Linformer: Self-Attention with Linear Complexity

Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, Hao Ma
Facebook AI, Seattle, WA
{sinongwang, belindali, hanfang, mkhabsa, haom}@fb.com

Abstract

Large transformer models have shown extraordinary success in achieving state-of-the-art results in many natural language processing applications. However, training and deploying these models can be prohibitively costly for long sequences, as the standard self-attention mechanism of the Transformer uses $O(n^2)$ time and space with respect to sequence length. In this paper, we demonstrate that the self-attention mechanism can be approximated by a low-rank matrix. We further exploit this finding to propose a new self-attention mechanism, which reduces the overall self-attention complexity from $O(n^2)$ to $O(n)$ in both time and space. The resulting linear transformer, the *Linformer*, performs on par with standard Transformer models, while being much more memory- and time-efficient.

1 Introduction

Transformer models (Vaswani et al., 2017) have become ubiquitous for wide variety of problems in natural language processing (NLP), including translation (Ott et al., 2018), text classification, question answering, among others (Raffel et al., 2019; Mohamed et al., 2019). Over the last couple of years, the number of parameters in state-of-the-art NLP transformers has grown drastically, from the original 340 million introduced in BERT-Large to 175 billion in GPT-3 (Brown et al., 2020). Although these large-scale models yield impressive results on wide variety of tasks, training and deploying such model are slow in practice. For example, the original BERT-Large model (Devlin et al., 2019) takes four days to train on 16 Cloud TPUs, and the recent GPT-3 (Brown et al., 2020) consumed orders of magnitude more petaflops / day to train compared to its predecessor, GPT-2 (Radford et al., 2019). Beyond training, deploying Transformer models to real world applications is also expensive, usually requiring extensive distillation (Hinton et al., 2015) or compression.

Knowledge distilation means to use a bigger model and use the parameters of the bigger model to train a smaller model.

The main efficiency bottleneck in Transformer models is its self-attention mechanism. Here, each token's representation is updated by attending to *all* other tokens in the previous layer. This operation is key for retaining long-term information, giving Transformers the edge over recurrent models on long sequences. However, attending to all tokens at each layer incurs a complexity of $O(n^2)$ with respect to sequence length. Thus, in this paper, we seek to answer the question: *can Transformer models be optimized to avoid this quadratic operation, or is this operation required to maintain strong performance?*

Prior work has proposed several techniques for improving the efficiency of self-attention. One popular technique is introducing sparsity into attention layers (Child et al., 2019; Qiu et al., 2019; Beltagy et al., 2020) by having each token attend to only a subset of tokens in the whole sequence. This reduces the overall complexity of the attention mechanism to $O(n\sqrt{n})$ (Child et al., 2019). However, as shown in Qiu et al. (2019), this approach suffers from a large performance drop with limited efficiency gains, i.e., a 2% drop with only 20% speed up. More recently, the Reformer (Kitaev et al., 2020) used locally-sensitive hashing (LSH) to reduce the self-attention complexity to $O(n \log(n))$. However, in practice, the Reformer's efficiency gains only appear on sequences with length > 2048 (Figure 5 in Kitaev et al. (2020)). Furthermore, the Reformer's multi-round hashing approach actually *increases* the number of sequential operations, which further undermines their final efficiency gains.

Preprint. Under review.

In this work, we introduce a novel approach for tackling the self-attention bottleneck in Transformers. Our approach is inspired by the key observation that *self-attention is low rank*. More precisely, we show both theoretically and empirically that the stochastic matrix formed by self-attention can be approximated by a low-rank matrix. Empowered by this observation, we introduce a novel mechanism that reduces self-attention to an $O(n)$ operation in both space- and time-complexity: we decompose the original scaled dot-product attention into multiple smaller attentions through linear projections, such that the combination of these operations forms a low-rank factorization of the original attention. A summary of runtimes for various Transformer architectures, including ours, can be found in Table 1.

One predominant application of Transformers, that has seen the most gains, is using them as pretrained language models, whereby models are first pretrained with a language modeling objective on a large corpus, then finetuned on target tasks using supervised data (Devlin et al., 2019; Liu et al., 2019; Lewis et al., 2019). Following Devlin et al. (2019), we pretrain our model on BookCorpus (Zhu et al., 2015) plus English Wikipedia using masked-language-modeling objective. We observe similar pretraining performance to the standard Transformer model. We then finetune our pretrained models on three tasks from GLUE (Wang et al., 2018) and one sentiment analysis task, IMDB reviews (Maas et al., 2011). On these tasks, we find that our model performs comparably, or even slightly better, than the standard pretrained Transformer, while observing significant training and inference speedups.

Model Architecture	Complexity per Layer	Sequential Operation
Recurrent	$O(n)$	$O(n)$
Transformer, (Vaswani et al., 2017)	$O(n^2)$	$O(1)$
Sparse Tansformer, (Child et al., 2019)	$O(n\sqrt{n})$	$O(1)$
Reformer, (Kitaev et al., 2020)	$O(n \log(n))$	$O(\log(n))$
Linformer	$O(n)$	$O(1)$

This paper.

Table 1: Per-layer time complexity and minimum number of sequential operations as a function of sequence length (n) for various architectures.

2 Backgrounds and Related works

2.1 Transformer and Self-Attention

The Transformer is built upon the idea of Multi-Head Self-Attention (MHA), which allows the model to jointly attend to information at different positions from different representation subspaces. MHA is defined as

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \text{head}_2, \dots, \text{head}_h)W^O, \tag{1}$$

where $Q, K, V \in \mathbb{R}^{n \times d_m}$ are input embedding matrices, n is sequence length, d_m is the embedding dimension, and h is the number of heads. Each head is defined as:

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) = \underbrace{\text{softmax}\left[\frac{QW_i^Q(KW_i^K)^T}{\sqrt{d_k}}\right]}_{P} VW_i^V, \tag{2}$$

where $W_i^Q, W_i^K \in \mathbb{R}^{d_m \times d_k}, W_i^V \in \mathbb{R}^{d_m \times d_v}, W^O \in \mathbb{R}^{hd_v \times d_m}$ are learned matrices and d_k, d_v are the hidden dimensions of the projection subspaces. For the rest of this paper, we will not differentiate between d_k and d_v and just use d.

The self-attention defined in (2) refers to a context mapping matrix $P \in \mathbb{R}^{n \times n}$. The Transformer uses P to capture the input context for a given token, based on a combination of all tokens in the sequence. However, computing P is expensive. It requires multiplying two $n \times d$ matrices, which is $O(n^2)$ in time and space complexity. This quadratic dependency on the sequence length has become a bottleneck for Transformers.

2.2 Related works

There has been much prior literature on improving the efficiency of Transformers, especially the self-attention bottleneck. The most common techniques for model efficiency that can be applied to Transformers (some specific to Transformers, others more general-purpose) include:

Mixed Precision (Micikevicius et al., 2017): Using half-precision or mixed-precision representations of floating points is popular in deep learning, and is also widely used in training Transformers (Ott et al., 2019). This technique can be further improved through Quantization Aware Training (Jacob et al., 2018; Fan et al., 2020), where the weights are quantized during training and the gradients are approximated with the Straight-Through Estimator. This line of work is orthogonal to our approach, and we use mixed-precision training by default.

Knowledge Distillation (Hinton et al., 2015): Knowledge distillation aims to transfer the "knowledge" from a large teacher model to a lightweight student model. The student model is then used during inference. However this approach has drawbacks: It does not address speeding up the *teacher* model during training, and moreover, student models usually suffer performance degradation compared to the teacher model. For example, when distilling a 12-layer BERT to a 6-layer BERT, the student model experiences an average 2.5% performance drop on several benchmark tasks (Sanh et al., 2019).

Sparse Attention (Child et al., 2019): This technique improves the efficiency of self-attention by adding sparsity in the context mapping matrix P. For example, the Sparse Transformer (Child et al., 2019) only computes P_{ij} around the diagonal of matrix P (instead of the all P_{ij}). Meanwhile, blockwise self-attention (Qiu et al., 2019) divides P into multiple blocks and only computes P_{ij} within the selected blocks. However, these techniques also suffer a large performance degradation, while having only limited additional speed-up, i.e., 2% drop with 20% speed up.

LSH Attention (Kitaev et al., 2020): Locally-sensitive hashing (LSH) attention utilizes a multi-round hashing scheme when computing dot-product attention, which in theory reduces the self-attention complexity to $O(n \log(n))$. However, in practice, their complexity term has a large constant 128^2 and it is only more efficient than the vanilla transformer when sequence length is extremely long.

Improving Optimizer Efficiency: Microbatching (Huang et al., 2019) splits a batch into small microbatches (which can be fit into memory), and then separately runs forward and backward passes on them with gradient accumulation. Gradient checkpointing (Chen et al., 2016) saves memory by only caching activations of a subset of layers. The uncached activations are recomputed during backpropagation from the latest checkpoint. Both techniques trade off time for memory, and do not speed up inference.

As we've noted, most common techniques have limitations in reducing both the training and inference time/memory consumption, we investigate how to optimize the self-attention layers and introduce our approach next.

3 Self-Attention is Low Rank

In this section, we demonstrate that the self-attention mechanism, i.e., the context mapping matrix P, is low-rank.

We first provide a spectrum analysis of the context mapping matrix P. We use two pretrained transformer models, RoBERTa-base (12-layer stacked transformer) and RoBERTa-large (24-layer stacked transformer) (Liu et al., 2019) on two tasks: masked-language-modeling task on Wiki103 (Merity et al., 2016) and classification task on IMDB (Maas et al., 2011). In Figure 1 (left), we apply singular value decomposition into P across different layers and different heads of the model, and plot the normalized cumulative singular value averaged over 10k sentences. The results exhibit a clear long-tail spectrum distribution across each layer, head and task. This implies that most of the information of matrix P can be recovered from the first few largest singular values. In Figure 1 (right), we plot a heatmap of the normalized cumulative singular value at the 128-th largest singular value (out of 512). We observe that the spectrum distribution in higher layers is more skewed than in lower layers, meaning that, in higher layers, more information is concentrated in the largest singular values and the rank of P is lower.

This is the point this paper is trying to place, i.e, most of the information retrieved by the transformer can be accessed by the initial matrix rows only.

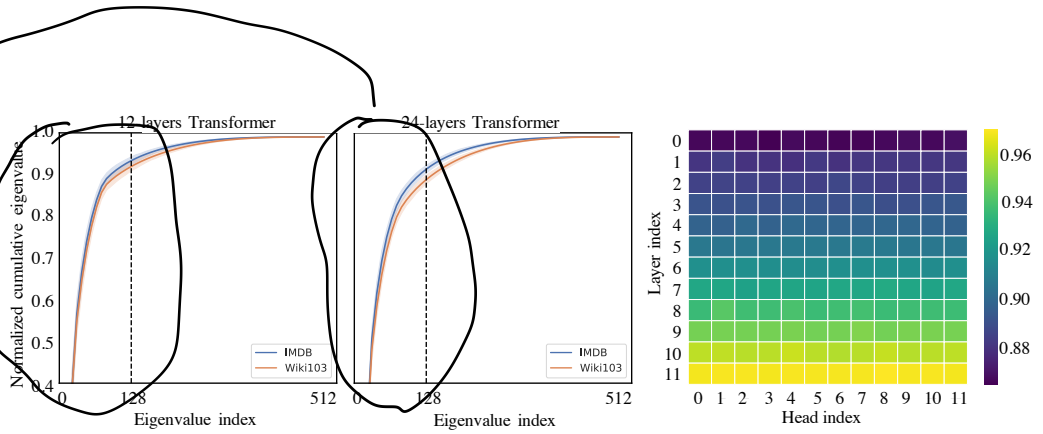
Below, we provide a theoretical analysis of the above spectrum results.

Theorem 1. (self-attention is low rank) *For any $Q, K, V \in \mathbb{R}^{n \times d}$ and $W_i^Q, W_i^K, W_i^V \in \mathbb{R}^{d \times d}$, for any column vector $w \in \mathbb{R}^n$ of matrix VW_i^V, there exists a low-rank matrix $\tilde{P} \in \mathbb{R}^{n \times n}$ such that*

$$\Pr(\|\tilde{P}w^T - Pw^T\| < \epsilon \|Pw^T\|) > 1 - o(1) \text{ and } \textit{rank}(\tilde{P}) = \Theta(\log(n)), \quad (3)$$

where the context mapping matrix P is defined in (2).

The theorem on which the ENTIRE paper is based.



As it can be seen, the graph flattens, which shows, the information gathered can be generalized by this point only.

Figure 1: Left two figures are spectrum analysis of the self-attention matrix in pretrained transformer model (Liu et al., 2019) with $n = 512$. The Y-axis is the normalized cumulative singular value of context mapping matrix P, and the X-axis the index of largest eigenvalue. The results are based on both RoBERTa-base and large model in two public datasets: Wiki103 and IMDB. The right figure plots the heatmap of normalized cumulative eigenvalue at the 128-th largest eigenvalue across different layers and heads in Wiki103 data.

Proof. Based on the definition of the context mapping matrix P, we can write

$$P = \text{softmax}\underbrace{\left[\frac{QW_i^Q(KW_i^K)^T}{\sqrt{d}}\right]}_{A} = \exp(A) \cdot D_A^{-1}, \tag{4}$$

where D_A is an $n \times n$ diagonal matrix. The main idea of this proof is based on the distributional Johnson–Lindenstrauss lemma (Lindenstrauss, 1984) (JL for short). We construct the approximate low rank matrix as $\tilde{P} = \exp(A) \cdot D_A^{-1} R^T R$, where $R \in \mathbb{R}^{k \times n}$ with i.i.d. entries from $N(0, 1/k)$. We can then use the JL lemma to show that, for any column vector $w \in \mathbb{R}^n$ of matrix VW_i^V, when $k = 5\log(n)/(\epsilon^2 - \epsilon^3)$, we have

$$\Pr\left(\|PR^T Rw^T - Pw^T\| \leq \epsilon \|Pw^T\|\right) > 1 - o(1). \tag{5}$$

A simple error coefficient.

For more details, refer to the supplementary materials. □

Given the low-rank property of the context mapping matrix P, one straightforward idea is to use singular value decomposition (SVD) to approximate P with a low-rank matrix P_{low}, as follows

$$P \approx P_{\text{low}} = \sum_{i=1}^{k} \sigma_i u_i v_i^T = \underbrace{\left[u_1, \cdots, u_k\right]}_{k} \text{diag}\{\sigma_1, \cdots, \sigma_k\} \left.\begin{bmatrix} v_1 \\ \vdots \\ v_k \end{bmatrix}\right\} k \tag{6}$$

where σ_i, u_i and v_i are the i largest singular values and their corresponding singular vectors. Based on the results in Theorem 1 and the Eckart–Young–Mirsky Theorem (Eckart & Young, 1936), one can use P_{low} to approximate self-attention (2) with ϵ error and $O(nk)$ time and space complexity. However, this approach requires performing an SVD decomposition in *each* self-attention matrix, which adds additional complexity. Therefore, we propose another approach for low-rank approximation that avoids this added complexity.

4 Model

In this section, we propose a new self-attention mechanism which allows us to compute the contextual mapping $P \cdot VW_i^V$ in linear time and memory complexity with respect to sequence length.

The main idea of our proposed linear self-attention (Figure 2) is to add two linear projection matrices $E_i, F_i \in \mathbb{R}^{n \times k}$ when computing key and value. We first project the original $(n \times d)$-dimensional

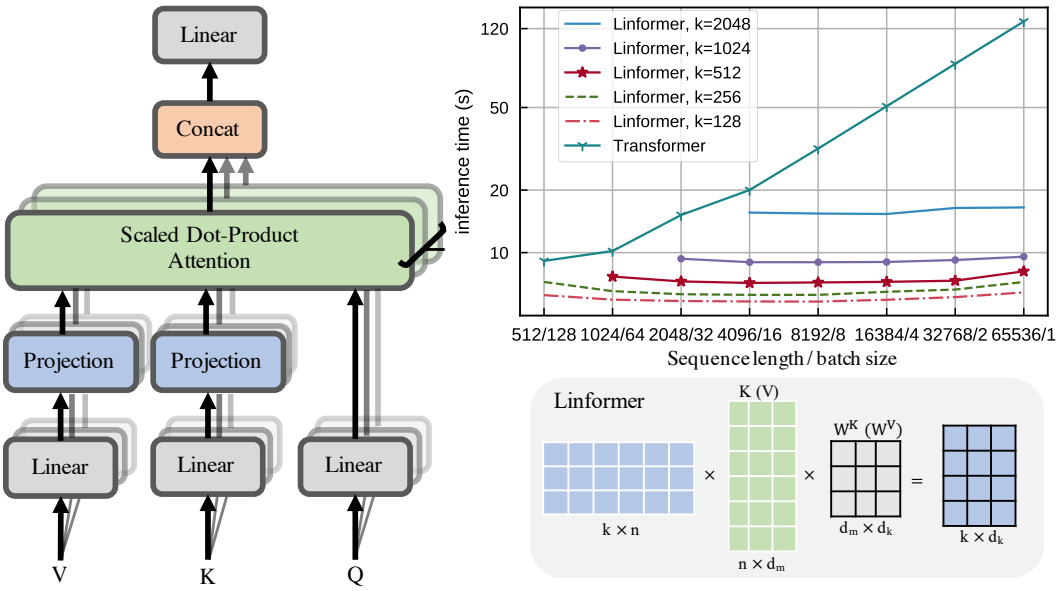


Figure 2: Left and bottom-right show architecture and example of our proposed multihead linear self-attention. Top right shows inference time vs. sequence length for various Linformer models.

key and value layers KW_i^K and VW_i^V into $(k \times d)$-dimensional projected key and value layers. We then compute an $(n \times k)$-dimensional context mapping matrix $\bar{P}$ using scaled dot-product attention.

$$\begin{aligned}\overline{\text{head}_i} &= \text{Attention}(QW_i^Q, E_iKW_i^K, F_iVW_i^V) \\ &= \underbrace{\text{softmax}\left(\frac{QW_i^Q(E_iKW_i^K)^T}{\sqrt{d_k}}\right)}_{\bar{P}:n\times k} \cdot \underbrace{F_iVW_i^V}_{k\times d},\end{aligned} \tag{7}$$

Finally, we compute context embeddings for each head_i using $\bar{P} \cdot (F_iVW_i^V)$. Note the above operations only require $O(nk)$ time and space complexity. Thus, if we can choose a very small projected dimension k, such that $k \ll n$, then we can significantly reduce the memory and space consumption. The following theorem states that, when $k = O(d/\epsilon^2)$ (independent of n), one can approximate $P \cdot VW_i^V$ using linear self-attention (7) with ϵ error.

Theorem 2. *(Linear self-attention) For any $Q_i, K_i, V_i \in \mathbb{R}^{n\times d}$ and $W_i^Q, W_i^K, W_i^V \in \mathbb{R}^{d\times d}$, if $k = \min\{\Theta(9d\log(d)/\epsilon^2), 5\Theta(\log(n)/\epsilon^2)\}$, then there exists matrices $E_i, F_i \in \mathbb{R}^{n\times k}$ such that, for any row vector w of matrix $QW_i^Q(KW_i^K)^T/\sqrt{d}$, we have*

$$\Pr\left(\|\text{softmax}(wE_i^T)F_iVW_i^V - \text{softmax}(w)VW_i^V\| \le \epsilon\|\text{softmax}(w)\|\|VW_i^V\|\right) > 1 - o(1) \tag{8}$$

As described in the first theorem.

Proof. The main idea of proof is based on the distributional Johnson–Lindenstrauss lemma (Lindenstrauss, 1984). We first prove that for any row vector $x \in \mathbb{R}^n$ of matrix $QW_i^Q(KW_i^K)^T/\sqrt{d_k}$ and column vector $y \in \mathbb{R}^n$ of matrix VW_i^V,

$$\Pr\left(\|\exp(xE_i^T)F_iy^T - \exp(x)y^T\| \le \epsilon\|\exp(x)y^T\|\right) > 1 - 2e^{-(\epsilon^2-\epsilon^3)k/4}, \tag{9}$$

where $E_i = \delta R$ and $F_i = e^{-\delta}R$, where $R \in \mathbb{R}^{k\times n}$ with i.i.d. entries from $N(0, 1/k)$ and δ is a small constant. Applying the result in (9) to every row vector of matrix A and every column vector of matrix V, one can directly prove that, for any row vector A_i of matrix A,

$$\Pr\left(\|\exp(A_iE_i^T)F_iV - \exp(A_i)V\| \le \epsilon\|\exp(A_i)V\|\right) > 1 - o(1), \tag{10}$$

by setting $k = 5\log(nd)/(\epsilon^2 - \epsilon^3)$. This result does not utilize the low rank property of matrix A (rank(A)=d) and the resultant k has a dependency on sequence length n. We will further utlize the fact that rank(A)=d to prove the choice of k can be constant and independent of sequence length n. For more details, refer to the supplementary materials. □

In Figure 2 (top right), we plot the inference speed of Linformer and standard Transformer versus sequence length, while holding the total number of tokens fixed. We see that while standard Transformer becomes slower at longer sequence lengths, the Linformer speed remains relatively flat and is significantly faster at long sequences.

Additional Efficiency Techniques Several additional techniques can be introduced on top of Linformer to further optimize for both performance and efficiency:

Parameter sharing between projections: One can share parameters for the linear projection matrices E_i, F_i across layers and heads. In particular, we experimented with 3 levels of sharing:

- Headwise sharing: for each layer, we share two projection matrices E and F such that $E_i = E$ and $F_i = F$ across all heads i.
- Key-value sharing: we do headwise sharing, with the additional constraint of sharing the key and value projections. For each layer, we create a single projection matrix E such that $E_i = F_i = E$ for each key-value projection matrix across all head i.
- Layerwise sharing: we use a single projection matrix E across *all layers*, for all heads, and for both key and value.

For example, in a 12-layer, 12-head stacked Transformer model, headwise sharing, key-value sharing and layerwise sharing will introduce 24, 12, and 1 distinct linear projection matrices, respectively.

Nonuniform projected dimension: One can choose a different projected dimension k for different heads and layers. As shown in Figure 1 (right), the contextual mapping matrices in different heads and layers have distinct spectrum distributions, and heads in higher layer tend towards a more skewed distributed spectrum (lower rank). This implies one can choose a smaller projected dimension k for higher layers.

General projections: One can also choose different kinds of low-dimensional projection methods instead of a simple linear projection. For example, one can choose mean/max pooling, or convolution where the kernel and stride is set to n/k. The convolutional functions contain parameters that require training.

5 Experiments

In this section, we present experimental results for the the techniques described above. We analyze the techniques one-by-one and explore how they impact performance.

5.1 Pretraining Perplexities

We first compare the pretraining performance of our proposed architecture against RoBERTa (Liu et al., 2019), which is based on the Transformer. Following Devlin et al. (2019), we use BookCorpus (Zhu et al., 2015) plus English Wikipedia as our pretraining set (3300M words). All models are pretrained with the masked-language-modeling (MLM) objective, and the training for all experiments are parallelized across 64 Tesla V100 GPUs with 250k updates.

Effect of projected dimension: We experiment with various values for the projected dimension k. (We use the same k across all layers and heads of Linformer.) In the Figure 3(a) and (b), we plot the validation perplexity curves for both the standard Transformer and the Linformer across different k, for maximum sequence lengths $n = 512$ and $n = 1024$. As expected, the Linformer performs better as projected dimension k increases. However, even at $k = 128$ for $n = 512$ and $k = 256$ for $n = 1024$, Linformer's performance is already nearly on par with the original Transformer. **Effect of sharing projections:** In Figure 3(c), we plot the validation perplexity curves for the three parameter sharing strategies (headwise, key-value, and layerwise) with $n = 512$. Note that when we use just a single projection matrix (i.e. for layerwise sharing), the resulting Linformer model's validation perplexity almost matches that of the the non-shared model. This suggests that we can decrease the number of additional parameters in our model, and consequently, it's memory consumption, without much detriment to performance.

Effect of longer sequences: We evaluate the effect of sequence length during Linformer pretraining. In the Figure 3(d), we plot the validation perplexity for Linformer with $n \in \{512, 1024, 2048, 4096\}$,

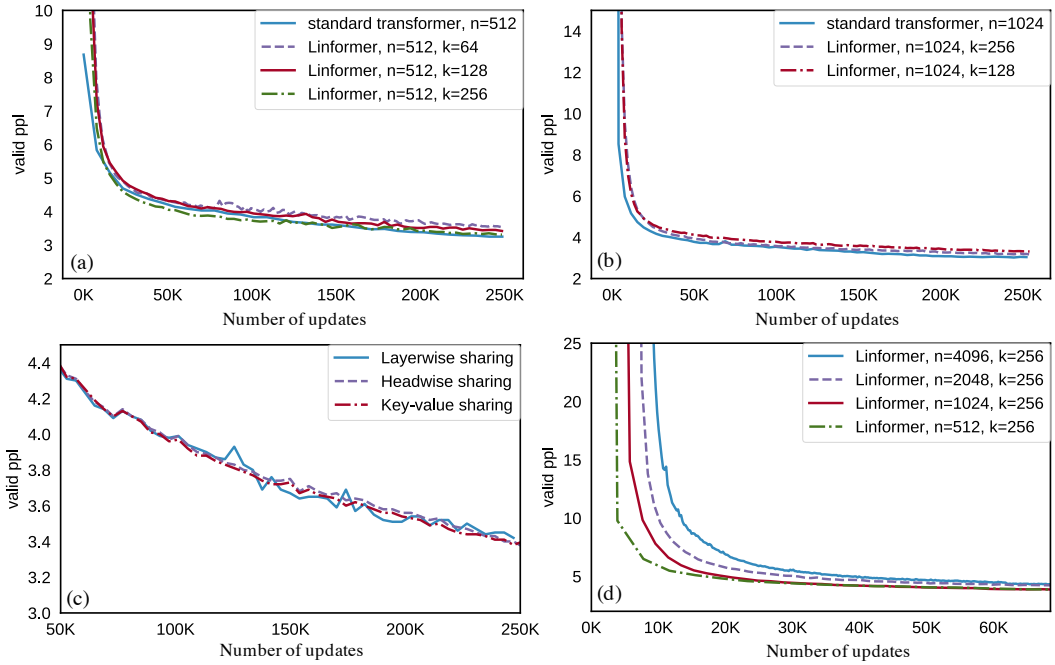


Figure 3: Pretraining validation perplexity versus number of updates.

holding projected dimension k fixed at 256. Note that as sequence length increases, even though our projected dimension is fixed, the final perplexities after convergence remain about the same. This further empirically supports our assertion that the Linformer is linear-time.

n	Model	SST-2	IMDB	QNLI	QQP	Average
512	Liu et al. (2019), RoBERTa-base	93.1	94.1	90.9	**90.9**	92.25
	Linformer, 128	92.4	94.0	90.4	90.2	91.75
	Linformer, 128, shared kv	**93.4**	93.4	90.3	90.3	91.85
	Linformer, 128, shared kv, layer	93.2	93.8	90.1	90.2	91.83
	Linformer, 256	93.2	94.0	90.6	90.5	92.08
	Linformer, 256, shared kv	93.3	93.6	90.6	90.6	92.03
	Linformer, 256, shared kv, layer	93.1	94.1	**91.2**	90.8	**92.30**
512	Devlin et al. (2019), BERT-base	92.7	93.5	91.8	89.6	91.90
	Sanh et al. (2019), Distilled BERT	91.3	92.8	89.2	88.5	90.45
1024	Linformer, 256	93.0	93.8	90.4	90.4	91.90
	Linformer, 256, shared kv	93.0	93.6	90.3	90.4	91.83
	Linformer, 256, shared kv, layer	93.2	**94.2**	90.8	90.5	92.18

Table 2: Dev set results on benchmark natural language understanding tasks. The RoBERTa-base model here is pretrained with same corpus as BERT.

5.2 Downstream Results

Thus far, we have only examined the pretraining perplexities of our model. However, we wish to show that our conclusions hold after *finetuning* on downstream tasks. We finetune our Linformer on IMDB (Maas et al., 2011) and SST-2 (Socher et al., 2013) (sentiment classification), as well as QNLI (natural language inference) (Rajpurkar et al., 2016), and QQP (textual similarity) (Chen et al., 2018) We do the same with RoBERTa, 12-layer BERT-base and 6-layer distilled BERT. All of our models, including the Transformer baselines, were pretrained with the same objective, pretraining corpus, and up to 250k updates (although our Linformer takes much less wall-clock time to get to 250k updates, and was consequently trained for less time). Results are listed in Table 2.

We observe that the Linformer model ($n = 512, k = 128$) has comparable downstream performance to the RoBERTa model, and in fact even slightly outperforms it at $k = 256$. Moreover, we note that although the Linformer's layerwise sharing strategy shares a single projection matrix across the entire model, it actually exhibits the best accuracy result of all three parameter sharing strategies. Furthermore, the Linformer pretrained with longer sequence length ($n = 1024, k = 256$) has similar results to the one pretrained with shorter length ($n = 512, k = 256$), this empirically supports the notion that *the performance of Linformer model is mainly determined by the projected dimension k instead of the ratio n/k.*

5.3 Inference-time Efficiency Results

In Table 3, we report the inference efficiencies of Linformer (with layerwise sharing) against a standard Transformer. We benchmark both models' inference speed and memory on a 16GB Tesla V100 GPU card. We randomly generate data up to some sequence length n and perform a full forward pass on a multiple batches. We also choose batch size based on the maximum batch size that can fit in memory, and our memory savings are computed based on this number.

length n	projected dimensions k				
	128	256	512	1024	2048
512	1.5x	1.3x	-	-	-
1024	1.7x	1.6x	1.3x	-	-
2048	2.6x	2.4x	2.1x	1.3x	-
4096	3.4x	3.2x	2.8x	2.2x	1.3x
8192	5.5x	5.0x	4.4x	3.5x	2.1x
16384	8.6x	7.8x	7.0x	5.6x	3.3x
32768	13x	12x	11x	8.8x	5.0x
65536	20x	18x	16x	14x	7.9x

length n	projected dimensions k				
	128	256	512	1024	2048
512	1.7x	1.5x	-	-	-
1024	3.0x	2.9x	1.8x	-	-
2048	6.1x	5.6x	3.6x	2.0x	-
4096	14x	13x	8.3x	4.3x	2.3x
8192	28x	26x	17x	8.5x	4.5x
16384	56x	48x	32x	16x	8x
32768	56x	48x	36x	18x	16x
65536	60x	52x	40x	20x	18x

Table 3: Inference-time efficiency improvements of the Linformer over the Transformer, across various projected dimensions k and sequence lengths n. Left table shows time saved. Right table shows memory saved.

From Table 3, we see that even with $n = 512$ and $k = 128$, Linformer has $1.5\times$ faster inference time and allows for a $1.7\times$ larger maximum batch size than the Transformer. As sequence length increases, the inference-time speed-up and memory savings are even more dramatic. We also plot inference times of both Linformer and Transformer on the 100 data samples in the top right of Figure 2.

6 Conclusion

Transformer models are notoriously slow to train and deploy in practice since their self-attention operations have $O(n^2)$ time and space complexity with respect to sequence length n. In this paper, we demonstrate, both theoretically and empirically, that the stochastic matrix formed by self-attention mechanism is low-rank. We further leverage this observation to propose a new, highly efficient self-attention mechanism. Through a combination of theoretical and empirical analysis, we demonstrate that our proposed approach is $O(n)$ with respect to sequence length.

Broader Impact

Our work focuses on making Transformers more efficient by introducing a mechanism that reduces self-attention to linear-time complexity. Potential positive impacts of efficient transformers include increasing the accessibility of our models, both for deployment on devices, as well as during training for research purposes. It also has potential impact on training transformer on images since we can support very long sequences. Furthermore, there are positive environmental benefits associated with decreasing the power consumption of models. As such, we see no immediate negative ethical or societal impacts of our work beyond what applies to other core building blocks of deep learning.

References

Rosa I Arriaga and Santosh Vempala. An algorithmic theory of learning: Robust concepts and random projection. *Machine Learning*, 63(2):161–182, 2006.

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

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*, 2016.

Zihan Chen, Hongbo Zhang, Xiaoji Zhang, and Leqi Zhao. Quora question pairs, 2018.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, 2019.

Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

Angela Fan, Pierre Stock, Benjamin Graham, Edouard Grave, Remi Gribonval, Herve Jegou, and Armand Joulin. Training with quantization noise for extreme fixed-point compression. *arXiv preprint arXiv:2004.07320*, 2020.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V Le, Yonghui Wu, et al. Gpipe: Efficient training of giant neural networks using pipeline parallelism. In *Advances in Neural Information Processing Systems*, pp. 103–112, 2019.

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2704–2713, 2018.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *International Conference on Learning Representations*, 2020.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *ACL*, 2019.

W Johnson J Lindenstrauss. Extensions of lipschitz maps into a hilbert space. *Contemp. Math*, 26: 189–206, 1984.

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

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*, pp. 142–150. Association for Computational Linguistics, 2011.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed precision training. *arXiv preprint arXiv:1710.03740*, 2017.

Abdelrahman Mohamed, Dmytro Okhonko, and Luke Zettlemoyer. Transformers with convolutional context for asr. *arXiv preprint arXiv:1904.11660*, 2019.

Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. Scaling neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pp. 1–9, 2018.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pp. 48–53, 2019.

Jiezhong Qiu, Hao Ma, Omer Levy, Scott Wen-tau Yih, Sinong Wang, and Jie Tang. Blockwise self-attention for long document understanding. *arXiv preprint arXiv:1911.02972*, 2019.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9, 2019.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

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

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1631–1642, 2013.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. *CoRR*, abs/1804.07461, 2018. URL `http://arxiv.org/abs/1804.07461`.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pp. 19–27, 2015.

A Proof of Theorem 1

Proof. The main proof idea is based on the distributional Johnson–Lindenstrauss lemma (Lindenstrauss, 1984) (JL, for short), the following version is from (Arriaga & Vempala, 2006).

Lemma 1. *Let R be an $k \times n$ matrix, $1 \leq k \leq n$, with i.i.d. entries from $N(0, 1/k)$. For any $x, y \in \mathbb{R}^n$, we have*

$$\Pr\left(\|Rx\| \leq (1+\epsilon)\|x\|\right) > 1 - e^{-(\epsilon^2-\epsilon^3)k/4}, \tag{11}$$

$$\Pr\left(\|xR^TRy^T - xy^T\| \leq \epsilon\|xy\|\right) > 1 - 2e^{-(\epsilon^2-\epsilon^3)k/4}. \tag{12}$$

For simplicity, we will omit the subscript i for matrix W_i^K, W_i^Q, W_i^V, E_i and F_i. We will regard Q as QW^Q, K as KW^K and V as VW^V. Define

$$A = \frac{QW_i^Q(KW_i^K)^T}{\sqrt{d}} \tag{13}$$

Based on the definition of contextual mapping matrix P, we have

$$\begin{aligned} P &= \text{softmax}\left[\frac{QW_i^Q(KW_i^K)^T}{\sqrt{d}}\right] \\ &= \exp\left(A\right) \cdot D_A^{-1}, \end{aligned} \tag{14}$$

where D_A is an $n \times n$ diagonal matrix such that

$$(D_A)_{ii} = \sum_{j=1}^{n} \exp\left(A_{ji}\right) \tag{15}$$

Here we provide a constructive proof. Given any approximation error $\epsilon > 0$, define the following matrix.

$$\tilde{P} = \exp\left(A\right) \cdot D_A^{-1}R^TR, \tag{16}$$

where R be an $k \times n$ matrix, $1 \leq k \leq n$, with i.i.d. entries from $N(0, 1/k)$. Clearly the rank of matrix $\tilde{P}$ satisifies

$$\text{rank}(\tilde{P}) \leq \text{rank}(R) = k. \tag{17}$$

We further show that, when $k = \log(n)$, we have that, for any column vector $w \in \mathbb{R}^n$,

$$\Pr\left(\|\tilde{P}h - Ph\| \leq \epsilon\|Ph\|\right) > 1 - o(1). \tag{18}$$

This concludes the theorem. For any row vector $u \in \mathbb{R}^n$ of matrix P and any column vector $w \in \mathbb{R}^n$ of matrix VW^V, applying the JL Lemma, we can obtain

$$\Pr\left(\|uR^tRw^T - uw^T\| \leq \epsilon\|uw^T\|\right) > 1 - 2e^{-(\epsilon^2-\epsilon^3)k/4}. \tag{19}$$

Therefore, we have

$$\begin{aligned} \Pr\left(\|\tilde{P}w^T - Pw^T\| \leq \epsilon\|Pw^T\|\right) &= \Pr\left(\|PR^TRw^T - Pw^T\| \leq \epsilon\|Pw^T\|\right) \\ &\overset{(a)}{\geq} 1 - \sum_{x \in P} \Pr\left(\|xR^TRw^T - xw^T\| > \epsilon\|xw^T\|\right) \\ &\overset{(b)}{>} 1 - 2ne^{-(\epsilon^2-\epsilon^3)k/4}. \end{aligned} \tag{20}$$

The above, step (a) is based on the union bound. The step (b) is utilizing the result of JL Lemma. Let $k = 5\log(n)/(\epsilon^2 - \epsilon^3)$, then theorem follows. □

B Proof of Theorem 2

Proof. Define $E = \delta R$ and $F = e^{-\delta} R$, where $R \in \mathbb{R}^{n\times k}$ with i.i.d. entries from $N(0, 1/k)$, δ is a constant with $\delta = 1/2^n$. We will first prove that for any row vector $x \in \mathbb{R}^n$ of matrix QK^T and column vector $y \in \mathbb{R}^n$ of matrix V,

$$\Pr\left(\|\exp(xE^T)Fy^T - \exp(x)y^T\| \le \epsilon\|\exp(x)y^T\|\right) > 1 - 2e^{-(\epsilon^2-\epsilon^3)k/4}. \tag{21}$$

Based on the triangle inequality, we have

$$\begin{aligned}
\|\exp(xE^T)Fy\exp(x)y^T\| &\le \|\exp(xE^T)Fy - \exp(x)R^TRy\| + \|\exp(x)R^TRy - \exp(x)y^T\| \\
&\overset{(a)}{\le} (1+\epsilon)\|y\|\|\exp(xE^T) - \exp(x)R^T\| + \|\exp(x)R^TRy - \exp(x)y^T\| \\
&\overset{(b)}{\le} \|\exp(x)R^TRy - \exp(x)y^T\| + o(\|\exp(x)\|\|y\|) \\
&\overset{(c)}{\le} \epsilon\|\exp(x)\|\|y\| + o(\|\exp(x)\|\|y\|)
\end{aligned} \tag{22}$$

The above, step (a) is based on the Cauchy inequality and JL Lemma in (11). The step (b) utilizes the fact that exponential function is Lipchitz continuous in a compact region. Then we can choose a small enough δ, i.e., $\delta = \theta(1/n)$ such that

$$\|\exp(\delta xR) - \exp(\delta x)R\| = o(\|\exp(x)\|) \tag{23}$$

The step (c) is based on the JL Lemma defined in (12).

Applying the result in (21) to every row vector of matrix A and every column vector of matrix V, one can directly prove that, for any row vector A_i of matrix A,

$$\Pr\left(\|\exp(A_iE^T)FV - \exp(A_i)V\| \le \epsilon\|\exp(A_i)\|\|V\|\right) > 1 - o(1), \tag{24}$$

by setting $k = 5\log(nd)/(\epsilon^2 - \epsilon^3)$. This result does not utilize the low rank property of matrix A (rank(A)=d) and the resultant k has a dependency on sequence length n. We will further prove the choice of k can be constant and independent of sequence length n.

Based on the fact that rank(A)=d, we can find a row submatrix $A_s \in \mathbb{R}^{2d\times d}$ of matrix $\exp(AE^T)FH$ such that rank(A_s)=d. Applying the result in (21) to every row vector of matrix A_s and every column vector of matrix V, and $k = 9\log(d)/(\epsilon^2 - \epsilon^3)$, we can obtain that, for any row vector A_i^s of matrix A^s,

$$\Pr\left(\|\exp(A_i^sE^T)FV - \exp(A_i^s)V\| \le \epsilon\|\exp(A_i^s)\|\|V\|\right) > 1 - o(1), \tag{25}$$

Furthermore, define the matrix $\Gamma \in \mathbb{R}^{n\times 2d}$ as

$$\Gamma = \begin{bmatrix}\exp(AE^T)FV \\ \exp(A)V\end{bmatrix} \cdot \begin{bmatrix}\exp(A_sE^T)FV \\ \exp(A_s)V\end{bmatrix}^{-1} \tag{26}$$

We have that, for any row vector A_i of matrix A, $1 \le i \le n$.

$$\begin{aligned}
\|\exp(A_iE^T)FV - \exp(A_i)V\| =& \|\Gamma_i\exp(A^sE^T)FV - \Gamma_i\exp(A^s)V\| \\
\overset{(a)}{\le}& \left\|[\exp(A^sE^T)FV - \exp(A^s)V]^T\right\|_2\|\Gamma_i\| \\
\overset{(b)}{\le}& \Theta(d)\|\exp(A^sE^T)FV - \exp(A^s)V\|_F \\
=& \Theta(d)\sum_{i=1}^{2d}\|\exp(A_i^sE^T)FV - \exp(A_i^s)V\| \\
\overset{(c)}{\le}& \epsilon\Theta(d)\sum_{i=1}^{2d}\|\exp(A_i^s)\|\|V\| \\
\le& \epsilon\Theta(d)\|\exp(A^s)\|\|V\|
\end{aligned}$$

The above, step (a) utilizes the inequality $\|Ax\| \le \|A\|_2 \cdot \|x\|$, where $\|A\|_2 = \sqrt{\lambda_{\max}(A^TA)}$ ($\lambda_{\max}(\cdot)$ is the largest eigenvalue) is the spectrum norm of a matrix A. The step (b) is based on matrix norm inequality $\|A\|_2 \le \|A\|_F$, where $\|A\|_F = (\sum_{1\le i,j\le n} A_{ij}^2)^{1/2}$ is the Frobenius norm of matrix A. The step (c) is based on the results of (24). □