, 1]$. However, we theoretically show that such a scaling strategy renders unbounded gradients for linear transformer models. As a result, the unbounded gradients em-

⋆Equal contribution. ✉ The corresponding author (Email: *zhongyiran@gmail.com*). This work was done when Weixuan Sun and Yiran Zhong were in the SenseTime Research.

Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 7025 - 7041
December 7-11, 2022

pirically lead to unstable convergence as our preliminary experiments suggest.

Attention dilution. Previous works (Titsias, 2016; Jang et al., 2016; Gao and Pavel, 2017; Qin et al., 2022; Sun et al., 2022b,a) suggest that in vanilla transformer, softmax attention maps tend to be local. In contrast, as shown in Fig 2, we observe that linear transformers often trivially distribute attention scores over the entire sequence even in early layers. Due to this issue, which we refer as *attention dilution*, important local information is less well preserved in linear models, resulting in inferior performance. This negative impact of attention dilution is also evidenced by the performance drop in our controlled experiments if partly replacing vanilla attention in transformer layers with linear attention ones.

To mitigate these issues, we propose a linear transformer model, called TRANSNORMER, which shows better performance than vanilla transformer on a wide range of task while being significantly faster during runtime, as shown in Fig. 1.

To avoid the unbounded gradients, we introduce NORMATTENTION, which gets rid of scaling over attention matrices while appending an additional normalization only *after* the attention layer. The choice of the normalization operator is unrestricted, for example, LayerNorm (Ba et al., 2016) or RMSNorm (Zhang and Sennrich, 2019) both serve the purpose. We show empirical results demonstrating that with NORMATTENTION, the gradients are more stable during training, which in turn leads to more consistent convergence.

To alleviate the attention dilution issue, we modify the vanilla attention and allow each token to only attend to its neighbouring tokens, resulting in a *diagonal* attention. To mimic the behaviors on local semantics of the vanilla transformer, we employ the diagonal attention on early layers while using NORMATTENTION for later ones. In this way, we encourage the model to capture both local and global language context. Note that our diagonal attention can be efficiently computed such that the overall linear space-time complexity of TRANSNORMER is preserved.

We perform extensive experiments on standard tasks, where TRANSNORMER demonstrates *lower language modeling perplexities* on WikiText-103 and overall *higher text classification accuracy* on GLUE than vanilla model and other competing methods. In addition, on the challenging Long-Range Arena benchmark, TRANSNORMER also shows favorable results while being *faster and more scalable* with longer inputs during both training and inference time.

2 Background and related work

We first briefly review vanilla transformer (Vaswani et al., 2017) and its efficient variants. The key component of transformers is the self-attention, which operates on query $\mathbf{Q}$, key $\mathbf{K}$ and value $\mathbf{V}$ matrices; each of them is the image of a linear projection taking $\mathbf{X} \in \mathbb{R}^{n \times d}$ as input:

$$\mathbf{Q} = \mathbf{X}\mathbf{W}_Q, \mathbf{K} = \mathbf{X}\mathbf{W}_K, \mathbf{V} = \mathbf{X}\mathbf{W}_V \in \mathbb{R}^{n \times d}, \quad (1)$$

with n the input length, d the hidden dimension. The output $\mathbf{O} \in \mathbb{R}^{n \times d}$ is formulated as:

$$\mathbf{O} = \mathrm{Softmax}(\mathbf{Q}\mathbf{K}^\top / \sqrt{d})\mathbf{V}, \quad (2)$$

where the $\mathrm{Softmax}(\cdot)$ step renders quadratic space-time complexity with respect to the input length, making it prohibitive for vanilla transformer to scale to long input sequences. To address this issue, numerous efficient transformers have been explored in the literature. These methods can be generally categorized into two families, *i.e., pattern based methods* and *kernel based methods*.

Pattern based methods (Zaheer et al., 2020; Beltagy et al., 2020; Tay et al., 2020a; Kitaev et al., 2020; Child et al., 2019) sparsify the attention calculation with handcrafted or learnable masking patterns. Kernel-based methods adopt kernel functions to decompose softmax attention, which reduces the theoretical space-time complexity to linear. In this paper, we refer the kernel-based variants as linear transformers for simplicity.

In the kernel-based methods (Choromanski et al., 2020; Katharopoulos et al., 2020; Peng et al., 2020; Qin et al., 2022; Zheng et al., 2022; Wang et al., 2020), a kernel function $\phi(\cdot)$ maps queries and keys to their hidden representations. Then the output of the linear attention can be rewritten as:

$$\begin{aligned} \mathbf{O} &= \mathbf{\Delta}^{-1}\phi(\mathbf{Q})[\phi(\mathbf{K})^\top \mathbf{V}], \\ \mathbf{\Delta} &= \mathrm{diag}(\phi(\mathbf{Q})[\phi(\mathbf{K})^\top \mathbf{1}_n]). \end{aligned} \quad (3)$$

where the product of keys and values are computed to avoid the quadratic $n \times n$ matrix. Existing methods mainly differ in the design of kernel functions. For example, Choromanski et al. (2020) and Katharopoulos et al. (2020) adopt activation function $1 + elu$ to process query and key. Wang

et al. (2020) assumes attention matrices are low-rank. Peng et al. (2020) and Zheng et al. (2022) approximate softmax under constrained theoretical bounds. Qin et al. (2022) propose a linear alternative to the attention based on empirical properties of the softmax function.

These methods focus on either approximating or altering the softmax operator while preserving its properties. Compared with the vanilla transformer, these methods often trade performance for efficiency, usually resulting in worse task performance. In this paper, we argue that there are two essential reasons leading to such a performance gap, discussed in detail as follows.

3 The devil in linear attention

In this section, we motivate the design principles of TRANSNORMER by providing theoretical evidence for the unbounded gradients, and empirical results showing the adverse influence of attention dilution.

3.1 Unbounded gradients

Few work on linear transformers analyzes their gradients during training. Our first key observation is that *kernel-based linear attention suffer from unbounded gradients, causing unstable convergence during training*. In the following, we highlight the main theoretical results while referring readers to Appendix D for the full derivation.

Consider a self-attention module, either vanilla or linear attention. Its attention matrix $\mathbf{P} \in \mathbb{R}^{n\times n}$ can be represented in the following unified form [1]:

$$p_{ij} = \frac{f(s_{ij})}{\sum_{k=1}^{n} f(s_{ik})}, f : \mathbb{R} \to \mathbb{R}. \tag{4}$$

Vanilla and linear attention differ mainly in their computation of token-wise similarities s_{ij}[2]. In vanilla attention, s_{ij} is computed as:

$$s_{ij} = \mathbf{q}_i^\top \mathbf{k}_j / \sqrt{d}, f(x) = \exp(x), \tag{5}$$

while for linear attentions, s_{ij} can be decomposed using a kernel function ϕ, such that:

$$s_{ij} = \phi(\mathbf{q}_i)^\top \phi(\mathbf{k}_j), f(x) = x. \tag{6}$$

Given the above definitions, the gradients of the attention matrix $\mathbf{P}$ is derived as:

$$\frac{\partial p_{ij}}{\partial s_{ik}} = \frac{f'(s_{ik})}{f(s_{ik})} \left(1_{j=k} p_{ij} - p_{ij} p_{ik}\right) \tag{7}$$

Therefore, for the vanilla attention, the partial derivative $\frac{\partial p_{ij}}{\partial s_{ik}}$ is:

$$\begin{aligned} f'(x) &= \exp(x) = f(x) \\ \frac{\partial p_{ij}}{\partial s_{ik}} &= 1_{j=k} p_{ij} - p_{ij} p_{ik} \\ &= \begin{cases} p_{ik} - p_{ik} p_{ik} \in [0, 1/4] & j = k \\ -p_{ij} p_{ik} \in [-1/4, 0] & j \neq k \end{cases} \end{aligned} \tag{8}$$

and it is bounded as:

$$\left| \frac{\partial p_{ij}}{\partial s_{ik}} \right| \leq \frac{1}{4}. \tag{9}$$

However, for linear attentions, we have:

$$\begin{aligned} f'(x) &= 1 \\ \frac{\partial p_{ij}}{\partial s_{ik}} &= \frac{1}{s_{ik}} \left(1_{j=k} p_{ij} - p_{ij} p_{ik}\right) \\ &= \begin{cases} \frac{1}{s_{ik}} \left(p_{ik} - p_{ik} p_{ik}\right) & j = k \\ \frac{1}{s_{ik}} (-p_{ij} p_{ik}) & j \neq k \end{cases} \end{aligned} \tag{10}$$

and[3]

$$\left| \frac{\partial p_{ij}}{\partial s_{ik}} \right| \leq \frac{1}{4|s_{ik}|}. \tag{11}$$

Since $|s_{ik}|^{-1} = |\phi(\mathbf{q}_i)\phi(\mathbf{k}_j)^\top|^{-1}$ can be arbitrarily large, the gradient of linear attention has no upper bound. On the other hand, we can also show that the gradient of linear attention has no lower bound[4]:

Proposition 3.1. *$\forall M > 0$, there exists $\mathbf{q}_i, \mathbf{k}_j \in \mathbb{R}^d, j = 1, \ldots, n$, such that:*

$$\left| \frac{\partial p_{ij}}{\partial s_{ik}} \right| > M. \tag{12}$$

The unbounded gradients lead to less stable optimization and worse convergence results in our preliminary studies.

3.2 Attention dilution

It is a known property of vanilla attention to emphasize on neighbouring tokens (Titsias, 2016; Qin et al., 2022). However, this property does not directly inherit to the linear transformer variants.

To quantify the attention dilution issue, we introduce a metric called *locally accumulated attention* score, which measures how much attention scores are distributed within the local neighbourhood of a particular token.

For an input sequence of length N, consider a local neighbourhood $\{x_{start}, ..., x_i ..., x_{end}\}$ *centering* around token x_i of total length $r \cdot N$, with r

[1] Here we assume that $f(s_{ij}) \geq 0$, the conclusion is satisfied in most cases.

[2] Note that s_{ij} is not directly computed in linear attention, but can still be represented in this unified form, see Appendix D for more detailed derivation

[3] A detailed proof of the upper bound can be found at Appendix B.

[4] The proof can be found in Appendix C.

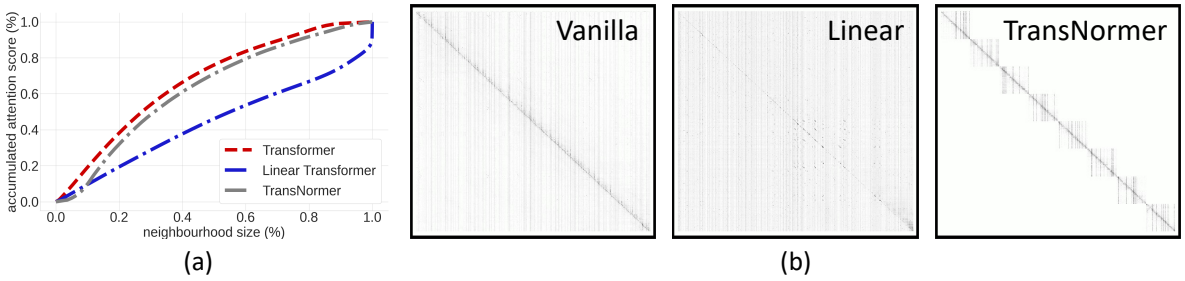

Figure 2: (a): Comparison of locally accumulated attention scores of different transformer variants. The x-axis denotes ratio of neighbourhood size relative to the input length; the y-axis denotes accumulated attention scores inside this neighbourhood for the centering token. The curve for the vanilla transformer model increases more sharply, indicating that the attention scores are more concentrated. Our model greatly alleviates the attention dilution issue for linear models. (b): Qualitative comparison of attention matrices in early model layers. The proposed TRANSNORMER produces more similar patterns to the original vanilla transformer, benefiting to better capture local-global language context, while the linear model suffers clearly from the issue of attention dilution and gets distracted by distant tokens in early layers.

the ratio relative to the total input, the locally accumulated attention score for token x_i is defined as $l(i, r, N) = p_{i,start} + ... + p_{i,end}$. A higher score indicates the particular attention layer concentrates on the local neighbourhood, while a lower score tends to indicate the issue of attention dilution, where scores are distributed more evenly to local and distant tokens. For example, $l(i, 0.4, N) = 0.6$ means that that 40% of the neighbors around i'th token contribute 60% of the attention score.

In Fig. 2 (a), we compare locally accumulated attention scores (y-axis) for vanilla transformer and linear transformer, with varying sizes of neighbourhood by ratio (x-axis). We show the average score over each position across the entire sequence. It can be seen that the area under the vanilla model curve is significantly larger than that of the linear model. This provides evidence that the vanilla attention is more concentrated locally, while the linear transformer suffers from the issue of attention dilution. This is further qualitatively supported by Fig. 2 (b), where the attention maps for vanilla model are more concentrated than the linear model.

4 Method

Based on the aforementioned observations, we propose a new linear transformer network called TRANSNORMER that addresses the above two limitations of current linear transformers. The overall architecture is shown in Fig. 3.

4.1 The overall architecture

Vanilla attention suffers less in attention dilution while linear attention is more efficient and scalable on longer sequences. This motivate us to design a method that exploits the best of the both worlds by using these mechanisms in combined.

Specifically, our network consists of two types of attention: DIAGATTENTION for the early stage of the model and NORMATTENTION for the later stage. The former addresses the attention dilution issue and the later aims to stabilize training gradients. Note that by properly reshaping the inputs, the diagonal attention can be efficiently computed in linear space-time, thus preserving the overall linear complexity.

4.2 NORMATTENTION

Table 1: **Ablation of linear attention with scaling operation.** Directly removing scaling operation *i.e.,* the denominator in Eq. 4, leads to significant performance drop. Our normalization strategy achieves better result.

method	ppl(val)
$1 + elu$	4.98
$1 + elu$ w/o scaling	797.08
NORMATTENTION	4.94

As proved in Sec. 3, the scaling operation, *i.e.,* the denominator in Eq. 4, in the linear transformers hinder the optimization due to the unbounded gradients. To solve this issue, we propose to remove the scaling operation in the linear transformers. However, as shown in Table. 1, directly removing the scaling operation leads to critical performance drop since the attention map becomes unbounded in the forward pass. Therefore, an alternative is required to bound both attention maps during forward and their gradients during backward passes in linear

attentions.

Our proposed solution is simple yet effective. Given a linear attention, the attention without scaling can be formulated as:

$$\mathbf{O} = \mathbf{Q}(\mathbf{K}^\top \mathbf{V}). \tag{13}$$

We empirically find that we can apply an arbitrary normalization on this attention to bound it, which leads to our NORMATTENTION as:

$$\mathbf{O}_{\text{norm}} = \text{XNorm}(\mathbf{Q}(\mathbf{K}^\top \mathbf{V})), \tag{14}$$

where the XNorm can be Layernorm(Ba et al., 2016) or RMSNorm (Zhang and Sennrich, 2019) and *etc.* We use the RMSNorm in our experiments as it is slightly faster than other options.

It can be proved that the gradients of NORMATTENTION is bounded by[5]:

$$\left| \frac{\partial \mathcal{L}}{\partial s_{ij}} \right| \leq \frac{3 c_1 c_2 d}{2\sqrt{\epsilon}} < \infty, \tag{15}$$

where $\mathcal{L}$ is the loss function, ϵ is the small constant that used in RMSNorm, d is the embedding dimension and

$$\begin{aligned} c_1 &= \max_{i=1}^{n} \|\nabla_{\mathbf{O}_i} \mathcal{L}\|_2 < \infty \\ c_2 &= \max_{i=1}^{n} \|\mathbf{V}_i\|_2 < \infty \end{aligned} \tag{16}$$

To demonstrate the gradients stability of the NORMATTENTION, we compare the relative standard deviation of gradients during each training iterations to other linear transformers and vanilla transformer. Specifically, we train our model for 50k iterations with RoBERTa architecture on the WikiText103 (Merity et al., 2017) and obtain the relative standard deviation of all iterations' gradients. As shown in Table 2, existing linear methods (Choromanski et al., 2020; Katharopoulos et al., 2020) have substantially higher deviations compared to vanilla attention, which leads to inferior results. The NORMATTENTION produces more stable gradients, which validates the effectiveness of our method.

4.3 DIAGATTENTION

To better understand the design principles, we show in Table 3 that by replacing partial layers of linear transformers with vanilla attention, the performance on language modeling is evidently improved. The results also suggest that capturing more local information in early layers are more helpful than otherwise.

[5] The full derivation can be found in Appendix D.

Table 2: **Relative standard deviation of training gradients over 50k iterations.** Our proposed NORMATTENTION provides more stable gradients which are closer to vanilla transformer.

method	Relative Standard Deviation
$1 + elu$ (Katharopoulos et al., 2020)	0.58
Performer(Choromanski et al., 2020)	0.47
Vanilla(Vaswani et al., 2017)	0.25
NORMATTENTION	0.20

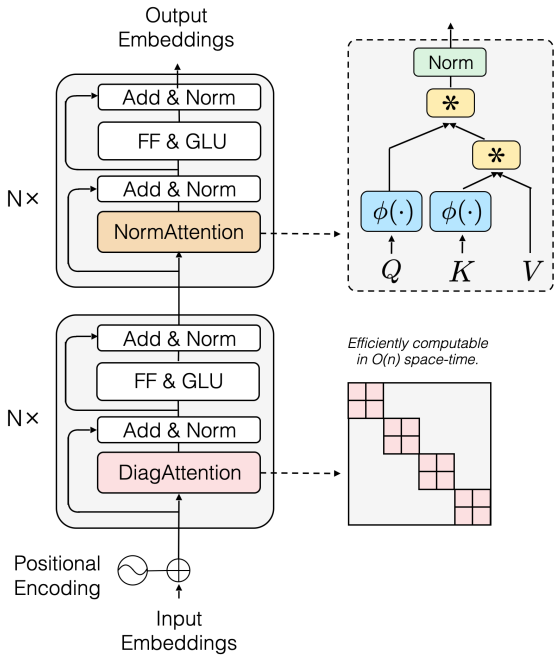

Figure 3: Architecture overview of the proposed TRANSNORMER. In the early stages, we leverage DIAGATTENTION, where attention is only calculated inside the blocks to enforce neighbouring focus. In late stages, NORMATTENTION assists to obtain a more stable gradients in linear complexity.

To this end, we leverage *none-overlapped* block-based strategy to reduce the space-time complexity of the vanilla attention. Based on the observation in Fig. 2, we utilize a strict diagonal blocked pattern to constraint the attention in a certain range. Since the attentions are calculated inside each block, the computation complexity of our diagonal attention is $O(nwd)$, where n is sequence length , w is the block size and d is feature dimension. When $d \ll n$, the complexity scales linearly respect to the sequence length n. In subsequent sections, we use DIAGATTENTION to refer to Diagonal attention.

We empirically find that applying DIAGATTENTION to the later stages of a model hurts the performance as shown in Table. 9. It indicates that the

Table 3: **Ablation on attention dilution issue.** We implement all structures under the same setting: Vanilla (Vaswani et al., 2017), $1 + elu$ (Katharopoulos et al., 2020).

Early layers	**Later layers**	**ppl** (val)
$1 + elu$	$1 + elu$	4.98
$1 + elu$	Vanilla	3.90
Vanilla	$1 + elu$	3.76

model requires a global field of view in the later layers, which also justifies our choices of NORMATTENTION in later layers of TRANSNORMER.

5 Experiments

In this section, we compare our method to other linear transformers and the vanilla transformer on autoregressive language modeling, bidirectional language modeling as well as the Long Range Arena benchmark (Tay et al., 2020b). We also provide an extensive ablation study to vindicate our choice in designing the TRANSNORMER.

We validate our method on two variants of the TRANSNORMER. The TRANSNORMER T1 uses the ReLA attention (Zhang et al., 2021) in the DIAGATTENTION and the *elu* as the activation function in the NORMATTENTION. The TRANSNORMER T2 uses the `Softmax` attention (Vaswani et al., 2017) in the DIAGATTENTION and the *1+elu* as the activation function in the NORMATTENTION.

For experiments, we first study the autoregressive language modeling on WikiText-103 (Merity et al., 2017) in section 5.2. Then in section 5.2 we test our method on bidirectional language modeling, which is pre-trained on WikiText-103 (Merity et al., 2017) and then fine-tuned on several downstream tasks from the GLUE benchmark (Wang et al., 2018). Finally, we test TRANSNORMER on the Long-Range Arena benchmark (Tay et al., 2020b) to evaluate its ability in modeling long-range dependencies and efficiencies in section 5.2.

5.1 Settings

We implement our models in the *Fairseq* framework (Ott et al., 2019) and train them on 8 V100 GPUS. We use the same training configuration for all competitors and we list detailed hyper-parameters in Appendix F. We choose the FLASH-quad, FLASH (Hua et al., 2022), Transformer-LS (Zhu et al., 2021), Performer (Choromanski et al., 2020), 1+elu (Katharopoulos et al., 2020) as our main competing methods.

For the autoregressive language modeling, we use 6 decoder layers (10 layers for the FlASH/FLASH-quad) as our base model and all models are trained on the WikiText-103 dataset (Merity et al., 2017) for 100K steps with a learning rate of 0.005. We use the perplexity (PPL) as the evaluation metric.

For the bidirectional language modeling, we choose the RoBERTa base (Liu et al., 2019) for all methods. It consists of 12 encoder layers (24 layers for the FLASH and FLASH-quad to match the number of parameters). All models are pre-trained on the WikiText-103 (Merity et al., 2017) for 50K steps with lr=0.005 and fine-tuned on the GLUE dataset (Wang et al., 2018). We use different learning rates among 1e-5, 3e-5, 6e-5, 1e-4 and choosing the best result after fine-tuning for 3 epochs.

For the Long-Range Arena benchmark, to make sure it reflect the practical speed in Pytorch platform, we re-implement the benchmark in Pytorch. We adopt the same configuration from the Skyformer (Chen et al., 2021) and make sure all models have a similar parameter size. We use the same training hyper parameters for all models as well.

Table 4: **Quantitative results in autoregressive language modeling.** The best result is highlighted with **bold** and the second with underlined. The smaller the better for the PPL metric. LS stands for transformer-LS.

Method	PPL (val)	PPL (test)	Params (m)
Vanilla	29.63	**31.01**	156.00
LS	32.37	32.59	159.46
FLASH-quad	31.88	33.50	153.51
FLASH	33.18	34.63	153.52
1+elu	32.63	34.25	156.00
Performer	75.29	77.65	156.00
TRANSNORMER T1	29.89	31.35	155.99
TRANSNORMER T2	**29.57**	**31.01**	155.99

5.2 Results

Autoregressive language modeling We report the results in Table 4. It can be found that both TRANSNORMER variants get comparable or better perplexity to the vanilla attention and outperform all existing linear models with a clear margin. For example, compared to previous state-of-the-art linear methods on validation set(Hua et al., 2022) and test set(Zhu et al., 2021), TRANSNORMER T2 achieves substantially lower perplexity by 2.31 and 1.58 respectively. It demonstrates the effectiveness of our method in causal models.

Table 5: **Quantitative results of the GLUE benchmark.** MNLI is reported by the match/mismatch splits. MRPC is reported by F1 score. CoLA is reported by Matthews correlation coefficient. All the other tasks are measured by the accuracy. LS stands for transformer-LS. The best result is highlighted with **bold** and the second with underlined. The larger the better for all metrics. "-" means unconverged.

Method	MNLI	QNLI	QQP	SST-2	MRPC	CoLA	AVG	Params (m)
Vanilla	79.37/79.07	87.79	88.04	90.25	88.35	38.63	78.79	124.70
FLASH-quad	78.71/79.43	86.36	88.95	90.94	81.73	41.28	78.20	127.11
FLASH	79.45/80.08	87.10	88.83	90.71	82.50	29.40	76.87	127.12
LS	77.01/76.78	84.86	86.85	90.25	82.65	40.65	77.01	128.28
Performer	58.85/59.52	63.44	79.10	81.42	82.11	19.41	63.41	124.70
1+elu	74.87/75.37	82.59	86.9	87.27	83.03	-	70.00	124.0
TRANSNORMER T1	79.06/79.93	87.00	88.61	91.17	84.50	45.38	**79.38**	124.67
TRANSNORMER T2	77.28/78.53	85.39	88.56	90.71	85.06	45.90	78.78	124.67

Table 6: **Quantitative results on the Long-Range Arena benchmark.** The best result is highlighted with **bold** and the second with underlined. The larger the better for all metrics.

Model	Text	ListOps	Retrieval	Pathfinder	Image	AVG.
Transformer	61.95	38.37	80.69	65.26	40.57	57.37
Kernelized Attention	60.22	38.78	81.77	70.73	41.29	58.56
Nystromformer	64.83	38.51	80.52	69.48	41.30	58.93
Linformer	58.93	37.45	78.19	60.93	37.96	54.69
Informer	62.64	32.53	77.57	57.83	38.10	53.73
Performer	64.19	38.02	80.04	66.30	41.43	58.00
Reformer	62.93	37.68	78.99	66.49	48.87	58.99
BigBird	63.86	39.25	80.28	68.72	43.16	59.05
Skyformer	64.70	38.69	82.06	70.73	40.77	59.39
LS	66.62	40.30	81.68	69.98	47.60	61.24
cosFormer	67.70	36.50	83.15	71.96	51.23	62.11
FLASH-quad	64.10	**42.20**	83.00	63.28	48.30	60.18
FLASH	64.10	38.70	**86.10**	70.25	47.40	61.31
TRANSNORMER T1	66.90	41.03	83.11	75.92	**51.60**	63.71
TRANSNORMER T2	**72.20**	41.60	83.82	**76.80**	49.60	**64.80**

Table 7: **Speed comparison on Long-Range Arena benchmark.** We mark it with a dash if a method exhausts GPU memory. The higher the better for all metrics. The **1K**,...,**5K** represent the input sequence length.

	Inference Speed(steps per sec)					Train Speed(steps per sec)				
model	**1K**	**2K**	**3K**	**4K**	**5K**	**1K**	**2K**	**3K**	**4K**	**5K**
Transformer	39.06	10.05	-	-	-	15.34	3.05	-	-	-
FLASH-quad	44.64	16.45	9.40	6.54	5.39	19.84	8.47	5.19	3.59	2.92
FLASH	40.32	23.15	16.89	14.04	13.16	20.49	11.06	8.47	7.23	6.93
LS	32.05	17.36	12.14	10.16	9.06	15.43	8.68	6.28	5.24	4.76
Performer	104.17	56.82	42.37	33.78	31.25	28.41	16.23	12.02	10.04	9.06
cosFormer	86.21	46.30	32.47	27.47	25.00	22.94	12.82	9.19	7.79	7.14
Linformer	104.17	58.14	40.32	31.25	26.32	27.17	15.63	11.26	8.77	7.42
Reformer	78.13	38.46	26.04	19.84	16.23	20.16	10.87	7.46	5.69	4.70
Nystorm	58.14	38.46	29.07	23.81	20.33	14.12	9.62	7.46	6.11	5.26
TRANSNORMER T1	113.64	65.79	46.30	39.06	35.71	28.41	17.12	12.76	10.87	10.12
TRANSNORMER T2	119.05	65.79	47.17	39.68	36.23	29.41	17.24	12.95	10.96	10.16

Bidirectional language modeling We show our bidirectional results on the GLUE benchmark in Table. 5. Our method achieves superior performance to all the competing methods in average. On three tasks, *i.e.,* SST-2, MRPC, CoLA, TRANSNORMER reports comprehensively better results than all competing linear methods, such as 4.62 higher on CoLA. Further, one of our variants *i.e.,* TRANSNORMER T1, even outperforms the vanilla attention with a notable margin. It proves the effectiveness of our method in bidirectional language modeling.

Long Range Arena Benchmark The results before the transformer Long-short (abbr. LS) are taken from the Skyformer (Chen et al., 2021). As shown in Table. 6, we achieve either first or second places across all five tasks. In terms of overall results, both TRANSNORMER variants (T1,T2) outperform all other competing methods including vanilla transformer (Vaswani et al., 2017), which

validates our capability to encode long sequences.

5.3 Speed comparison

We compare the training and inference speed of the TRANSNORMER with other methods. For a fair and comprehensive comparison, we follow exactly the same configurations of the Skyformer(Chen et al., 2021) and report step per second under different sequence lengths. Timing is conducted on a Nvidia A6000 GPU with 48G GPU memory. Table. 7 suggests that the vanilla transformer is substantially slow and exhausts GPU memory with sequence longer than 3k. Compared to other efficient transformers, our TRANSNORMER achieves faster speed with comparable GPU memory footprints, while competing efficient methods all report worse results compared to our TRANSNORMER. For instance, compared to FLASH-quad (Hua et al., 2022) that achieves previous best linear results on both autoregressive and bidirectional benchmarks, our model performs over 300% faster during training and 150% faster during inference.

5.4 Ablation study

In this section, we justify our design choice of the TRANSNORMER, including , the selection of the FFN module, and the size of the attention block in DIAGATTENTION. We use the PPL from the Roberta pre-training stage as our evaluation metric.

Table 8: **Ablation of the proportion of the attentions.** We empirically find that the balanced structure achieves the best result. We abbreviate the DIAGATTENTION as BlockAtt and NORMATTENTION as NormAtt.

Early stage BlockAtt	Later stage NormAtt	T1 ppl(val)	T2 ppl(val)
0	12	4.23	4.48
3	9	4.13	3.83
6	6	**3.82**	**3.81**
9	3	3.87	3.86
12	0	4.75	4.66

Table 9: **Ablation of the order of two proposed attention.** Using DIAGATTENTION in the early stage achieves better results than using it on later stage.

Early stage	Later stage	T1 ppl(val)	T2 ppl(val)
NormAtt	BlockAtt	4.13	4.21
BlockAtt	NormAtt	**3.82**	**3.81**

Structure design As aforementioned, we empirically choose the first 6 layers as the early stage of the model and the rest as the later stage. We provide the designing ground for this choice in Table. 8. It can be also observed that either choosing the DIAGATTENTION or NORMATTENTION for the entire model will lead to inferior performance. We also provide the ablation results of swapping the order of the DIAGATTENTION and the NORMATTENTION in Table. 9. Using DIAGATTENTION in the early stage achieves significantly better results than using it on later stage. It further proves our claim that the early stage focuses on neighbouring tokens while the later stage needs long-range attentions.

Table 10: **Ablation of the selection of the FFN modules.** The GLU leads to better results.

FFN type	T1 ppl(val)	T2 ppl(val)
FFN	3.93	3.93
GLU(ours)	**3.82**	**3.81**

FFN module We ablate the selection of the FFN modules in Table. 10. Compared with the traditional FFN (Vaswani et al., 2017), the GLU (Shazeer, 2020) achieves better results.

Table 11: **Ablation of on block sizes in the DIAGATTENTION.** The larger block size the better results.

Block size	T1 ppl(val)	T2 ppl(val)
32	3.92	3.90
64	3.82	3.81
128	**3.72**	**3.69**

Block size From the Table. 11, we observe clear performance improvements with increased block sizes. However, since the complexity of the DIAGATTENTION is $O(nwd)$, larger block size w leads to heavier computational overhead. We choose a block size as 64 as a trade-off between performance and computational cost.

Combination of attentions Finally, we study the effect that whether we should use both attentions in one layer. In particular, we compare either to 1) use DIAGATTENTION and NORMATTENTION sequentially in a layer with different orders; or to 2) use them in parallel in each attention layer and then concatenate their embedding output. Table. 12 shows that we should not use these attentions sequentially within a layer and apply them in parallel will double the computation complexities without improving the performance.

6 Conclusion

In this paper, we identified two key issues that cause the inferior performance of existing linear transformer models: 1) unbounded gradients; 2)

Table 12: **Ablation of the combination of two proposed attention.** In first two rows, the two attention layers appear in an interleaved manner. D for the DIAGATTENTION and N for the NORMATTENTION.

approach	T1 ppl(val)	T2 ppl(val)
altering D→N	4.19	4.23
altering N→D	4.11	4.21
parallel	**3.77**	3.82
TRANSNORMER	3.82	**3.81**

attention dilution. For the former issue, we proposed a new NORMATTENTION to stabilize the training gradients. For the latter, we develop DIAGATTENTION to force the model concentrate attention in neighbouring tokens. The resultant model TRANSNORMER marries the strength of the vanilla transformers and the linear transformers, outperforming competing linear transformers on both autoregressive and bidirectional language modeling, text classification tasks and the challenging Long-range arena benchmark.

Limitations

In this paper, we identified two main issues of current linear transformers and provided a comprehensive analysis in natural language processing tasks. However, with the booming development of vision transformers, whether they share the same issues of linear NLP transformers is yet to be discovered. We will validate our method on the linear vision transformers in our future work.

Ethics Statement

The proposed technique is beneficial to develop large-scale environment-friendly language models by reducing computing resource demand. Corpus used to train the model is from public web sources, which may contain biased, explicit or improper content. Further assessment and regulation have to be in-place before deploying the model in practice.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization.

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Yifan Chen, Qi Zeng, Heng Ji, and Yun Yang. 2021. Skyformer: Remodel self-attention with gaussian kernel and nyström method. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*.

Xuelian Cheng, Huan Xiong, Deng-Ping Fan, Yiran Zhong, Mehrtash Harandi, Tom Drummond, and Zongyuan Ge. 2022a. Implicit motion handling for video camouflaged object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13864–13873.

Xuelian Cheng, Yiran Zhong, Mehrtash Harandi, Tom Drummond, Zhiyong Wang, and Zongyuan Ge. 2022b. Deep laparoscopic stereo matching with transformers. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 464–474. Springer.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. 2020. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*.

Bolin Gao and Lacra Pavel. 2017. On the properties of the softmax function with application in game theory and reinforcement learning. *arXiv preprint arXiv:1704.00805*.

Weizhe Hua, Zihang Dai, Hanxiao Liu, and Quoc V Le. 2022. Transformer quality in linear time. *arXiv preprint arXiv:2202.10447*.

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. 2020. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, pages 5156–5165. PMLR.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Zexiang Liu, Dong Li, Kaiyue Lu, Zhen Qin, Weixuan Sun, Jiacheng Xu, and Yiran Zhong. 2022. Neural architecture search on efficient transformers and beyond. *arXiv preprint arXiv:2207.13955*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. *5th International Conference on Learning Representations, ICLR, Toulon, France*.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. *arXiv preprint arXiv:1904.01038*.

Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah Smith, and Lingpeng Kong. 2020. Random feature attention. In *International Conference on Learning Representations*.

Zhen Qin, Weixuan Sun, Hui Deng, Dongxu Li, Yunshen Wei, Baohong Lv, Junjie Yan, Lingpeng Kong, and Yiran Zhong. 2022. cosformer: Rethinking softmax in attention. In *International Conference on Learning Representations*.

Noam Shazeer. 2020. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*.

Jingyu Sun, Guiping Zhong, Dinghao Zhou, Baoxiang Li, and Yiran Zhong. 2022a. Locality matters: A locality-biased linear attention for automatic speech recognition. *arXiv preprint arXiv:2203.15609*.

Weixuan Sun, Zhen Qin, Hui Deng, Jianyuan Wang, Yi Zhang, Kaihao Zhang, Nick Barnes, Stan Birchfield, Lingpeng Kong, and Yiran Zhong. 2022b. Vicinity vision transformer. *arXiv preprint arXiv:2206.10552*.

Yi Tay, Dara Bahri, Liu Yang, Donald Metzler, and Da-Cheng Juan. 2020a. Sparse sinkhorn attention. In *International Conference on Machine Learning*, pages 9438–9447. PMLR.

Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. 2020b. Long range arena: A benchmark for efficient transformers. In *International Conference on Learning Representations*.

Michalis K Titsias. 2016. One-vs-each approximation to softmax for scalable estimation of probabilities. *arXiv preprint arXiv:1609.07410*.

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 *Advances in neural information processing systems*, pages 5998–6008.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355.

Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. 2020. Big bird: Transformers for longer sequences. In *NeurIPS*.

Biao Zhang and Rico Sennrich. 2019. Root Mean Square Layer Normalization. In *Advances in Neural Information Processing Systems 32*, Vancouver, Canada.

Biao Zhang, Ivan Titov, and Rico Sennrich. 2021. Sparse attention with linear units. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6507–6520, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Lin Zheng, Chong Wang, and Lingpeng Kong. 2022. Linear complexity randomized self-attention mechanism. *arXiv preprint arXiv:2204.04667*.

Jinxing Zhou, Jianyuan Wang, Jiayi Zhang, Weixuan Sun, Jing Zhang, Stan Birchfield, Dan Guo, Lingpeng Kong, Meng Wang, and Yiran Zhong. 2022. Audio-visual segmentation. In *European Conference on Computer Vision*.

Chen Zhu, Wei Ping, Chaowei Xiao, Mohammad Shoeybi, Tom Goldstein, Anima Anandkumar, and Bryan Catanzaro. 2021. Long-short transformer: Efficient transformers for language and vision. In *Advances in Neural Information Processing Systems*.

Appendix

A Mathematical Notations

We use bold uppercase letters for matrices($\mathbf{M}$), bold lowercase letters for vectors($\mathbf{m}$), and lowercase letters for scalars(m_{ij}). We represent all vectors as column vectors and denote the ith row of matrix $\mathbf{M}$ by $\mathbf{m}_i^\top$ or $\mathbf{M}_i$. We use $\|.\|_2$ to denote the l_2 norm and $\|.\|_F$ to denote the Frobenius norm of the matrix and the vector.

The main mathematical symbols are input $\mathbf{X} \in \mathbb{R}^{n \times d}$, $\mathbf{Q}$ (**Query**), $\mathbf{K}$ (**Key**) and $\mathbf{V}$

(**Value**), which has the following form:

$$\begin{aligned}
\mathbf{X} &= \begin{bmatrix} \mathbf{x}_1^\top \\ \vdots \\ \mathbf{x}_n^\top \end{bmatrix} \in \mathbb{R}^{n\times d}, \\
\mathbf{Q} &= \begin{bmatrix} \mathbf{q}_1^\top \\ \vdots \\ \mathbf{q}_n^\top \end{bmatrix} = \mathbf{X}\mathbf{W}_Q = \begin{bmatrix} \mathbf{x}_1^\top \mathbf{W}_Q \\ \vdots \\ \mathbf{x}_n^\top \mathbf{W}_Q \end{bmatrix} \in \mathbb{R}^{n\times d}, \\
\mathbf{K} &= \begin{bmatrix} \mathbf{k}_1^\top \\ \vdots \\ \mathbf{k}_n^\top \end{bmatrix} = \mathbf{X}\mathbf{W}_K = \begin{bmatrix} \mathbf{x}_1^\top \mathbf{W}_K \\ \vdots \\ \mathbf{x}_n^\top \mathbf{W}_K \end{bmatrix} \in \mathbb{R}^{n\times d}, \\
\mathbf{V} &= \begin{bmatrix} \mathbf{v}_1^\top \\ \vdots \\ \mathbf{v}_n^\top \end{bmatrix} = \mathbf{X}\mathbf{W}_V = \begin{bmatrix} \mathbf{x}_1^\top \mathbf{W}_V \\ \vdots \\ \mathbf{x}_n^\top \mathbf{W}_V \end{bmatrix} \in \mathbb{R}^{n\times d},
\end{aligned} \tag{17}$$

where $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V \in \mathbb{R}^{d\times d}$.

B Proof of gradients' upper bound

In this part, we will proof the bound in (8) and (10), all we need to prove is:

$$0 \le p_{ik}(1-p_{ik}) \le \frac{1}{4}, 0 \le p_{ij}p_{ik} \le \frac{1}{4}. \tag{18}$$

We adopt the theorem that geometric mean is bounded by arithmetic mean, *i.e.,*

$$\sqrt{ab} \le \frac{a+b}{2} \Longleftrightarrow ab \le \left(\frac{a+b}{2}\right)^2, \forall a, b \ge 0. \tag{19}$$

We take $a = p_{ik}, b = 1 - p_{ik}$ to complete the proof. The first bound can be proven by:

$$0 \le p_{ik}(1-p_{ik}) \le \left(\frac{p_{ik}+1-p_{ik}}{2}\right)^2 = \frac{1}{4}. \tag{20}$$

For the second bound, we first use the fact that:

$$0 \le p_{ij} + p_{ik} \le 1 \Rightarrow p_{ij} \le 1 - p_{ik}. \tag{21}$$

So we have:

$$0 \le p_{ij}p_{ik} \le (1-p_{ik})p_{ik} \le \frac{1}{4}. \tag{22}$$

C Proof of Proposition 3.1

Proof of Proposition 3.1. $\forall \epsilon > 0$ and kernel function ϕ, let[6]:

$$\begin{aligned}
\mathbf{q}_i = \mathbf{k}_j = \phi^{-1}(\mathbf{x}_0), \\
0 < \|\mathbf{x}_0\|_2 \le \sqrt{\epsilon}, i, j = 1, \ldots, n.
\end{aligned} \tag{23}$$

[6] We assume that the image of ϕ contains vectors arbitrary close to $\mathbf{0}$, which is a common case in kernel function.

Then

$$\phi(\mathbf{q}_i) = \phi(\mathbf{k}_j) = \mathbf{x}_0, i, j = 1, ..., n. \tag{24}$$

So

$$s_{ij} = \phi(\mathbf{q}_i)^\top \phi(\mathbf{k}_j) = \mathbf{x}_0{}^\top \mathbf{x}_0 \in (0, \epsilon], \tag{25}$$

and

$$p_{ij} = \frac{s_{ij}}{\sum_{k=1}^n s_{ik}} = \frac{\mathbf{x}_0{}^\top \mathbf{x}_0}{\sum_{k=1}^n \mathbf{x}_0{}^\top \mathbf{x}_0} = \frac{1}{n}. \tag{26}$$

According to (10), we have:

$$\begin{aligned}
\frac{\partial p_{ij}}{\partial s_{ik}} &= \begin{cases} \frac{1}{\mathbf{x}_0{}^\top \mathbf{x}_0}\frac{1}{n}(1-\frac{1}{n}) & j = k \\ -\frac{1}{\mathbf{x}_0{}^\top \mathbf{x}_0}\frac{1}{n^2} & j \ne k \end{cases}, \\
\left|\frac{\partial p_{ij}}{\partial s_{ik}}\right| &= \begin{cases} \frac{1}{\mathbf{x}_0{}^\top \mathbf{x}_0}\frac{1}{n}(1-\frac{1}{n}) & j = k \\ \frac{1}{\mathbf{x}_0{}^\top \mathbf{x}_0}\frac{1}{n^2} & j \ne k \end{cases} \\
&\ge \begin{cases} \frac{1}{\epsilon n}(1-\frac{1}{n}) & j = k \\ \frac{1}{\epsilon n^2} & j \ne k \end{cases}.
\end{aligned} \tag{27}$$

Let $\epsilon \to 0^+$, then $\frac{1}{\epsilon n^2}, \frac{1}{\epsilon n}(1-\frac{1}{n}) \to \infty$, so $\left|\frac{\partial p_{ij}}{\partial s_{ik}}\right| \to \infty$.

□

D Analyze the gradient of each method

In this section, let's consider a one-layer Transformer, for a multi-layer Transformer, we can prove our conclusion using induction.

We begin this section by introducing some mathematical notations.

D.1 Notations

In vanilla attention, we have:

$$\begin{aligned}
\mathbf{S} &= \mathbf{Q}\mathbf{K}^\top \in \mathbb{R}^{n\times n}, \\
\mathbf{P} &= \mathrm{Softmax}(\mathbf{S}) \in \mathbb{R}^{n\times n}, \\
\mathbf{O} &= \mathbf{P}\mathbf{V} \in \mathbb{R}^{n\times d}.
\end{aligned} \tag{28}$$

In linear attention, we have:

$$\begin{aligned}
\mathbf{S} &= \phi(\mathbf{Q})\phi(\mathbf{K})^\top \in \mathbb{R}^{n\times n}, \\
\mathbf{\Delta} &= \mathrm{diag}(\mathbf{S}\mathbf{1}_n) \in \mathbb{R}^{n\times n}, \\
\mathbf{P} &= \mathbf{\Delta}^{-1}\mathbf{S} \in \mathbb{R}^{n\times n}, \\
\mathbf{O} &= \mathbf{P}\mathbf{V} \in \mathbb{R}^{n\times d}.
\end{aligned} \tag{29}$$

Although this term is not calculated in linear attention, we discuss it conceptually. Note that

the above formulations can be unified into the following form [7]:

$$\begin{aligned}\mathbf{S} &= f(\psi(\mathbf{Q})\psi(\mathbf{K})^\mathsf{T}) \in \mathbb{R}^{n\times n},\\ \mathbf{\Delta} &= \mathrm{diag}(\mathbf{S}\mathbf{1}_n) \in \mathbb{R}^{n\times n},\\ \mathbf{P} &= \mathbf{\Delta}^{-1}\mathbf{S} \in \mathbb{R}^{n\times n},\\ \mathbf{O} &= \mathbf{P}\mathbf{V} \in \mathbb{R}^{n\times d},\end{aligned} \tag{30}$$

where in vanilla attention, we have:

$$\psi(\mathbf{x}) = \mathbf{x}, f(x) = \exp(x), \tag{31}$$

and in linear attention, we have:

$$\psi(\mathbf{x}) = \phi(\mathbf{x}), f(x) = x. \tag{32}$$

In NORMATTENTION, we have:

$$\begin{aligned}\mathbf{S} &= \phi(\mathbf{Q})\phi(\mathbf{K})^\mathsf{T} \in \mathbb{R}^{n\times n},\\ \mathbf{T} &= \mathbf{S}\mathbf{V} \in \mathbb{R}^{n\times d},\\ \mathbf{O} &= \mathrm{RMSNorm}(\mathbf{T})\\ &\triangleq \begin{bmatrix}\mathrm{RMSNorm}(\mathbf{t}_1)^\mathsf{T}\\ \vdots\\ \mathrm{RMSNorm}(\mathbf{t}_n)^\mathsf{T}\end{bmatrix} \in \mathbb{R}^{n\times d},\end{aligned} \tag{33}$$

where RMSNorm is defined as follows:

Definition D.1.

$$\begin{aligned}\mathrm{RMSNorm}(\mathbf{x}) &= \frac{\mathbf{x}}{\sqrt{\sigma^2+\epsilon}},\\ \sigma^2 &= \frac{\sum_{i=1}^d x_i^2}{d},\\ \epsilon &> 0,\\ \mathbf{x} &\in \mathbb{R}^d.\end{aligned} \tag{34}$$

In the subsequent discussion, we define gradient $\nabla_{\mathbf{M}}\mathcal{L}$ as:

Definition D.2.

$$[\nabla_{\mathbf{M}}\mathcal{L}]_{ij} = \frac{\partial \mathcal{L}}{\partial m_{ij}}, \tag{35}$$

where $\mathcal{L}$ stands for loss function, $\mathbf{M}$ *is a parameter matrix.*

Then we define the mapping h as:

Definition D.3.

$$\begin{aligned}h : \mathbb{R}^{n\times m} \to \mathbb{R}, h(\mathbf{X}) &= \max_{i=1}^n \|\mathbf{X}_i\|_2,\\ \mathbf{X} &\in \mathbb{R}^{n\times m}.\end{aligned} \tag{36}$$

[7] Here, the function $f(\mathbf{X})$ is applied element-wise to the matrix $\mathbf{X} \in \mathbb{R}^{n\times m}$, that is, $[f(\mathbf{X})]_{ij} = [f(x_{ij})]$

The mapping h has the following property:

Proposition D.4. $\forall \mathbf{X} \in \mathbb{R}^{n\times m}, \mathbf{Y} \in \mathbb{R}^{r\times m}$, *we have:*

$$h(\mathbf{X}\mathbf{Y}^\top) \le \sqrt{r}h(\mathbf{X})h(\mathbf{Y}). \tag{37}$$

Proof. Since

$$\begin{aligned}[\mathbf{X}\mathbf{Y}^\top]_{ij} &= \mathbf{X}_i[\mathbf{Y}_j]^\top\\ &\le \|\mathbf{X}_i\|_2\|\mathbf{Y}_j\|_2\\ &\le h(\mathbf{X})h(\mathbf{Y}),\end{aligned} \tag{38}$$

so

$$\begin{aligned}\|[\mathbf{X}\mathbf{Y}^\top]_i\|_2 &= \sqrt{\sum_{j=1}^r([\mathbf{X}\mathbf{Y}^\top]_{ij})^2}\\ &\le \sqrt{r(h(\mathbf{X})h(\mathbf{Y}))^2}\\ &= \sqrt{r}h(\mathbf{X})h(\mathbf{Y}),\\ h(\mathbf{X}\mathbf{Y}^\top) &= \max_{i=1}^r \left\|[\mathbf{X}\mathbf{Y}^\top]_i\right\|_2\\ &\le \sqrt{r}h(\mathbf{X})h(\mathbf{Y}).\end{aligned} \tag{39}$$

□

D.2 Gradient analysis

D.2.1 Preliminary

Given gradient $\nabla_{\mathbf{O}}\mathcal{L} \in \mathbb{R}^{n\times d}$, let's compute $\nabla_{\mathbf{S}}\mathcal{L}$ in every situation.

We first define:

$$\begin{aligned}c_1 &= h(\nabla_{\mathbf{O}}\mathcal{L})\\ &= \max_{i=1}^n \|\nabla_{\mathbf{O}_i}\mathcal{L}\|_2,\\ c_2 &= h(\mathbf{V})\\ &= \max_{i=1}^n \|\mathbf{V}_i\|_2 < \infty,\\ c_3 &= \min_{i,j} |s_{ij}| \ge 0.\end{aligned} \tag{40}$$

Before we get started, we have the following propositions. The proof can be found in Appendix D.3.

Proposition D.5. $c_1 < \infty$.

Proposition D.6. $\forall \mathbf{X} \in \mathbb{R}^{n\times m}$, *we have:*

$$\|\mathbf{X}\|_2 \le \sqrt{n}h(\mathbf{X}). \tag{41}$$

Take $\mathbf{X} = \mathbf{V}$, we get:

$$\|\mathbf{V}\|_2 \le \sqrt{n}h(\mathbf{V}) = \sqrt{n}c_2. \tag{42}$$

D.2.2 Vanilla/Linear attention

According to (30), we can discuss vanilla and linear attention under one formula:

$$\nabla_{\mathbf{P}}\mathcal{L} = [\nabla_{\mathbf{O}}\mathcal{L}]\mathbf{V}^{\mathsf{T}} \in \mathbb{R}^{n\times n}. \tag{43}$$

Then define matrix $\mathbf{U}^{(i)} \in \mathbb{R}^{n\times n}$:

$$[\mathbf{U}^{(i)}]_{jk} = \frac{\partial p_{ik}}{\partial s_{ij}}. \tag{44}$$

According to (9), in vanilla attention, we have:

$$\left|[\mathbf{U}^{(i)}]_{jk}\right| \le \frac{1}{4}, \tag{45}$$

while in linear attention, we have:

$$\left|[\mathbf{U}^{(i)}]_{jk}\right| \le \frac{1}{4|s_{ij}|} \le \frac{1}{4c_3}. \tag{46}$$

Since:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial s_{ij}} &= \sum_{k=1}^{n}\frac{\partial \mathcal{L}}{\partial p_{ik}}\frac{\partial p_{ik}}{\partial s_{ij}}\\ &= (\nabla_{\mathbf{P}_i}\mathcal{L})(\mathbf{U}_j^{(i)})^{\top}\\ &= (\nabla_{\mathbf{O}_i}\mathcal{L})\mathbf{V}^{\mathsf{T}}(\mathbf{U}_j^{(i)})^{\top}.\end{aligned} \tag{47}$$

So we have:

$$\begin{aligned}\left|\frac{\partial \mathcal{L}}{\partial s_{ij}}\right| &\le \|(\nabla_{\mathbf{O}_i}\mathcal{L})\mathbf{V}^{\mathsf{T}}\|_2 \left\|\mathbf{U}_j^{(i)\mathsf{T}}\right\|_2\\ &\le \|\nabla_{\mathbf{O}_i}\mathcal{L}\|_2\|\mathbf{V}^{\mathsf{T}}\|_2\|\mathbf{U}_j^{(i)}\|_2\\ &\le c_1\times\sqrt{n}c_2\times\frac{1}{4t}\\ &= \frac{\sqrt{n}c_1c_2}{4t},\end{aligned} \tag{48}$$

where $t = 1$ in vanilla attention and $t = c_3$ in linear attention.

On the other hand, according to Appendix C, in linear attention, there exist $\mathbf{q}_i, \mathbf{k}_j$, such that:

$$\begin{aligned}\frac{\partial p_{ik}}{\partial s_{ij}} &= \frac{1}{\|\mathbf{x}_0{}^{\mathsf{T}}\mathbf{x}_0\|}t_{ijk},\\ t_{ijk} &= \begin{cases}\frac{1}{n}(1-\frac{1}{n}) & j=k\\ -\frac{1}{n^2} & j\ne k\end{cases}.\end{aligned} \tag{49}$$

Then

$$\begin{aligned}\left|\frac{\partial \mathcal{L}}{\partial s_{ij}}\right| &= \left|\sum_{k=1}^{n}\frac{\partial \mathcal{L}}{\partial p_{ik}}\frac{\partial p_{ik}}{\partial s_{ij}}\right|\\ &= \frac{1}{\|\mathbf{x}_0{}^{\mathsf{T}}\mathbf{x}_0\|}\left|\sum_{k=1}^{n}\frac{\partial \mathcal{L}}{\partial p_{ik}}t_{ijk}\right|\\ &\ge \frac{1}{\epsilon}\left|\sum_{k=1}^{n}\frac{\partial \mathcal{L}}{\partial p_{ik}}t_{ijk}\right|.\end{aligned} \tag{50}$$

Let $\epsilon \to 0^+$, then $\left|\frac{\partial \mathcal{L}}{\partial s_{ik}}\right| \to \infty$. This means that the gradient in linear attention is unbounded.

D.2.3 NormAttention

We first define the second-moment of i'th row of $\mathbf{T}$:

$$\sigma_i^2 = \frac{\sum_{j=1}^{d}t_{ij}^2}{d}. \tag{51}$$

Then $\frac{\partial o_{ij}}{\partial t_{ik}}$ is as follows:

$$\frac{\partial o_{ij}}{\partial t_{ik}} = \frac{1}{\sqrt{\sigma_i^2+\epsilon}}\left[1\{j=k\} - \frac{1}{d}\frac{t_{ij}t_{ik}}{\sigma_i^2+\epsilon}\right]. \tag{52}$$

Notice that we have the following upper bound:

$$\begin{aligned}&\left|\frac{\partial o_{ij}}{\partial t_{ik}}\right|\\ =&\frac{1}{\sqrt{\sigma_i^2+\epsilon}}\left[1\{j=k\} - \frac{1}{d}\frac{t_{ij}t_{ik}}{\frac{\sum_{s=1}^{d}t_{is}^2}{d}+\epsilon}\right]\\ =&\frac{1}{\sqrt{\sigma_i^2+\epsilon}}\left[1\{j=k\} + \frac{t_{ij}t_{ik}}{\sum_{s=1}^{d}t_{is}^2+d\epsilon}\right]\\ \le&\frac{1}{\sqrt{\sigma_i^2+\epsilon}}\left[1\{j=k\} + \frac{1}{2}\frac{t_{ij}^2+t_{ik}^2}{\sum_{s=1}^{d}t_{is}^2}\right]\\ \le&\frac{1}{\sqrt{\sigma_i^2+\epsilon}}\left[1+\frac{1}{2}\right]\\ \le&\frac{3}{2\sqrt{\sigma_i^2+\epsilon}}.\end{aligned} \tag{53}$$

Let's define matrix $\mathbf{R}^{(i)} \in \mathbb{R}^{d\times d}$ as follows:

$$[\mathbf{R}^{(i)}]_{jk} = \frac{\partial o_{ik}}{\partial t_{ij}}. \tag{54}$$

Since

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial t_{ij}} &= \sum_{k=1}^{n}\frac{\partial \mathcal{L}}{\partial o_{ik}}\frac{\partial o_{ik}}{\partial t_{ij}}\\ &= (\nabla_{\mathbf{O}_i}\mathcal{L})(\mathbf{R}_j^{(i)})^{\top}.\end{aligned} \tag{55}$$

Then we can get:

$$\nabla_{\mathbf{T}_i}\mathcal{L} = (\nabla_{\mathbf{O}_i}\mathcal{L})(\mathbf{R}^{(i)})^\mathsf{T} \in \mathbb{R}^{1\times d}. \tag{56}$$

According to (53), we have:

$$\begin{aligned}\|\mathbf{R}^{(i)}\|_2 &\le \|\mathbf{R}^{(i)}\|_F \\ &\le \sqrt{\sum_{j=1}^{d}\sum_{k=1}^{d}\left[\frac{\partial o_{ij}}{\partial t_{ik}}\right]^2} \\ &\le \frac{3d}{2\sqrt{\sigma_i^2+\epsilon}} \\ &\le \frac{3d}{2\sqrt{\epsilon}}.\end{aligned} \tag{57}$$

Finally, we get:

$$\begin{aligned}\nabla_{\mathbf{S}_i}\mathcal{L} &= (\nabla_{\mathbf{T}_i}\mathcal{L})\mathbf{V}^\mathsf{T} \\ &= (\nabla_{\mathbf{O}_i}\mathcal{L})(\mathbf{R}^{(i)})^\mathsf{T}\mathbf{V}^\mathsf{T} \in \mathbb{R}^{1\times n}, \\ \frac{\partial\mathcal{L}}{\partial s_{ij}} &= (\nabla_{\mathbf{O}_i}\mathcal{L})(\mathbf{R}^{(i)})^\mathsf{T}\mathbf{V}_j, \\ \left|\frac{\partial\mathcal{L}}{\partial s_{ij}}\right| &= \left|(\nabla_{\mathbf{O}_i}\mathcal{L})(\mathbf{R}^{(i)})^\mathsf{T}\mathbf{V}_j\right| \\ &\le \|\nabla_{\mathbf{O}_i}\mathcal{L}\|_2\|\mathbf{R}^{(i)}\mathbf{V}_j\|_2 \\ &\le \|\nabla_{\mathbf{O}_i}\mathcal{L}\|_2\|\mathbf{R}^{(i)}\|_2\|\mathbf{V}_j\|_2 \\ &\le c_1 \times \frac{3d}{2\sqrt{\epsilon}} \times c_2 \\ &= \frac{3c_1c_2d}{2\sqrt{\epsilon}}.\end{aligned} \tag{58}$$

Let's summarize the previous results.
In vanilla attention, we have:

$$\left|\frac{\partial\mathcal{L}}{\partial s_{ij}}\right| \le \frac{\sqrt{n}c_1c_2}{4} < \infty. \tag{59}$$

In linear attention, there exist $\mathbf{q}_i, \mathbf{k}_j$, such that:

$$\left|\frac{\partial\mathcal{L}}{\partial s_{ij}}\right| \to \infty. \tag{60}$$

In NORMATTENTION, we have:

$$\left|\frac{\partial\mathcal{L}}{\partial s_{ij}}\right| \le \frac{3c_1c_2d}{2\sqrt{\epsilon}} < \infty. \tag{61}$$

So $\frac{\partial\mathcal{L}}{\partial s_{ij}}$ is bounded in vanilla attention and NORMATTENTION, while it's unbounded in linear attention. This makes the training of linear transformer unstable.

D.3 Proof of the proposition

Proof of Proposition D.5. Let's consider a one layer Transformer for classification tasks. The input is $\mathbf{X} \in \mathbb{R}^{n\times d}$, the label is $\mathbf{Y} \in \mathbb{R}^{n\times m}$, where m is the number of categories and $\mathbf{Y}_i$ is one-hot vector,. f_1, f_2 are the activation functions, here we take $f_1 = f_2 = \mathrm{ReLU}$ as an example. The parameters of the model are:

$$\begin{gathered}\mathbf{W}_1 \in \mathbb{R}^{d\times d_1}, \mathbf{W}_2 \in \mathbb{R}^{d_1\times d}, \\ \mathbf{W}_3 \in \mathbb{R}^{d\times m}.\end{gathered} \tag{62}$$

The forward pass of the model is[8]:

- $\mathbf{X}_1 = \mathrm{XAttention}(\mathbf{X}) \in \mathbb{R}^{n\times d}$.
- $\mathbf{X}_2 = f_1(\mathbf{X}_1\mathbf{W}_1) \in \mathbb{R}^{n\times d_1}$.
- $\mathbf{X}_3 = f_2(\mathbf{X}_2\mathbf{W}_2) \in \mathbb{R}^{n\times d}$.
- $\mathbf{O} = \mathbf{T}\mathbf{W}_3 \in \mathbb{R}^{n\times m}$.
- $\mathbf{P} = \mathrm{Softmax}(\mathbf{O}) \in \mathbb{R}^{n\times m}$.
- $\mathcal{L} = \mathrm{corss\_entropy}(\mathbf{P}, \mathbf{Y}) \in \mathbb{R}$.

The backward pass of the model is:

1. $\nabla_{\mathbf{O}}\mathcal{L} = \mathbf{P} - \mathbf{Y} \in \mathbb{R}^{n\times m}$.
   (a) The upper bound is:

$$\begin{aligned}&h\left(\nabla_{\mathbf{O}}\mathcal{L}\right) \\ =&\max\{\sum_{i=1}^{m} p_i^2 - 2p_1 + 1, \\ &p_i \ge 0, \sum_{i=1}^{m} p_i = 1\} \\ \triangleq& a_0 \\ <&\infty.\end{aligned} \tag{63}$$

2. $\nabla_{\mathbf{X}_3}\mathcal{L} = (\nabla_{\mathbf{O}}\mathcal{L})W_3^\top \in \mathbb{R}^{n\times d}$.
   (a) The upper bound is:

$$\begin{aligned}&h\left(\nabla_{\mathbf{X}_3}\mathcal{L}\right) \\ \le&\sqrt{d}h\left(\nabla_{\mathbf{O}}\mathcal{L}\right)h\left(\mathbf{W}_3\right) \\ \le&\sqrt{d}a_0h\left(\mathbf{W}_3\right) \\ \triangleq& a_1 < \infty.\end{aligned} \tag{64}$$

[8] XAttention stands for vanilla/norm attention.

3. $\nabla_{\mathbf{X}_2}\mathcal{L} = \left(f_2'(\mathbf{X}_2\mathbf{W}_2) \odot \nabla_{\mathbf{X}_3}\mathcal{L}\right)\mathbf{W}_2^\top \in \mathbb{R}^{n\times d}$.

   (a) The upper bound is:

$$\begin{aligned} & h\left(\nabla_{\mathbf{X}_2}\mathcal{L}\right) \\ \leq & \sqrt{d_1} h\left(f_2'(\mathbf{X}_2\mathbf{W}_2) \odot \nabla_{\mathbf{X}_3}\mathcal{L}\right) h\left(\mathbf{W}_2\right) \\ \leq & \sqrt{d_1} a_1 h(\mathbf{W}_2) \\ \triangleq & a_2 \\ < & \infty. \end{aligned} \tag{65}$$

4. $\nabla_{\mathbf{X}_1}\mathcal{L} = \left(f_1'(\mathbf{X}_1\mathbf{W}_1) \odot \nabla_{\mathbf{X}_2}\mathcal{L}\right)\mathbf{W}_1^\top \in \mathbb{R}^{n\times d_1}$.

   (a) The upper bound is:

$$\begin{aligned} & h\left(\nabla_{\mathbf{X}_1}\mathcal{L}\right) \\ \leq & \sqrt{d} h\left(f_1'(\mathbf{X}_1\mathbf{W}_1) \odot \nabla_{\mathbf{X}_2}\mathcal{L}\right) h\left(\mathbf{W}_1\right) \\ \leq & \sqrt{d} a_2 h(\mathbf{W}_2) \\ \triangleq & a_3 \\ < & \infty. \end{aligned} \tag{66}$$

So the gradient passed to XAttention module is bounded, *i.e.,* $c_1 = a_3 < \infty$. □

Proof of Proposition D.6.

$$\begin{aligned} \|\mathbf{X}\|_2 &\leq \|\mathbf{X}\|_F \\ &= \sqrt{\sum_{i=1}^{n} \|\mathbf{X}_i\|_2^2} \\ &\leq \sqrt{\sum_{i=1}^{n} [h(\mathbf{X})]^2} \\ &= \sqrt{n} h(\mathbf{X}). \end{aligned} \tag{67}$$

□

E Experiment configs

In this section, we will introduce detailed training hyperparameters. We introduce the configurations for autoregressive/bidirectional language model in table F. For LRA benchmark, we use the same configuration as Skyformer, which use 2-layer transformer model with 64 hidden dimensions, 2 attention heads, 85 GLU dimensions, Swish as GLU activation function. For batch size and learning rate , we use 16,1e-4 for Text Classification, 32,1e-4 for ListOps, 16,2e-4 for Document Retrieval, 128,2e-4 for Pathfinder, 256,1e-4 for Image Classification, the same as Skyformer.

F Pseudocode for visualization.

In this section, we provide pseudo codes for the 4th column of Figure 2 in Python:

```
import torch

def get_curve(w):
    n, m = w.shape
    num = 100
    P =  torch.linspace(0, 1, num)
    cnts = torch.zeros(num)
    for i in range(n):
        cnt = torch.zeros(num)
        w1 = w[i].clone()
        center = i % m
        s = w1[center].item()
        L = 1
        l = center - 1
        r = center + 1
        j = 1
        l_thre = 0
        r_thre = m
        flag = 0

        while L < m and j < num:
            if (s >= P[j].item()):
                cnt[j] = L
                j += 1
                continue
            if flag == 1:
                if r != r_thre:
                    s += w1[r].item()
                r = min(r_thre, r + 1)
                flag = 0
            else:
                if l != l_thre:
                    s += w1[l].item()
                l = max(l_thre, l - 1)
                flag = 1
            L = min(r - l + 1, m)

        if L >= m:
            for u in range(j, num):
                cnt[u] = min(L, m)

        cnt[0] = 0
        cnts += cnt
    cnts = cnts / n / m

    plt.plot(cnts, P)

    return cnts
```

Table 13: Detailed configurations used in our experiments. "Total batch size" means $\mathrm{batch\_per\_gpu} \times \mathrm{update\_freq} \times \mathrm{num\_gpus}$. "Attention dropout" is only used for vanilla attention. "ALM": autoregressive Language Model. "BLM": bidirectional Language Model.

	AML	BLM
Data	WikiText-103	WikiText-103
Tokenizer method	BPE	BPE
Src Vocab size	267744	50265
Encoder layers	0	12
Decoder layers	6	0
Hidden dimensions	512	768
Number of heads	8	12
GLU dimensions	2048	1365
GLU activation function	Swish	Swish
Sequence length	512	512
Total batch size	128	512
Number of updates	100k	50k
Warmup steps	8k	3k
Peak learning rate	5e-4	5e-4
Learning rate scheduler	Inverse sqrt	Polynomial decay
Optimizer	Adam	Adam
Adam ϵ	1e-8	1e-6
Adam (β_1, β_2)	(0.9, 0.98)	(0.9, 0.98)
Weight decay	0.01	0.01
Gradient clipping	0.0	0
Hidden dropout	0.1	0.1
Attention dropout	0	0.1