Efficient Attention

Sahil Chaudhary* **Mrigank Pawagi*** **Rohit Jorige*** **Jajapuram Nagasai***

Abstract

The transformer architecture revolutionized sequence transduction with its novel "Scaled Dot-Product Attention" by outperforming previous recurrent and convolutional models, while dispensing recurrence and convolutions entirely. However, these transformers are prohibitively slow for very long sequences due to their $\mathcal{O}(N^2)$ time complexity in the forward pass, where N is the sequence length. By expressing the self-attention in these transformers as a linear dot-product of kernel feature maps and making use of the associativity property of matrix products can reduce the complexity to $\mathcal{O}(N)$. In addition to these theoretical results, recent empirical results suggest that training of and inference from these models can be made faster and more scalable. One such technique, 'Low-Rank Adaption' or LoRA has proven to be particularly valuable in the scalable adaptation of pre-trained models to new tasks through fine-tuning. In this article, we will discuss the results mentioned above. Besides reproducing some of the experiments in these papers, our contributions include new experiments to explore these results.

1 Attention Architecture

The self-attention mechanism makes it possible to capture dependencies between different positions in a sequence by simultaneously attending to all positions. The "Scaled Dot-Product Attention" proposed by Vaswani et al. [6] for an input sequence $x \in \mathbb{R}^{N \times F}$ is computed as $\text{Attention}(x) = \text{softmax}(QK^T/\sqrt{D})V$ where $Q = xW_Q$ (queries), $K = xW_K$ (keys) and $V = xW_V$ (values). Note that $W_Q, W_K \in \mathbb{R}^{F \times D}$ and $W_V \in \mathbb{R}^{F \times F}$ are learned weights. Then the attention vector for the i-th position is given by the i-th row,

$$\text{Attention}(x)_i = \text{softmax}\left(\frac{(QK^T)_i}{\sqrt{D}}\right)V = \text{softmax}\left(\frac{Q_i K^T}{\sqrt{D}}\right)V = \frac{\sum_{j=1}^{N} \exp(Q_i^T K_j/\sqrt{D})V_j}{\sum_{j=1}^{N} \exp(Q_i^T K_j/\sqrt{D})}$$

Since $\exp(Q_i K_j^T/\sqrt{D})$ has to be determined and stored for every $i, j \in [N]$, this leads to $\mathcal{O}(N^2)$ time and memory complexity (precisely $\mathcal{O}(N^2 \max\{D, F\})$).

2 Towards Efficient Attention – Linearized Attention

The exponential similarity function above can be replaced with any map $\text{sim} : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}_+$. This includes all kernels between the two spaces. Katharopoulos et al. [3] show that given such a kernel with a feature representation ϕ, we can rewrite the attention computation as

$$\text{Attention}(x)_i = \frac{\sum_{j=1}^{N} \phi(Q_i)^T \phi(K_j) V_j}{\sum_{j=1}^{N} \phi(Q_i)^T \phi(K_j)} = \frac{\phi(Q_i)^T \sum_{j=1}^{N} \phi(K_j) V_j^T}{\phi(Q_i)^T \sum_{j=1}^{N} \phi(K_j)}$$

Since $\sum_{j=1}^{N} \phi(K_j) V_j^T$ and $\sum_{j=1}^{N} \phi(K_j)$ can be computed once reused for every query, this formulation has $\mathcal{O}(N)$ time and memory complexity (precisely $\mathcal{O}(NCF)$ where C is the dimension of the range space of the feature map).

2.1 Transformers are RNNs

The transformer architecture can further be used to efficiently train autoregressive models by masking the attention computation such that a position cannot be influenced by the subsequent positions. Then the attention for the i-th position

*All authors contributed equally to this work.

can we written as given below. Note that we define $S_i = \sum_{j=1}^{i} \phi(K_j)V_j^T$ and $Z_i = \sum_{j=1}^{i} \phi(K_j)$.

$$\text{Attention}(x)_i = \frac{\phi(Q_i)^T \sum_{j=1}^{i} \phi(K_j)V_j^T}{\phi(Q_i)^T \sum_{j=1}^{i} \phi(K_j)} = \frac{\phi(Q_i)^T S_i}{\phi(Q_i)^T Z_i}$$

Note that, S_i and Z_i can be computed from S_{i-1} and Z_{i-1} in constant time. This enables the transformer to be formulated as an RNN using the following recurrence relations. We set $S_0 = 0$ and $Z_0 = 0$, and for all $i \geq 1$,

$$S_i = S_{i-1} + \phi(K_i)V_i^T = S_{i-1} + \phi(x_i W_K)(x_i W_V)^T$$
$$Z_i = Z_{i-1} + \phi(K_i) = Z_{i-1} + \phi(x_i W_K)$$

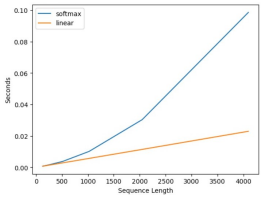
The resulting RNN thus has two hidden states – the attention memory s and the normalizer memory z. Then the output for the i-th transformer layer is given by

$$T_i(x) = f_i\left(\left[\frac{\phi(x_k W_Q)^T S_k}{\phi(x_k W_Q)^T Z_k} + x_k\right]_{k=1}^{N}\right)$$

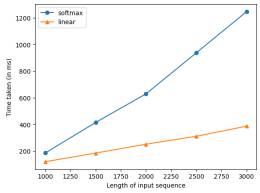
2.2 Experiments

2.2.1 Performance on random sequences

To examine the claims regarding time complexity, we experimented with random sequences of different lengths and measured the time taken to measure the attention in each case. We also measured the total time taken in the forward and backward passes for each sequence.



Total time taken for forward and backward passes vs Sequence length



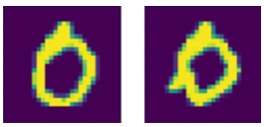
Time taken for attention calculation vs Sequence length

2.2.2 Occluded Image Completion

We trained a softmax transformer (using the attention from Vaswani et al. [6]) and a linear transformer (using the attention from Katharopoulos et al. [3]) for autoregressive image generation on the MNIST dataset [4]. We then compared the performance of the two models on image completion from occluded images.

MNIST

Occluded MNIST



Outputs

The linear transformer achieved faster image completion on occluded inputs in comparison to the softmax transformer. To evaluate the two transformers, we generated 600 images with each of the two models and compared the accuracy of a neural network trained on the MNIST dataset [4] on these two datasets. The accuracies were 87% and 86.3% on images generated by the linear and the softmax transformers respectively.

Number of Images Completed	Linear	Softmax
100	33.45s	367.91s
200	67.46s	811.99s
300	76.46s	1248.15s

3 Low-Rank Adaptation (LoRA) of Large Language Models

Due to the compute requirements of large-scale training, fine-tuning is a widely adopted paradigm for adapting models pre-trained on general domain data to particular tasks or domains. However, for large models even fine-tuning may

be prohibitively expensive. Work from Li et al. [5] and Aghajanyan et al. [1] has demonstrated that models may be over-parameterized and may actually reside on a low intrinsic dimension. Inspired from these results, Hu et al. [2] showed empirically that the change in weights during model adaptation also has a low intrinsic rank. This not only makes it possible to achieve comparable accuracies with faster fine-tuning (due to fewer trainable parameters) but also makes it possible to easily switch different adaptators in and out during deployment. A matrix $M \in \mathbb{R}^{n \times m}$ can be rank decomposed as $M = UV$ where $U \in \mathbb{R}^{n \times r}$, $V \in \mathbb{R}^{r \times m}$ and r is the rank of M. This reduces the number of trainable parameters from mn to $r(n + m)$. Note that $r(n + m) \ll nm$ for small r. This decomposition of the update matrices during fine-tuning lies at the heart of LoRA. These decomposed update matrices can be quickly switched in and out for different applications. Note that r is practically a hyperparameter during fine-tuning.

3.1 Experiments

We train a base neural network on an image classification task on the MNIST dataset [4]. Our base model was composed of three linear layers which together had 55.1K trainable parameters. This model has a test accuracy of approximately 93.2%. We then created our variants of MNIST, namely Quantized MNIST, Rotated MNIST, and Inverted MNIST. These are illustrated below through an example.

MNIST Quantized MNIST Rotated MNIST Inverted MNIST

The accuracies of our base model on these modified datasets were approximately 85.58%, 12.38%, and 5.52% respectively.

3.1.1 Full Fine-Tuning

We first fine-tuned our base model on the three datasets by modifying all 55.1K parameters. Our fine-tuned models achieved accuracies of approximately 93.57%, 91.97%, and 76.41% on Quantized MNIST, Rotated MNIST, and Inverted MNIST respectively. These form our baselines for the fine-tuned models.

3.1.2 Fine-Tuning with LoRA

We then fine-tuned our base model on the three datasets using LoRA, with different values of r. We found that our fine-tuned models achieved accuracies comparable to their full fine-tuned counterparts with fewer trainable parameters. However due to the small size of our models, we could not observe any time improvements in training. The accuracies of our fine-tuned models in each domain and for each value of r are given below.

r	Trainable Parameters	Quantized MNIST	Rotated MNIST	Inverted MNIST
1	1.1K	91.20%	37.53%	16.23%
2	2.1K	91.30%	49.01%	17.92%
4	4.2K	91.42%	69.10%	16.32%
8	8.4K	91.39%	77.49%	32.19%
16	16.8K	91.72%	86.95%	62.26%
32	33.6K	92.31%	89.50%	68.06%
64	67.2K	93.19%	90.41%	71.88%

3.1.3 Full Training with LoRA

We further explore whether LoRA can be used to train models from scratch, to observe if models have intrinsically low ranks. We trained separate models on each of our three modified MNIST datasets as well as the original MNIST. The accuracies of these models are given below along with the respective r values chosen.

r	Trainable Parameters	MNIST	Quantized MNIST	Rotated MNIST	Inverted MNIST
1	1.1K	56.79%	23.50%	25.91%	22.21%
2	2.1K	71.90%	37.48%	43.96%	45.81%
4	4.2K	84.44%	64.87%	62.60%	69.67%
8	8.4K	89.12%	77.96%	82.39%	83.11%
16	16.8K	92.64%	88.2%	86.76%	87.38%
32	33.6K	93.98%	90.13%	90.62%	90.25%
64	67.2K	94.85%	91.66%	91.85%	86.01%

4 Acknowledgements

We thank Dhruva Kashyap, one of the teaching assistants in UMC 203 and one of the last Emacs users, for his unwavering technical and emotional support throughout the preparation of this report. We are also extremely grateful to Professor Chiranjib Bhattacharyya for providing us with the opportunity to explore this topic through a graded term-paper. The code for the experiments in section 2 was adapted from code provided by Katharopoulos et al. [3] in colab.research.google.com/drive/1BV4OaWRAHYGeimO0cqHD86GfnfgUaKD4. We also used the provided pre-trained weights for both the softmax and linear transformers used in our experiments. The code for the experiments in section 3 was adapted from github.com/sunildkumar/lora_from_scratch.

All of our code is released publicly at github.com/mrigankpawagi/EfficientAttention-TermPaper and can be used to reproduce the experiments mentioned in this paper.

References

[1] Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7319–7328, Online, August 2021. Association for Computational Linguistics.

[2] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

[3] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: fast autoregressive transformers with linear attention. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

[4] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

[5] Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. Measuring the intrinsic dimension of objective landscapes. In *International Conference on Learning Representations*, 2018.

[6] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.