Evaluating Linformer's performance on the WMT14 EN-DE machine translation task

Marco Pampaloni

Department of Computer Science

October 13, 2024

Contents

1 Introduction

The Transformer architecture, since its introduction in 2017 [Vas+17], has revolutionized the field of natural language processing (NLP), reaching state of the art in various downstream tasks. Despite its massive parallelism capabilities, the Transformer struggles with longer sequence lengths due to its attention mechanism, which scales quadratically with the number of input tokens. This is a problem at both training and inference time.

For this reason, there has been a huge research effort in recent years to develop faster attention mechanisms, either exploiting the IO properties of the hardware these models run on [Dao+22], or by approximating the result of scaled dot product attention (SDPA).

The *Linformer* architecture [Wan+20] is an example of the latter approach. The authors first empirically show that the attention matrix is often low-rank, meaning that it could be approximated with an SVD decomposition by only keeping the largest singular values. This would of course introduce additional asymptotical complexity to the method, so the authors propose the adoption of linear projections on the keys and values matrices K, V to reduce their dimensionality and drive the computational complexity of the attention mechanism from $O(n^2)$ to $O(kn)$. The authors further show that the choice of k does not depend on the sequence length n, so that the scaling can be considered linear in n.

The standard multi head attention (MHA) introduced by [Vas+17] is computed as follows

$$\begin{aligned} \text{MultiHead}(Q, K, V) &= \text{Concat}(\text{head}_1, \dots, \text{head}_h) W^O \\ \text{where head}_i &= \text{Attention}(QW^Q, KW^K, VW^V) \\ &= \underbrace{\text{softmax}\left(\frac{QW^Q (KW^K)^T}{\sqrt{d_{model}}}\right)}_{P_i} VW^V \end{aligned} \tag{1}$$

The Linformer attention first projects the key and value matrices into a space of dimension $\mathbb{R}^{k \times d}$ with projection matrices $E, F \in \mathbb{R}^{k \times n}$ and then computes MHA as before:

$$\overline{\text{head}}_i = \text{Attention}\left(Q, E_i K W_i^K, F_i V W_i^V\right) \tag{2}$$

This produces an attention matrix $\bar{P}_i \in \mathbb{R}^{n \times k}$, which is computed in time linear with n. Since the projection matrices E, F are fixed in size before training, an obvious donwside of this approach is that the maximum sequence length n has to be known beforehand and the model cannot then scale to inputs with more tokens than this value. One workaround is to set the maximum sequence length n to a large number and handle shorter inputs by slicing the projection matrices along their columns before multiplying them with the inputs K and V.

This work aims at replicating the results of [Vas+17] and [Wan+20] and comparing the two architecture on the WMT14 EN-DE machine translation task.

1.1 Masking

In a standard Transformer architecture, sequences are usually batched together in order to exploit the model's parallelism. This requires padding to be applied to the batch,

but pad tokens should not contribute to the attention's output. This can be achieved by masking the attention matrix, setting zeroing out its elements in correspondence with input pad tokens. This if often done by setting each corresponding element to $-\infty$ prior to applying the row-wise softmax.

In Linformer this method cannot be applied, as the attention matrix is linearly projected to a lower-dimensional space. Instead, we apply masking by zeroing out elements in Q, K, V corresponding to pad tokens. This ensures that the pad tokens do not contribute to the final attention result.

1.1.1 Causal masking

The standard Transformer was trained on the WMT14 EN-DE machine translation dataset, adopting an encoder-decoder architecture. The self-attention mechanism in the decoder's layers need to employ causal masking in order for future (right) tokens not to contribute to the output of past positions. In the Transformer this again can be achieved by masking, in this case by adding an upper triangular matrix filled with $-\infty$ values to the pre-softmax attention matrix.

Linformer, which was originally developed for encoder-only architectures, does not allow for causal masking, because however you mask the resulting pre-activation attention matrix, future tokens leak into the past due to the linear projections of the K and V matrices.

The Linformer attention mechanism thus cannot be applied in the self-attention layers of the decoder, while it can safely be used in the cross-attention stages because of the lack of causal requirements. This hinders the full scaling potential of the encoder-decoder Linformer architecture, which is empirically shown in Sections 7.2 and 7.3.

2 Prior work

TODO

3 Data

TODO

- Tokenizer used: BART byte-encoding 50k vocab size
- Dataset description (WMT14 EN-DE): 50% used for memory reasons
- Max length set to 256
- Padding on the right
- Padding trimmed within batches
- Average sequence length in training dataset
- Training, validation, test split
- Use of test dataset

Model	PPL (test)	BLEU (test)
Transformer	**3.41**	29.92
Linformer (k=32)	3.96	**30.08**
Linformer (k=64)	3.84	27.74

Table 1: Linformer performance against a vanilla Transformer model on the WMT14 EN-DE (test) dataset. The Linformer has slightly worse perplexity than the Transformer, but their BLEU scores are comparable.

4 Architecture

5 Hardware

5.1 CPU bound

6 Experiments

6.1 Docker container (NVCR)

6.2 Mixed precision training

7 Results and Analysis

7.1 Model performance

TODO

- Perplexity is wrong: need to compute first all log likelihoods across all batches and then exponentiate them
- BLEU might be missing a dimension in the reference corpus

7.2 Training time

7.3 Inference time

8 Conclusions

Bibliography

References

[Vas+17] Ashish Vaswani et al. *Attention Is All You Need*. Dec. 5, 2017. arXiv: 1706.03762 [cs]. URL: http://arxiv.org/abs/1706.03762. Pre-published.

[Wan+20] Sinong Wang et al. *Linformer: Self-Attention with Linear Complexity*. June 14, 2020. DOI: 10.48550/arXiv.2006.04768. arXiv: 2006.04768 [cs, stat]. URL: http://arxiv.org/abs/2006.04768. Pre-published.

[Dao+22] Tri Dao et al. *FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness*. June 23, 2022. DOI: 10.48550/arXiv.2205.14135. arXiv: 2205.14135 [cs]. URL: http://arxiv.org/abs/2205.14135. Pre-published.