Linformer: Self-Attention with Linear Complexity

The paper describes a method to reduce the time and space complexity of transformers from $O(n^2)$ to O(n) using a low-rank approximation of the self-attention matrix.

Key takeaways

- Projecting to a lower-dimensional matrix reduces complexity from $O(n^2)$ to O(nk). We need to choose k << n in order to get the most benefit from the architecture.
- The paper shows that this approach gives good performance for a sequence length of 4096, which is **eight times** more than the sequence length used in BERT.
- The main advantage of this approach is a faster inference time as well as a faster training time, leading to a marked improvement in time required to compute results.

1. Problem

Transformer models (described in paper Attention is all you need) have become ubiquitous for a wide variety of problems in natural language processing (NLP) such as text classification, question answering, etc. As opposed to RNN, transformers can be trained in Parallel, but the main efficiency bottleneck in Transformer models is its self-attention mechanism. 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, and thus they are very slow for longer sequences.

2. Recap about Transformers and Self-Attention mechanism

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

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \text{head}_2, \ldots, \text{head}_h)W_0$$

Let us consider the following variables:

n – input sequence length

$\mathbf{d_m}$ – Embedding dimension

$\mathbf{d_k}, \mathbf{d_v}, \mathbf{d_q}$ – hidden Dimensions of projection subspaces

The table below represents the dimension of different matrices computed in self-attention:

W^k, W^q, W^v are the learned matrices during the training process

K, Q, V is the same as the input matrix

The table below represents the dimension of different matrices computed in self-attention:

Matrix	**K,Q,V (input)**	**W^k**	**W^q**	**W^v**	**K x W^k**	**Q x W^q**	**V x W^v**
Dimensions	n x d_m	d_m x d_k	d_m x d_q	d_m x d_v	n x d_k	n x d_q	n x d_v

Hidden dimensions of projection subspaces are all equal hence we have:

$d_k = d_v = d_q = d$

We can write attention for each head as:

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

Looking at the dimensions of content mapping matrix **P:**

QW^Q = (n x d_m) x (d_m x d) = n x d

KW^K = (n x d_m) x (d_m x d) = n x d

Multiplying the above two matrices, we get the following dimension:

$QW^Q (KW^K)^T$ = **(n x d) x (d x n)**

*To compute **P**, we are multiplying two (n x d) matrices, which makes space and time complexity to $O(n^2)$*

3. Solution: Self-Attention is Low Rank

The paper proved that the **context mapping matrix P** is a low-rank matrix and can be approximated in linear time and space. They provided a theoretical analysis by using *distributional Johnson–Lindenstrauss lemma*. They used the following theorem as their basis for the hypothesis. More details about the proof can be found in the paper.

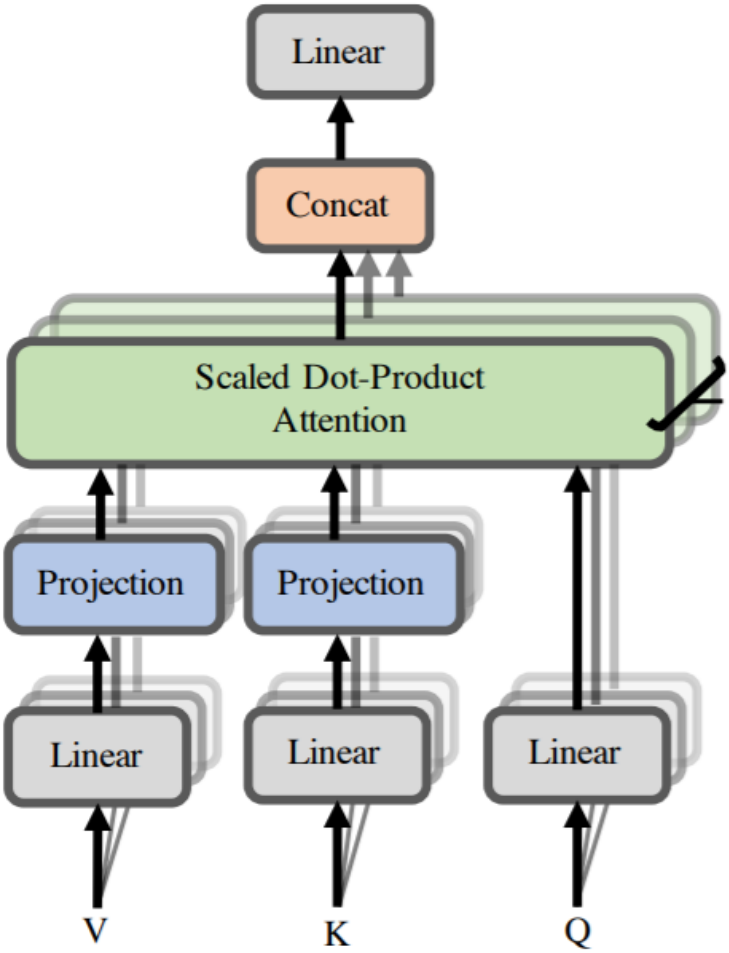
Theorem 1. (self-attention is low rank) *For any* $Q, K, V \in \mathbb{R}^{n\times d}$ *and* $W_i^Q, W_i^K, W_i^V \in \mathbb{R}^{d\times d}$*, for any column vector* $w \in \mathbb{R}^n$ *of matrix* VW_i^V*, there exists a low-rank matrix* $\tilde{P} \in \mathbb{R}^{n\times n}$ *such that*

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

4. Model Architecture

The model architecture is similar to Transformers; they have only added two linear projection matrices to compute keys and value matrix.

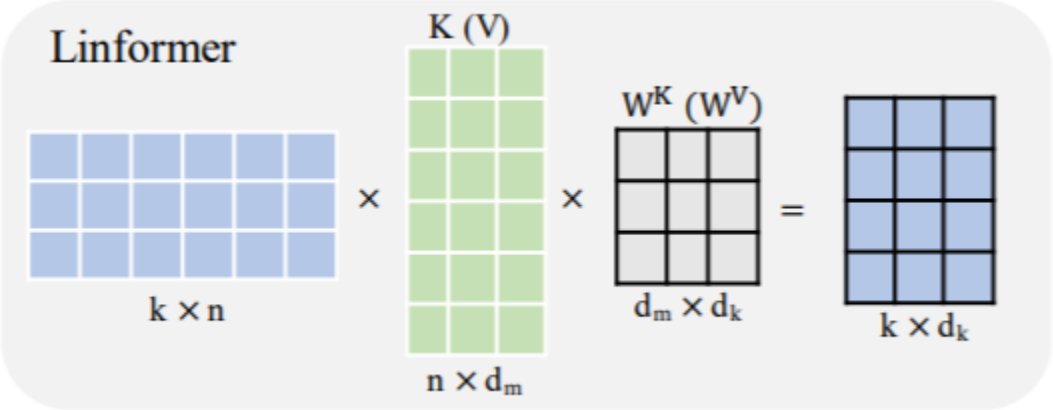
It proposed a linear projection matrix to first reduce the dimensions of key and value matrix, and hence improving on the time and space complexities.



Model Architecture as defined in the paper. The projection matrix added after Values and Key layer is the only difference in Linformer paper.

Here two projection layers have been added while computing the matrix multiplication for Values and Keys. The dimensions for both matrices are *(n x k)*.

Let us assume the two projection matrices are $\mathbf{E_i}, \mathbf{F_i} \in \boldsymbol{R^{n\,x\,k}}$



Linear Projection matrix in Linformer

Following steps are being done to compute the self-attention in Linformer:

- Compute the KW^k, QW^Q, and VW^v as before (the dimension of each of these will be (n x d)
- Project the (n × d)-dimensional key and value layers KW^k and VW^v into (k × d)-dimensional projected key and value layers.
- Compute an (n × k)-dimensional context mapping matrix P using scaled dot-product attention

$$= \underbrace{\text{softmax}\left(\frac{QW_i^Q(E_iKW_i^K)^T}{\sqrt{d_k}}\right)}_{\bar{P}:n\times k} \cdot \underbrace{F_iVW_i^V}_{k\times d},$$

Here multiplying by matrix E_i and F_i reduces the size of the matrix to *(n x k)* instead of *(n x d).*

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 << n, we can significantly reduce the memory and space consumption.

4.1. Comparison between Transformers and Linformers

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

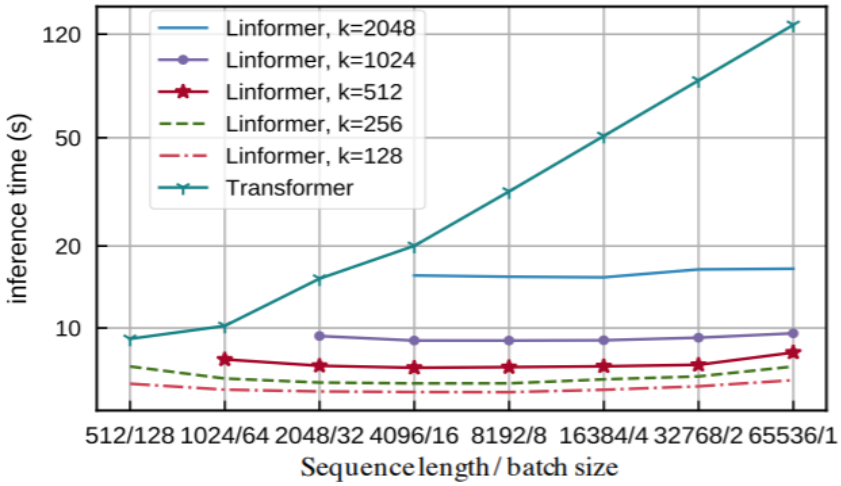
Transformers where the dimension of P is *n x d* and multiplied by (*d x n*) matrix

$$\underbrace{\text{softmax}\left(\frac{QW_i^Q(E_iKW_i^K)^T}{\sqrt{d_k}}\right)}_{\bar{P}:n\times k} \cdot \underbrace{F_iVW_i^V}_{k\times d},$$

Linformer where the dimension of P is *n x k* and multiplied by *(k x d)* matrix

4.2. How to choose K

- The authors found that empirical evidence supports the fact that "*the performance of Linformer model is mainly determined by the projected dimension k instead of the ratio n/k*".
- Even when increasing sequence lengths, it may be fine to keep a relatively low, constant k (the authors showed with k=256 that it still performed almost as good as a vanilla transformer).
- The authors recommend that k = O(d/eps^2), if self-attention wants to be approximated by full attention, with eps error.



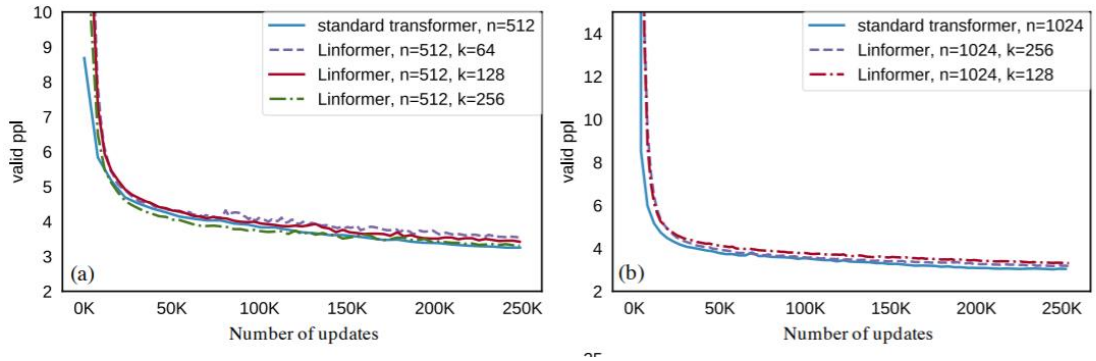
For k = 512, inference time for a sequence length of 4096 is one-fourth to that of a transformer model.

5. Experiments:

Authors experimented the proposed architecture against RoBERTa, which is based on transformers. BookCorpus plus English Wikipedia is used as the 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.

The authors used validation perplexity to compare the results for different values of n and k.

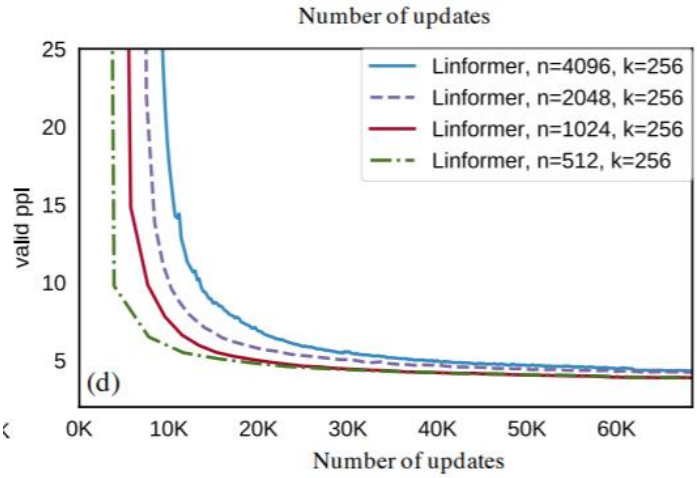
Effect of projected dimension: Validation perplexity curves were computed for both the standard Transformer and the Linformer across different k, for maximum sequence lengths n = 512 and n = 1024.



Validation perplexities for different value of k

- The Linformer performs better as projected dimension k increases.
- 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 longer sequences: The authors tested the validation perplexity for Linformer with n ∈ {512, 1024, 2048, 4096}



Validation perplexities for different sequence length with k = 256

- We can see that the ppl reduces and becomes constant after 50K updates for sequence length of 4096 with k = 256
- We can also see that the sequence length seems independent of k

5.1. Experiments on downstream tasks:

Experiments were also conducted on downstream tasks. Linformer was finetuned on the following tasks:

- IMDB and SST-2 (sentiment classification)
- QNLI (natural language inference)
- QQP (textual similarity)

The results of Linformer were compared with **RoBERTa,** 12-layer BERT-base, and 6-layer distilled BERT.

All the models, including the Transformer baselines, were pretrained with the same objective, pretraining corpus, and up to 250k updates (**Linformer took much less wall-clock time to get to 250k updates and was trained for less time**). Results are listed below:

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

- The Linformer model (n = 512, k = 128) has the comparable downstream performance to the RoBERTa model, and in fact even slightly outperforms it at k = 256.
- 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 supports the notion that the **performance of the Linformer model is mainly determined by the projected dimension k instead of the ratio n/k**

6. Conclusion

The paper demonstrated that the stochastic matrix formed by the self-attention mechanism is low-rank. Through a combination of theoretical and empirical analysis, it demonstrated that the proposed approach is O(n) with respect to the sequence length.

References

1. Linformer Paper
2. RoBERTa: A Robustly Optimized BERT Pretraining Approach
3. Attention Is All You Need
4. The Illustrated Transformer
5. Pytorch Implementation of Linformer
6. Linformer: Self-Attention with Linear Complexity (Paper Explained)