Linformer: Self-Attention with Linear Complexity

Jacques Maubert
MASH

Stanislas Boiton Ledenko
MASH

Maxence Lasbordes
MASH

Abstract

The transformers and the attention mechanism, introduced in the paper *Attention Is All You Need*, has revolutionized machine learning by its ability to focus on specific parts of the input data, allowing models to capture long-range dependencies and context effectively. However, this architecture has been shown to be highly computationally expensive. The Linformer addresses this by leveraging low-rank approximations, reducing the complexity of self-attention from $O(n^2)$ to $O(nk)$. While this trade-off incurs minor information loss, it significantly improves efficiency and inference speed, enabling transformer models to be trained on smaller hardware. This innovation democratizes access to advanced AI technologies and makes training large-scale models feasible for resource-constrained applications, such as document translation and genomic analysis.

1 Analysis of the paper

1.1 Introduction

In this report, we will explore and present the *Linformer: Self-Attention with Linear Complexity* paper published in 2020 by Facebook AI (currently Meta). This paper discusses the efficiency bottleneck in transformer models, particularly the expensive self-attention mechanism in computation. Transformer models like BERT and GPT have revolutionized PNL tasks but they suffer from high computational costs due to the quadratic complexity of the self-attention mechanism.

We will provide a detailed description of the key ideas of the Linformer, with emphasis on the authors' approach to reducing the complexity of personal attention by exploiting its low-rank properties. We will also examine the limitations of this approach, including possible performance trade-offs and practical considerations. Finally, we will conduct experiments to validate the article's assertions, providing insights into how the proposed method works in practice and demonstrating our understanding of the concepts presented. The aim of these experiments is to highlight the strengths and obstacles of applying this method to transformer models.

1.2 Related Works

Several prior papers have been published that try to address the Transformer bottleneck of quadratic time complexity. Some of the most efficient techniques are:

Sparse Attention (Child et al., 2019; Qiu et al., 2019; Beltagy et al., 2020). Introducing Sparsity into attention layers, which suffers from a 2% performance drop but has a 20% speed up.

LSH Attention (Kitaev et al., 2020) used locally-sensitive hashing (LSH) to reduce the self-attention complexity to $O(nlog(n))$. The efficiency gains only appear on sequences with length larger than 2048.

Mixed Precision, which uses half-precision or mixed-precision floating points to reduce memory usage and improve training performance. Quantization Aware Training can further optimize this process.

Knowledge Distillation, where the idea is to transfers knowledge from a large "teacher" model to a smaller "student" model for efficient inference. However, student models often perform worse, and it doesn't speed up the teacher's training. So it doesn't really solve our issue.

Finally, **Improving Optimizer Efficiency**, two main techniques, Microbatching, which splits batches into smaller microbatches to save memory but doesn't speed up inference, and Gradient Checkpointing, which saves memory by recomputing some activations, trading time for memory without improving inference.

Basically, every technique has limited efficiency improvements or inherent limitations.

1.3 Core Insight: Low-Rank Approximations in Linformer

The intuition of the authors of the article was that the context mapping matrix is approximately low rank. Recall that Let $Q, K, V \in \mathbb{R}^{n \times d_m}$ be the input embedding matrices, where n is the sequence length, d_m is the embedding dimension and h is the number of heads. Each head is defined as follows:

$$\text{Head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) = \text{softmax}\left(\frac{QW_i^Q(KW_i^K)^\top}{\sqrt{d_k}}\right)VW_i^V \tag{1}$$

$$P = \text{softmax}\left(\frac{QW_i^Q(KW_i^K)^\top}{\sqrt{d_k}}\right) \tag{2}$$

where W_i^Q, W_i^K, W_i^V are learnable projection matrices for the head i -th, and d_k is the dimension of the key vectors.

The authors validated their hypothesis with empirical results before providing a mathematical proof. They did that by analyzing the cumulative sum of the context matrix regularized eigenvalues on both a large model and roberta, on the IMDB and Wiki103 datasets. The results of this analysis showed that the first 128 eigenvalues contain 90% of the information. We did the same analysis on the IMDB data set and on random distributions. Our results can be seen on Figure 1.

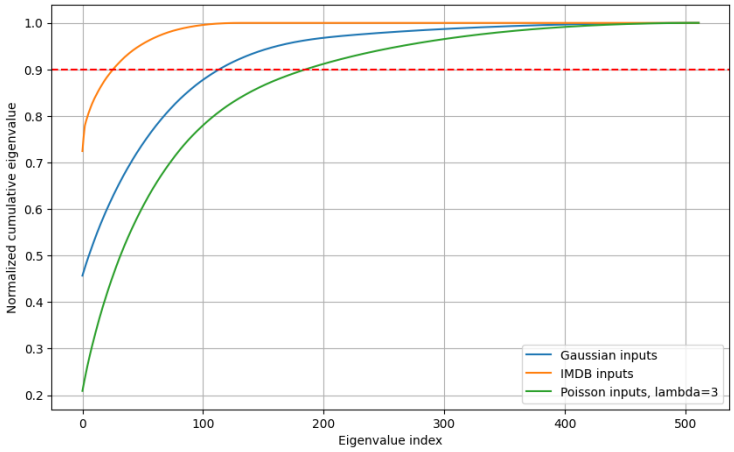

Figure 1: Our Spectrum analysis of the context mapping matrix

The idea is to draw $Q, K \sim D$ with D being gaussian, poisson, or from pre-embedded IMDB data. For the IMDB data, the context matrix is indeed of low rank ~ 100, but for the other distribution, P is just approximately of low rank.

1.4 Model

The main idea of the new self-attention proposed in the Linformer paper is to add two linear projection matrices $E_i, F_i \in \mathbb{R}^{n \times k}$ when computing the key and value.

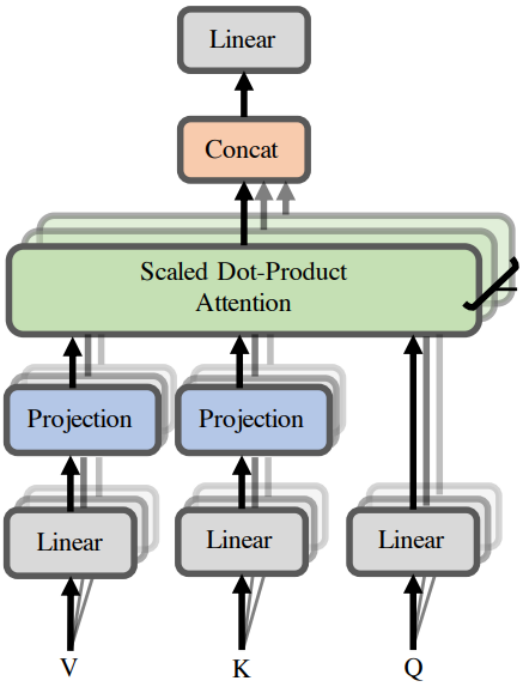

Figure 2: The architecture of the multihead linear self-attention used in the Linformer

This implementation leverages the fact that the context mapping is low-rank. Consequently, we modify the Transformer's self-attention mechanism.

$$\text{Head}_i = \text{Attention}(QW_i^Q, E_iKW_i^K, F_iVW_i^V) = \text{softmax}\left(\frac{QW_i^Q(E_iKW_i^K)^\top}{\sqrt{d_k}}\right) \cdot F_iVW_i^V \quad (3)$$

$$P' = \text{softmax}\left(\frac{QW_i^Q(E_iKW_i^K)^\top}{\sqrt{d_k}}\right) \quad (4)$$

In this configuration the shape of *P'* is $(n \times k)$ and the shape of $(F_iVW_i^V)$ is $(k \times d)$

All these operations require only *O(nk)* time and space complexity, which represents a significant reduction in computation compared to Transformers, especially for longer sequences.

1.5 Findings: Performance Comparison

The Linformer has similar performances compared to a vanilla transformer, with perplexity scores on benchmarks like IMDB, SST-2, and QNLI matching those of the RoBERTa-base model. Its linear time and memory complexity make it particularly efficient for longer sequences, as inference times grow significantly slower than those of vanilla transformers.

1.6 First Thoughts

The Linformer paper addresses a significant bottleneck in transformer models by reducing self-attention complexity from $O(n^2)$ to $O(nk)$, enabling their effective deployment in environments with limited resources. There is one key insight, the necessity of adjusting k the projected dimension, based on the embedding dimension d. Empirical results show that while reducing k leads to minor information loss, the trade-off enables a reduction in time complexity, making the Linformer highly effective for long sequences.

Reducing computational costs has significant implications for accessibility and sustainability. The Linformer could enable smaller organizations to train and deploy transformer models using limited hardware resources, democratizing access to advanced AI models. Furthermore, its lower resource demands contribute to environmental sustainability by reducing the energy consumption associated with large-scale AI models.

From an ethical standpoint, while the Linformer introduces only positive technical improvements, its applications warrant careful regulation to ensure responsible use. This underscores the importance of governing AI deployments in areas like surveillance or misinformation.

2 Experiments

2.1 Model Implementation Details

We implemented both the vanilla transformer and the Linformer from scratch to compare their computational efficiency and validate the claims of the paper. The vanilla transformer served as a baseline. Both models were trained using identical hyperparameters for consistency. Due to hardware constraints, our implementation focused on smaller-scale models, emphasizing key principles over full-scale performance.

2.2 Experimental Results: Efficiency Gains

We first wanted to replicate some of the results obtained in the paper with our own implementation of the Linformer model.

	Linformer	**Transformer**
Parameters (M)	6.11	5.98
d_ff	1024	1024
d_embed	256	256
n_heads	4	4
n_layers	1	1
max_length	512	512
k	128	-

Table 1:
Hyperparameters and model size of the Linformer and Transformer models used for the experiments.

As shown in Table 1, the difference between the Linformer and Transformer models lies in the self-attention layer and the addition of two extra linear layers of shape $(max\_length \times k)$. Therefore, the difference in the number of parameters becomes more significant as the *max_length* increases. This means that even for small-sized models, as in these experiments, when increasing the *max_length* the difference in size between the two models can quickly become substantial. But the longer the sequence and the faster the Linformer is supposed to be compared to the vanilla Transformer.

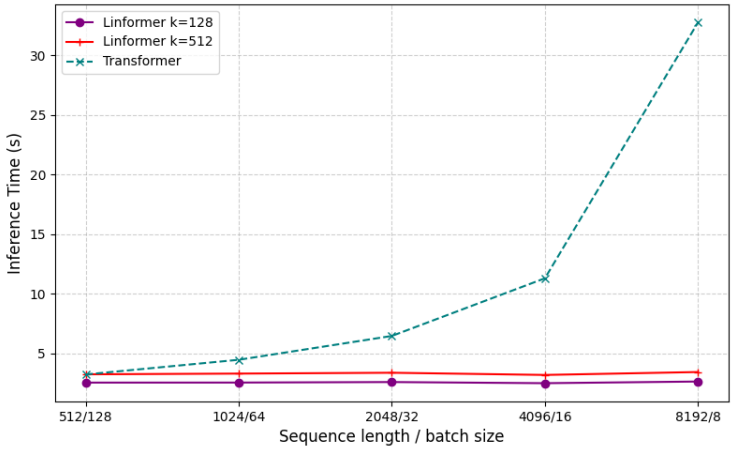

Figure 3: Inference time vs. sequence length with our own implementation of the Linformer model and the vanilla Transformer model from scratch.

The Figure 3 illustrates the difference in inference time between the Linformer and the vanilla transformer as sequence length increases. While the vanilla transformer exhibits quadratic growth, the Linformer maintains nearly constant inference time, showcasing the efficiency of its linear complexity. Notably, setting $k = 128$ results in significantly faster inference compared to $k = 512$, demonstrating the trade-off between precision and computational cost.

2.3 Discussion

Our implementation of the Linformer and vanilla transformer focused on validating the principles of linear self-attention with limited computational resources. Due to hardware constraints, we were unable to train full-scale models or compare perplexity across large datasets. Instead, we performed smaller-scale experiments to confirm efficiency gains, which aligned well with the theoretical expectations from the paper.

Future work could involve scaling up the implementation to larger models and datasets, enabling a detailed comparison of accuracy metrics, such as perplexity, across benchmarks like WikiText-103 and GLUE. Additionally, exploring dynamic k-values based on sequence characteristics could provide further insights into the Linformer's adaptability.

Key Insights and Lessons Learned: One of the paper's most valuable insights is the necessity of adjusting k, the projected dimension, based on the embedding dimension d. Here, d is the size of the vector space where token embeddings are represented, typically 256 or 512 in standard transformers. By ensuring k aligns with d and task requirements, the Linformer achieves an optimal balance between preserving information and reducing complexity.

For example, in the spectrum analysis experiment, with $n = 512$, setting $k \leq 100$ resulted in significant information loss—approximately 10% for Gaussian distributions and 20% for Poisson distributions. However, this trade-off enabled a fivefold reduction in time complexity, showcasing the Linformer's efficiency in handling long sequences. These results underline the Linformer's suitability for tasks such as document translation and genomic sequence analysis, where long sequences are prevalent, and its efficiency gains outweigh the compromises in precision.

Future Directions: The paper provides a solid experimental foundation, validating its claims across various datasets and tasks. The spectrum analysis of the self-attention matrix is particularly compelling, as it empirically demonstrates the low-rank property central to the Linformer's design. However, the speed improvements achieved by this design also lead to a noticeable diminution in output quality.

3 Conclusion

To conclude, the Linformer seems like a highly efficient self-attention mechanism with O(n) complexity, leveraging the theoretical and empirical observation that the self-attention matrix is low-rank, it still has trade-offs, such as reduced output quality. But new advancements adress this issue by leveraging techniques like FlashAttention, which significantly reduces inference time by caching previous computations without compromising model quality. Additionally, incorporating quantization methods can further enhance inference speed, making the model more suitable for production-scale deployment.

4 Appendix

4.1 AI Usage

We used ChatGPT, Mistral AI, and Writefull to assist in refining the report's structure and improving sentence clarity. These tools were particularly helpful in summarizing complex concepts, ensuring consistency in terminology, and enhancing overall readability.