Efficient Attention

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

Abstract

The transformer architecture proposed by Vaswani et al. revolutionized sequence transduction with its novel "Scaled Dot-Product Attention" by outperforming previously dominant 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. Katharopoulos et al. expressed the self-attention in these transformers as a linear dot-product of kernel feature maps and made use of the associativity property of matrix products to reduce the complexity to $\mathcal{O}(N)$. In addition to these theoritical 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 been proposed by Hu et al. based on previous theoretical results by Li et al. and Aghajanyan et al. LoRA has proven to be particularly valuable in 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

Attention mechanisms have become an integral part of complex sequence modeling and transduction models in various task, allowing modeling of dependencies without regard to their distance in the input or output sequences. Let $x \in \mathbb{R}^{N\times F}$ denote a sequence of N feature vectors of dimensions 'F'. As proposed by Vaswani et. al., transformer is a function $T : \mathbb{R}^{N\times F} \to \mathbb{R}^{N\times F}$ defined by the composition of L transformer layers $T_1(\cdot), T_2(\cdot), \cdots, T_i(\cdot), \cdots, T_L(\cdot)$, where $T_i(\cdot)$ is defined as:

$$T_i(x) = f_i(A_i(x) + x) \tag{1}$$

The function $f_i(\cdot)$ transforms each feature independently and implemented with some layer of feedforward Network. $A_i(\cdot)$ is the heart of our transformer i.e. attention. Query, Key and values are defined as follows: $Q = xW_Q$ where, $W_Q \in \mathbb{R}^{F\times D}$, then $Q \in \mathbb{R}^{N\times D}$, $K = xW_K$ where, $W_K \in \mathbb{R}^{F\times D}$, then $K \in \mathbb{R}^{N\times D}$, $V = xW_V$ where, $W_V \in \mathbb{R}^{F\times F}$, then $V \in \mathbb{R}^{N\times F}$. Let's measure the match using sim function, where $sim : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}_+$. According to definition of attention,

$$A_i(x) = V' = \left[\frac{\sum_{j=1}^{N} sim(Q_k, K_j)V_j}{\sum_{j=1}^{N} sim(Q_k, K_j)}\right]_{k\in\{N\}} \tag{2}$$

Then the entries of this matrix is, $\frac{\sum_{j=1}^{N} sim(Q_k,K_j)V_j}{\sum_{j=1}^{N} sim(Q_k,K_j)}$. This is more generalized version of attention from whatever vaswani et. al came up with. If one just substitute the similarity function with $sim(q, k) = exp(\frac{q^T K}{\sqrt{D}})$, we get

$$A_i(x) = softmax(\frac{QK^T}{\sqrt{D}})V \tag{3}$$

1.1 Transofrmers are RNNs

The definition of attention in $eq^n - 2$ is generic and can be use to define several other attention implementation. Now, its upto you to decide a function that mimics the properties of similarity function. This includes all kernels $K : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}_+$. If we adapt kernels for the measure of match, entries of $A_i(x)$ would be $\frac{\sum_{j=1}^{N} K(Q_k,K_j)V_j}{\sum_{j=1}^{N} K(Q_k,K_j)}$.

Given such a kernel with a feature representation $\Phi(x)$, we can rewrite each entries of $A_i(x)$ as,

$$V'_k = \frac{\sum_{j=1}^{N} \Phi(Q_k)^T \Phi(K_j)V_j}{\sum_{j=1}^{N} \Phi(Q_k)^T \Phi(K_j)} = \frac{\Phi(Q_k)^T \sum_{j=1}^{N} \Phi(K_j)V_j^T}{\Phi(Q_k)^T \sum_{j=1}^{N} \Phi(K_j)} \tag{4}$$

Note that, the softmax attention has the computational cost of $O(N^2)$. On the other hand, the new kernel based attention only has computational cost of $O(N)$. Transformer architecture can be used to efficiently train autoregressive models by masking the attention computation. Let $S_k = \sum_{j=1}^{k} \Phi(K_j)V_j^T$ and $Z_k = \sum_{j=1}^{k} \Phi(K_j)$. i.e.

$$V'_k = \frac{\Phi(Q_k)^T \sum_{j=1}^{k} \Phi(K_j)V_j^T}{\Phi(Q_k)^T \sum_{j=1}^{k} \Phi(K_j)} = \frac{\Phi(Q_k)^T S_k}{\Phi(Q_k)^T Z_k} \tag{5}$$

Note that, S_k and Z_k can be computed from S_{k-1} and Z_{k-1} in constant time. This motivates us to see relation between Linear Transformer and RNN. If Linear Transformer with the masking introduced is thought as RNN, we have two hidden states, viz. S and Z. Assume the value of S and Z to be 0 for recurrence. Then,

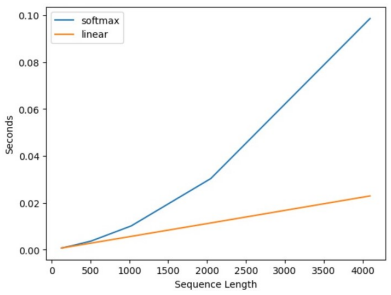
$$S_0 = 0,\ S_k = S_{k-1} + \Phi(K_k)V_k^T = S_{k-1} + \Phi(x_k W_K)(x_k W_V)^T$$
$$Z_0 = 0,\ Z_k = Z_{k-1} + \Phi(K_k) = Z_{k-1} + \Phi(x_k W_K)$$
$$T_i(x) = f_i\left(\left[\frac{\Phi(x_k W_Q^T)S_k}{\Phi(x_k W_Q^T)Z_k} + x_k\right]_{k \in N}\right)$$

Theoretically, it just not only shows that linear transformer are better than softmax based transformer but also, it satisfies few of the properties of RNN which motivates us to experiment Transformer in autoregressive tasks.

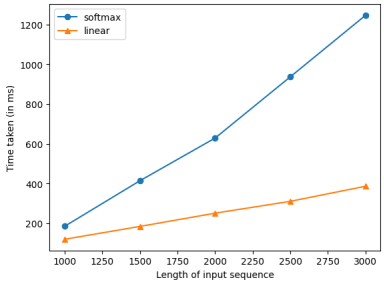
1.2 Experiments

1.2.1 Performance on random sequences

To test the claim made for time complexity, we experimented on various length of random sequences for forward pass and backward pass in both of the models. We also experimented with the time taken to measure the attention in each of the case on random sequences.



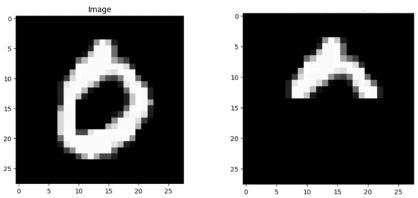
Time taken for forward and backward pass vs sequential length



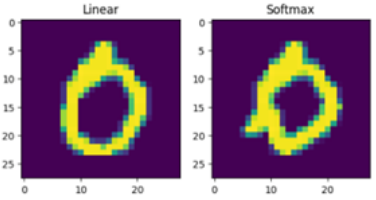
Time taken for attention measure vs sequential length

1.2.2 Occluded Image Completion

We experimented the linear transformer over image generation tasks. We evaluate the model on image generation with autoregressive transformers on the widely used MNIST dataset. The architecture for this experiment is same to Katharopoulos et. al. Thanks to Katharaopoulos et. al., we had pretrained model for MNIST. We test the model on random image generation and measured the time, image completion task and quality measurement and time difference.



MNIST Occluded MNIST



Output

Linear model was able to achieve faster and better image completion for given occluded input in comprasion to softmax based transformer

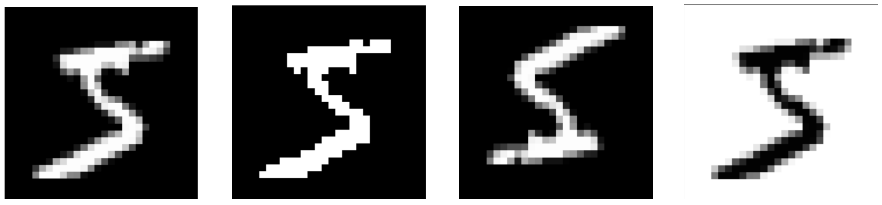
S.N	Number of Images generated	Linear	Softmax
1	100	33.45 s	367.91 s
2	200	67.46 s	811.99 s
3	300	76.46 s	1248.15 s

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

Due to the compute requirements of large-scale pre-training of Large Language Models (LLMs), fine-tuning is a widely adopted paradigm for adapting models pre-trained on general domain data to particular tasks or domains. However, the large number of parameters in such models can make even fine-tuning prohibitively expensive. Work from Li et al. [3] and Aghajanyan et al. [1] has demonstrated that models may be over-parameterized and may reside on a low intrinsic dimension. Taking inspiration 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. Note that r is not known *a priori* and one may have to try several small values of r.

2.1 Experiments

We train a base neural network on an image classification task on the MNIST dataset. 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.

2.1.1 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.

2.1.2 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%

2.1.3 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. 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	84.33%	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%

3 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.

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