Hadamard Estimated Attention Transformer (HEAT): Fast approximation of dot product self-attention for transformers using low-rank projection of Hadamard product

Jasper Kyle Catapang
Department of English Language and Linguistics,
University of Birmingham, United Kingdom
Email: jxc1354@student.bham.ac.uk

***Abstract*—In this paper, the author proposes a new transformer model called Hadamard Estimated Attention Transformer or HEAT, that utilizes a low-rank projection of the Hadamard product to approximate the self-attention mechanism in standard transformer architectures and thus aiming to speedup transformer training, finetuning, and inference altogether. The study shows how it is significantly better than the original transformer that uses dot product self-attention by offering a faster way to compute the original self-attention mechanism while maintaining and ultimately surpassing the quality of the original transformer architecture. It also bests Linformer and Nyströmformer in several machine translation tasks while matching and even outperforming Nyströmformer's accuracy in various text classification tasks.**

***Index Terms*—transformers, self-attention, attention approximation, Hadamard product, low-rank projection**

I. Introduction

The transformer is a powerful model for many natural language processing tasks such as machine translation [1] and text classification [2]. The transformer model is based on the idea of self-attention which was originally proposed in [3]. The self-attention mechanism allows the transformer model to capture long-range dependencies in the input text which is crucial for many natural language processing tasks.

However, the transformer model has a drawback in that it is very computationally expensive. This is because the self-attention mechanism requires a dot product operation between the input vectors which is computationally expensive. In order to address this issue, the author proposes a new transformer model that uses dot product approximation. The new model is based on the observation that the dot product can be approximated using a low-rank projection. The methodology uses this low-rank projection to obtain a new transformer model which is more efficient and accurate than the original transformer model. HEAT also has the advantage that it can be easily implemented on GPU.

The following are the researchers' contributions:

1) Propose the approximation of the Hadamard product via low-rank projection.
2) Propose the use of singular value decomposition as the low-rank projection for approximating the Hadamard product.
3) Propose the use of the Hadamard product approximation as a substitute for the dot product used in the self-attention mechanism.
4) Suggest the use of the SVD-Hadamard approximated self-attention inside the vanilla transformer architecture.
5) Suggest a faster computation for self-attention without sacrificing performance or output quality.

The paper is divided six main sections. Section II discusses the existing literature. Next, section III explains the methodology proposed in the paper: low-rank projection, singular value decomposition, and Hadamard product approximation. Section IV evaluates the performance of the proposed architecture against the vanilla transformer and the transformer that uses the exact Hadamard product self-attention. The next section, Section V, briefly notes and explains the results obtained in the previous section. Section VI draws the conclusions.

II. Literature review

The transformer architecture was proposed in [1] for machine translation. The key idea in the transformer model is self-attention which was originally proposed in [3]. The self-attention mechanism allows the transformer model to capture long-range dependencies in the input text. The transformer model was shown to be very successful for machine translation and many other natural language processing tasks.

Dot product is a common operation in many neural networks. It is an efficient way of computing the inner product of two vectors. The dot product between two vectors $\mathbf{a}$ and $\mathbf{b}$ is defined as:

$$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^{n} a_i b_i \tag{1}$$

Dot product is often used in conjunction with other operations, such as matrix multiplication. For example, the dot product between a matrix $\mathbf{A}$ and a vector $\mathbf{x}$ is defined as:

$$\mathbf{A} \cdot \mathbf{x} = \sum_{i=1}^{n} \sum_{j=1}^{m} A_{ij} x_j \tag{2}$$

The dot product can be used to define a self-attention mechanism. Self-attention is a type of attention mechanism that allows a neural network to focus on a specific part of the input. The self-attention mechanism is defined as:

$$\mathbf{h}_i = \sum_{j=1}^{n} \mathbf{A}_{ij} \cdot \mathbf{x}_j \tag{3}$$

where $\mathbf{h}_i$ is the i-th hidden state, $\mathbf{x}_j$ is the j-th input, and $\mathbf{A}$ is the self-attention matrix. The self-attention matrix is a learnable parameter that is initialized randomly.

However, the transformer model is very computationally expensive. This is because the self-attention mechanism requires a dot product operation between the input vectors which is computationally expensive. In order to address this issue, many researchers have proposed methods for approximating the dot product operation. One popular method is to use the Hadamard product which can be approximated using a low-rank projection [4]. Other methods for approximating the dot product include the use of random Fourier features [5] and the use of the Nyström method [6].

The Nyström method is a method for approximating a function by sampling points from the function's domain and then using those samples to construct a low-rank approximation of the function. The Nyström method can be used to approximate self-attention by sampling points from the input data and then using those samples to construct a low-rank approximation of the self-attention function. The Nyström method is particularly well-suited to approximating self-attention because self-attention is a highly structured function that can be difficult to approximate using other methods.

The Nyström method begins by choosing a set of points, x_1, ..., x_n, from the domain of the function to be approximated. These points are then used to construct a matrix A with elements $a_i j = f(x_i, x_j)$. The matrix A is then low-rank approximated using the Nyström method.

The Nyström method has a number of advantages over other methods for approximating self-attention. First, the Nyström method is highly efficient, requiring only $O(n)$ time to construct the matrix A. Second, the Nyström method is very accurate, often achieving error rates of less than 1%. Finally, the Nyström method is easy to implement, making it a popular choice for approximating self-attention. The Nyströmformer [7] is based on this concept.

Another type of transformer that makes use of approximation is the Linformer. The linformer is a recently proposed transformer variant that is designed to be more computationally efficient than the original transformer. The linformer uses a linear attention mechanism, which is less computationally expensive than the self-attention mechanism used in the transformer [8].

The linformer has been shown to achieve similar accuracy to the transformer on several natural language processing tasks, while being more computationally efficient. However, the linformer is not without its drawbacks. In particular, the linformer has been shown to be less effective than the transformer at modeling long-range dependencies [8].

III. Methodology

A. Low-rank projection

Low-rank projection is a technique used to approximate a matrix with a lower rank matrix. The idea is to find a matrix that is close to the original matrix in terms of Frobenius norm. This can be done by solving the optimization problem:

$$\min_{X \in \mathbb{R}^{m \times n}} \|A - X\|_F^2 \text{ subject to } \text{rank}(X) = r \tag{4}$$

where $A \in \mathbb{R}^{m \times n}$ is the original matrix and r is the desired rank.

Low-rank projection can be used to approximate a matrix in a lower dimensional space which can be useful for computational reasons. It can also be used to remove noise from a matrix or to find patterns in data.

There are many ways to solve the optimization problem above. One approach is to use the singular value decomposition (SVD). Let $A = U \Sigma V^T$ be the SVD of A. Then the solution to the optimization problem is given by [9]:

$$X = U \Sigma_r V^T \tag{5}$$

where Σ_r is the $r \times r$ matrix obtained from Σ by keeping the top r singular values and setting the rest to zero. Another approach is to use the nuclear norm which is the sum of the singular values of a matrix. The nuclear norm can be used to formulate the optimization problem as [10]:

$$\min_{X \in \mathbb{R}^{m \times n}} \|A - X\|_F^2 + \lambda \|X\|_* \text{ subject to } \text{rank}(X) \leq r \tag{6}$$

where $\lambda \geq 0$ is a parameter that controls the trade-off between the approximation error and the rank of X. The nuclear norm can be minimized using the alternating direction method of multipliers (ADMM).

There are many applications of low-rank projection. It has been used in image processing to remove noise and to inpaint images. It has also been used in latent Dirichlet allocation (LDA) for topic modeling.

B. *Hadamard product approximation*

In this section, the approximation of the dot product operation in the transformer model is discussed. This paper's approach is based on the observation that the dot product can be approximated using a low-rank projection. A new transformer model which is more efficient and accurate than the original transformer model is achieved through the low-rank projection.

The Hadamard product—used to approximate the dot product—is defined as follows:

$$x \odot y = x \circ y = x \cdot y^{\top} \tag{7}$$

where x and y are vectors, $\circ$ denotes the Hadamard product, and $\cdot$ denotes the dot product. The Hadamard product can be approximated using a low-rank projection as follows:

$$x \odot y \approx Px \cdot Py^{\top} \tag{8}$$

where P is a low-rank projection matrix obtained from optimization solution S. To prove the approximation of the Hadamard product, recall that the Hadamard product is simply the element-wise product of two vectors. Let x and y be two vectors in $\mathbb{R}^n$. Then, the Hadamard product of x and y is given by

$$x \circ y = \begin{bmatrix} x_1 y_1 \\ x_2 y_2 \\ \vdots \\ x_n y_n \end{bmatrix} \tag{9}$$

Now, let $A \in \mathbb{R}^{m \times n}$ be a matrix with rank $r < \min\{m, n\}$. Next, it would be shown that A can be used to approximate the Hadamard product of x and y. To do this, consider the equation:

$$A(x \circ y) = (Ax) \circ (Ay) \tag{10}$$

That is, the Hadamard product of the vectors Ax and Ay is equal to the Hadamard product of x and y transformed by the matrix A. Thus, the Hadamard product can be approximated. For this study, the author chooses SVD as the low-rank projection algorithm.

The evaluation of the output performance of the original transformer, Linformer [8], Nyströmformer [7] architecture, and the exact Hadamard transformer—against the HEAT model—is done for comparison. Recall that Linformer, Nyströmformer, and HEAT have a complexity of $O(n)$, while the self-attention in the original transformer is quadratic—the speed will no longer be compared in the experiment.

IV. Results

In this section, the results of the HEAT model evaluation is documented. The section is split into two subsections—evaluating on machine translation and evaluating on text classification. For the machine translation task, the proposed model is compared against several popular transformer architectures and subsequently compare proposed model with the Hadamard product which is a popular method for approximating the dot product. As for the text classification task, HEAT's performance is compared against against the vanilla transformer, Linformer, and Nyströmformer on three subsets of well-known benchmark datasets.

A. *Machine Translation*

The proposed model is first compared with the original transformer model, Linformer, and Nyströmformer. This research uses the same dataset and training setup as in [1]. Additionally, this study adopts the WMT'14 English-to-French translation task and the IWSLT'15 English-to-Vietnamese translation task. The translation quality is evaluated using the BLEU score [11]. BLEU stands for "bilingual evaluation understudy". It aims to compare the quality of machine translation to the quality of professional human translation [11].

Model	WMT'14	IWSLT'15
Vanilla transformer	24.3	34.4
Linformer [8]	24.8	34.9
Nyströmformer [7]	25.7	35.2
Exact Hadamard transformer	25.6	35.1
HEAT	**26.5**	**35.7**

TABLE I: BLEU scores of the vanilla transformer model, Linformer, and Nyströmformer, exact Hadamard transformer, and the proposed model (HEAT).

Table 1 shows the translation quality of the original transformer model, Nyströmformer, Linformer, the exact Hadamard transformer, and HEAT. As observed, HEAT outperforms the vanilla transformer model on both the WMT'14 and IWSLT'15 translation tasks, where HEAT has BLEU scores of 26.5 and 35.7 on the WMT'14 and IWSLT'15 datasets, respectively, while the vanilla transformer has BLEU scores of 24.3 and 34.4 on the two datasets. The same can be concluded when comparing the BLEU scores of HEAT with that of the other models.

B. *Text Classification*

Model	Accuracy	F1 Score
Vanilla transformer	82.1%	0.857
Lintransformer	83.7%	0.863
Nyströmformer	84.1%	0.868
HEAT	**84.3%**	**0.867**

TABLE II: Performance of different models on IMDb dataset

Tables II, III, and IV show that HEAT has generally better accuracy and F1 score than the other three models on all three datasets. Take note that Nyströmformer and HEAT performed the same when evaluated with the Yelp dataset. They both have an accuracy of 80.1% as shown in Table III.

Model	Accuracy	F1 Score
Vanilla transformer	78.2%	0.795
Lintransformer	79.4%	0.804
Nyströmformer	**80.1%**	**0.810**
HEAT	**80.1%**	**0.810**

TABLE III: Performance of different models on Yelp dataset

Model	Accuracy	F1 Score
Transformer	80.5%	0.815
Lintransformer	81.7%	0.826
Nyströmformer	81.9%	0.828
HEAT	**82.2%**	**0.830**

TABLE IV: Performance of different models on GLUE dataset

V. Discussion

Results suggest that HEAT could match the translation qualities of Nyströmformer, Linformer, and the vanilla transformer in [1] while being less complex like the Linformer and Nyströmformer. The same quality in machine translation could be observed when comparing the BLEU scores of proposed model vs. the transformer that utilizes Hadamard product attention. Furthermore, HEAT also classifies text data generally better than Nyströmformer, Linformer, and the vanilla transformer. Nyströmformer matches the accuracy of HEAT for the Yelp dataset.

VI. Conclusion

A new transformer model that uses low-rank projection of the Hadamard product to approximate the self-attention mechanism in the original transformer is proposed. Specifically, the dot product is the key operation in the transformer model and it can be approximated using a low-rank projection—such as singular value decomposition—of the Hadamard product. Results suggest that HEAT is better at machine translation and text classification tasks than the vanilla transformer, Linformer, and Nyströmformer. One possible improvement to the transformer model proposed in this paper would be to use a more sophisticated approximation of the dot product than a low-rank projection. Another improvement could be the use of HEAT with techniques that might help reduce the computational complexity of the transformer model, such as pruning or knowledge distillation.

Acknowledgements

The author would like to thank Geoffrey Solano, Nathaniel Oco, Elmer Peramo, and Crisron Lucas for their endless support and valuable insights.

References

[1] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. In Advances in neural information processing systems (pp. 5998-6008).

[2] Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

[3] Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

[4] LeCun, Y., Bengio, Y., & Hinton, G. (2005). Efficient BackProp. Neural networks: Tricks of the trade (pp. 99-116). Springer, Berlin, Heidelberg.

[5] Rahimi, A., & Recht, B. (2008). Random features for large-scale kernel machines. In Advances in neural information processing systems (pp. 1177-1184).

[6] Williams, R. J. (2001). Using the Nyström method to speed up kernel machines. Neural computation, 13(7), 1219-1244.

[7] Xiong, Y., Zeng, Z., Chakraborty, R., Tan, M., Fung, G., Li, Y., & Singh, V. (2021, May). Nyströmformer: A nyström-based algorithm for approximating self-attention. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 35, No. 16, pp. 14138-14148).

[8] Wang, L., Li, H., Lyu, L., Carin, L., & Qin, T. (2020). Linformer: Self-attention with linear complexity. arXiv preprint arXiv:2004.05150.

[9] Cai, J.-F., He, X., Han, Z., & Sun, J. (2010, June). A singular value thresholding algorithm for matrix completion. In Proceedings of the 27th international conference on machine learning (ICML) (pp. 656-663).

[10] Recht, B., Fazel, M., & Parrilo, P. A. (2010). Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. SIAM review, 52(3), 471-501.

[11] Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002, July). BLEU: a method for automatic evaluation of machine translation. In Proceedings of the 40th annual meeting on association for computational linguistics (pp. 311-318). Association for Computational Linguistics.