OPT2024: 16th Annual Workshop on Optimization for Machine Learning

Optimizing Attention

Hanno Ackermann
Hong Cai
Markus Nagel
Leyla Mirvakhabova
Farhad Zanjani
Fatih Porikli
Qualcomm AI Research, Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc.

Abstract

The attention mechanism is an important part of transformer architectures. It enables the network to compare samples within a sequence. Before the comparison is performed, tokens are multiplied by trainable matrices. These matrices can constitute a significant part of the total number of parameters. Their size creates problems on systems with limited cache in the compute unit, especially if there is limited bandwidth between compute unit and memory. In particular, GPUs on mobile devices suffer from this double bottleneck.

Prior works mitigate this problem for instance by storing low-rank approximations, quantization or minimizing the amount of data that needs to be transferred. In this paper, an alternative to the traditional attention mechanism is proposed which does not require any trainable matrices to perform the attention. The idea rests upon solving optimization problems, whereby memory is substituted for compute. It will be shown however, that the computational demand can be reduced such that auto-differentiation becomes possible. An experimental evaluation shows that the proposed algorithm performs favorable compared with several baselines.

1. Introduction

Transformers [12] have been widely used in various applications, such as natural language processing [5] and image recognition [6]. More recently, very large transformer models have been successfully applied to auto-regressive language modeling, such as GPT [10], Llama [11], and Gemini [1]. These powerful transformers require a vast amount of memory to run due to their large model sizes. While they can be loaded layer by layer during inference, it still poses a significant challenge when running them on mobile devices, which typically have very limited cache in the neural processing unit – often, a few megabytes only. In addition, the attention operation incurs quadratic complexity w.r.t. number of input tokens, both in terms of memory usage and computation.

Recently, researchers have proposed various techniques to reduce the computation and/or memory costs of attention. Efficient attention methods such as [8] only requires linear computation and memory costs while maintaining model accuracy. Other works optimize the implementation of attention. For instance, FlashAttention [3] uses tiling to decompose the softmax computation and other techniques to reduce the memory cost to linear. However, in these approaches, the large matrices mapping the input to query, key, and value tensors prior to attention are still there. This requires transferring (part of) these matrices between slow and fast memory units on hardware via limited IO bandwidth, which incurs high latency especially on resource-constrained devices.

In this work, we propose a novel, optimization-based approach to perform both self- and cross-attention, in order to eliminate both the heavy weight matrices and softmax-based attention computation. We formulate attention as an optimization problem. Instead of remixing the values based on the similarity/attention matrix between queries and keys, we directly solve for the mixing coefficients for the value tensor. Specifically, we impose that the remixed values needs to be similar to the queries. We additionally impose a sparsity constraint, which leads to a sparse reconstruction problem, ie. the well-known Lasso optimization task. To reduce the high cost solving these problems iteratively during any forward pass and to reduce the memory required for backward passes, we propose to use several solutions to increase efficiency. It is possible to use auto-differentiation with the proposed algorithm. To mitigate the errors induced by the proposed approximations, we also propose include random tokens in our algorithm, similar to the idea proposed by Darcet et al. [4], yet we offer a geometric intuition regarding the effect these tokens have in the proposed algorithm.

The main difference to standard attention is that here the transformer learns to generate similarities based on the scalar products between particular tokens that allow the subsequent heads to achieve low training losses. Conversely, the proposed algorithm requires the transformer to modify the tokens such that the token remixing induced by the Lasso yields results that eventually contribute to low training losses.

2. Standard Attention

Attention performs exhaustive pairwise comparisons between the set of query tokens, $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_{N_q}$, with the set of key tokens, $\boldsymbol{k}_1, \ldots, \boldsymbol{k}_{N_k}$. The resulting similarity matrix, i.e., attention matrix, is used to linearly combine a set of value tokens, $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{N_k}$.

In self-attention, the elements, $\boldsymbol{q}_i$, $\boldsymbol{k}_j$, $\boldsymbol{v}_j$, are created by multiplying the input by the weight matrices, $\boldsymbol{W}_q$, $\boldsymbol{W}_k$, and $\boldsymbol{W}_v$, which correspond to the queries, keys, and values, respectively. The weight matrices have sizes $D \times D_{q,k,v}$. Here, the three numbers $D_{q,k,v}$ indicate the dimensions of the three token sequences $\boldsymbol{q}_i$, $\boldsymbol{k}_j$, and $\boldsymbol{v}_j$.

In cross-attention, two different input sequences are used. The queries $\boldsymbol{q}_i$ are created from one sequence by multiplication with $\boldsymbol{W}_q$, whereas the keys $\boldsymbol{k}_j$ and values $\boldsymbol{v}_j$ are created from another sequence by multiplications with $\boldsymbol{W}_k$ and $\boldsymbol{W}_v$, respectively. Some models such as the one proposed by [2] even use three different input sequences.

For multi-head attention, the vectors $\boldsymbol{q}_i$, $\boldsymbol{k}_j$ and $\boldsymbol{v}_j$ are split into parts of equal size, for instance, $\boldsymbol{q}_i = \begin{bmatrix} \boldsymbol{q}_{i1}^T & \boldsymbol{q}_{i2}^T & \cdots & \boldsymbol{q}_{N_H}^T \end{bmatrix}$, where N_H indicates the number of heads. The dimensions of $\boldsymbol{q}_i$, $\boldsymbol{k}_j$ and $\boldsymbol{v}_j$, D_q and D_v must be integer-divisible by N_H.

Stacking the tokens into matrices $\boldsymbol{Q}^T = \begin{bmatrix} \boldsymbol{q}_1 & \cdots \end{bmatrix}$, $\boldsymbol{K}^T = \begin{bmatrix} \boldsymbol{k}_1 & \cdots \end{bmatrix}$ and $\boldsymbol{V}^T = \begin{bmatrix} \boldsymbol{v}_1 & \cdots \end{bmatrix}$, the attention mechanism proposed by [12] is defined by $\hat{\boldsymbol{Q}} = \text{soft-argmax}\left(\eta \boldsymbol{Q}\boldsymbol{K}^T\right)\boldsymbol{V}$, where $\eta = D^{-1/2}$. The linear combinations of the rows of $\boldsymbol{V}$ yields the rows $\hat{\boldsymbol{q}}^T$ of matrix $\hat{\boldsymbol{Q}}^T$.

3. Proposed Approach: Optimizing Attention

The original transformer [12] can be formulated as a message-passing algorithm, as shown by [13, 14]. We draw on this formulation and use a simplified intuition in the following: At each attention, the nodes in the *value* set $\{v_j\}$ send update messages $\boldsymbol{v}_j$ to the nodes of the *query* set $\{q_i\}$. The incoming messages into each node of the query set are weighted by the normalized scalar products $\text{soft-argmax}(\eta \boldsymbol{q}_i^T \boldsymbol{K}^T)$. These messages are sum-aggregated before the nodes in the query set are updated by an MLP. Normalization by $\mathrm{soft\text{-}argmax}$ maintains the original numerical ranges for the sum-aggregated update messages.

First, we formulate attention as a reconstruction problem, where we solve for the linear transformation that linearly combines the values so as to approximate the queries

$$\min_{\boldsymbol{x}_i} \left\| \boldsymbol{q}_i^T - \boldsymbol{x}_i^T \cdot \boldsymbol{V} \right\| \tag{1}$$

for a suitable norm, where $\boldsymbol{x}_i$ is an N_v-dimensional variable to be optimized. The scalars $\boldsymbol{x}_i$ control the linear combination and subsume the role of the attention coefficients $\mathrm{soft\text{-}argmax}\left(\eta\, \boldsymbol{Q}\boldsymbol{K}^T\right)$.

In the case of cross-attention, the query and value tensors are the two inputs, respectively, without being transformed by the $\boldsymbol{W}_q$ and $\boldsymbol{W}_v$ weight matrices. By solving the optimization, we project the values to the subspace of the queries if the l_2-norm is being used; this resembles what the original cross-attention does. In self-attention, both query and value tensors are from the same input, so we require the ith element of $\boldsymbol{x}_i$ to be zero to avoid a trivial solution. In this way, the optimization reveals the correlation structure between every input element and the rest via this reconstruction.

Next, we impose the requirement that only few nodes in the value set should be allowed to send messages to a particular node q_i, which helps mitigate overfitting. Using an l_1 regularization, we arrive at the sparse reconstruction problem

$$\min_{\boldsymbol{x}_i} \left\| \boldsymbol{q}_i^T - \boldsymbol{x}_i^T \cdot \boldsymbol{V} \right\|_2^2 + \lambda \left\| \boldsymbol{x}_i \right\|_1 , \tag{2}$$

where the scalar λ controls the sparsity. We can solve Eq. (2) by means of the alternating direction method of multipliers (ADMM). Defining auxiliary variables $\boldsymbol{z}$, $\boldsymbol{\mu}$ and a scalar ρ, the ADMM can be optimized by iterating the following steps

$$\boldsymbol{x}^{(k+1)} = \left(\boldsymbol{V}\boldsymbol{V}^T + \rho \boldsymbol{I}\right)^{-1} \left(\boldsymbol{V}\boldsymbol{q}_i + \rho \left(\boldsymbol{x}^{(k)} - \boldsymbol{z}^{(k)}\right)\right) \tag{3a}$$

$$\boldsymbol{z}^{(k+1)} = \tau_{\lambda/\rho} \left(\boldsymbol{x}^{(k)} - \boldsymbol{\mu}^{(k)}\right) \tag{3b}$$

$$\boldsymbol{\mu}^{(k+1)} = \boldsymbol{\mu}^{(k)} + \boldsymbol{x}^{(k+1)} - \boldsymbol{z}^{(k+1)} \tag{3c}$$

where $\tau_a(\cdot)$ in Eq. (3b) denotes the proximal operator, the superscript (k) the iteration number and $\boldsymbol{I}$ the identity matrix. Since the token $\boldsymbol{q}_i$ is contained in one of the rows of $\boldsymbol{V}$ in case of self-attention, we zero-out the corresponding entry of $\boldsymbol{x}_i$ at each iteration. While the original convergence guarantee is not longer applicable, we notice that clamping the entries of $\boldsymbol{x}_i$ prevents divergence.

Naïvely inverting $\boldsymbol{V}\boldsymbol{V}^T + \rho\boldsymbol{I}$ in Eq. (3a) is computationally expensive, since it is quadratic in the number of tokens in the value set and can be very large. Yet, we notice that $\boldsymbol{V}$ is extremely narrow; in fact, we usually have $N_v \gg D_v/N_H$. This implies that the eigenvalue decomposition $\boldsymbol{V}^T\boldsymbol{V} = \boldsymbol{X}\boldsymbol{D}\boldsymbol{X}^T$ can be efficiently computed. Since the non-zero eigenvalues of $\boldsymbol{V}\boldsymbol{V}^T$ equal those of $\boldsymbol{V}^T\boldsymbol{V}$, we only need the left singular vectors $\boldsymbol{Y} \approx \boldsymbol{V}\boldsymbol{X}\boldsymbol{D}^{-1/2}$ of $\boldsymbol{V} = \boldsymbol{Y}\boldsymbol{S}\boldsymbol{X}^T$[1] since they equal the eigenvectors of $\boldsymbol{V}\boldsymbol{V}^T$ corresponding to the non-zero eigenvalues. Thereby, we can efficiently compute the updates

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{Y}\boldsymbol{S}^{-1} \left(\boldsymbol{Y}^T \left(\boldsymbol{V}\boldsymbol{q}_i + \rho \left(\boldsymbol{x}^{(k)} - \boldsymbol{z}^{(k)}\right)\right)\right) \tag{4}$$

1. Since it is standard to denote the value matrix by $\boldsymbol{V}$, we are using $\boldsymbol{V} = \boldsymbol{Y}\boldsymbol{S}\boldsymbol{X}^T$ for the singular value decomposition.

Model	AP	AP50	AP75	AP_S	AP_M	AP_L
DETR original	40.6	61.6	-	19.9	44.3	60.2
bl no weights SA dec	38.5	58.0	40.3	17.5	41.3	58.3
bl no weights CA dec	36.0	57.7	37.2	15.1	38.4	55.6
prop decoder SA	38.5	57.9	40.1	18.8	41.5	57.4
prop decoder CA	37.7	59.4	39.0	16.5	40.4	57.4

Table 1: Comparison between the two baseline models *bl no weights SA dec* and *bl no weights CA dec* (middel two rows) with the proposed algorithm (bottom two rows). The bottom rows must be compared with *bl no weights CA dec*, while the second row from the bottom needs to be compared with *bl no weights SA dec*. It can be seen that the proposed algorithm achieves superior results. AP indicates the average precision metric used in object detection.

without ever having to allocate memory to the large inverse. We augment the matrix $\boldsymbol{S}$ by adding ρ to the diagonal of $\boldsymbol{S} = \boldsymbol{D}^{1/2}$. To avoid exploding gradients during backpropagation, we add $\boldsymbol{r} \cdot \boldsymbol{I}$ to the diagonal of $\boldsymbol{V}^T\boldsymbol{V}$ where the entries of $\boldsymbol{r}$ are drawn from a uniform distribution $\mathcal{U}(0, \sigma)$ with σ being small.

We notice that after some training, the matrix of values tokens $\boldsymbol{V}$ degenerates, i.e., many of the eigenvalues of $\boldsymbol{V}^T\boldsymbol{V}$ become very small. This necessitates more iterations, thus slowing down forward and backward passes through the network and increasing memory demand. Furthermore, this rank-deficiency prevents ADMM from reaching a reasonable solution if the query $\boldsymbol{q}_i$ is far from the range space.

Instead of reverting to the slow and memory-intensive formulation in Eq. (3a), we propose to create a set of tokens whose entries are drawn from $\mathcal{N}(0, 1)$ and are appended to $\boldsymbol{V}$. First, this increases the eigenvalues of $\boldsymbol{V}^T\boldsymbol{V}$. Secondly, it endows the left singular vectors $\boldsymbol{Y}$ with components which span parts of the nullspace, thereby enabling the algorithm to regress components of $\boldsymbol{q}_i$ in the kernel space. The idea of using additional tokens is similar to the idea proposed by Darcet et al. [4] except that the tokens used here are not trainable. Furthermore, they serve a specific purpose interpretable in terms of linear algebra. Lastly, to avoid the computationally expensive soft-argmax operator, we raise each entry of the final $\boldsymbol{x}_i$ to its 5th power and normalize the resulting vector by dividing by the sum of its absolute values.

4. Experiments

We evaluate the impact of the proposed optimized attention by comparing with a transformer model that uses both self- and cross-attention. To this end, we compare with the model (*DETR*) proposed by Carion et al. [2] which comprises six transformer layers with self-attention, followed by another six layers with both self- and cross-attention. We compare against several baselines that omit weight matrices in their attention mechanisms. We follow the procedure in [2] and perform experiments on the COCO 2017 detection dataset [9]. We do all experiments using 4 V100 GPUs and a batch size of 8 for each GPU. We use the original learning rate, weight decay and learning rate scheduling.

4.1. Quantitative Evaluation

Table 1 show results of the proposed method with several baselines. On top are the results provided in the paper after training for 300 epochs on 16 V100s with a batch size of 4 per GPU. The metrics used for the comparisons in table 1 are the average precisions popular in object detection which are based on precision and recall [7]. The baseline models we use for comparison are indicated

Model	AP	AP50	AP75	AP_S	AP_M	AP_L
l_2-optim (CA)	28.0	49.3	27.4	7.8	28.4	48.8
vanilla ADMM (SA)	38.0	57.3	39.8	17.3	41.2	57.2
efficient ADMM (SA)	38.5	57.9	40.1	18.8	41.5	57.4

Table 2: Ablation study: The top row shows results using an l_2-norm reconstruction objective, the following row results of using a slow vanilla ADMM (Eq. (3)) and the bottom row the proposed efficient ADMM.

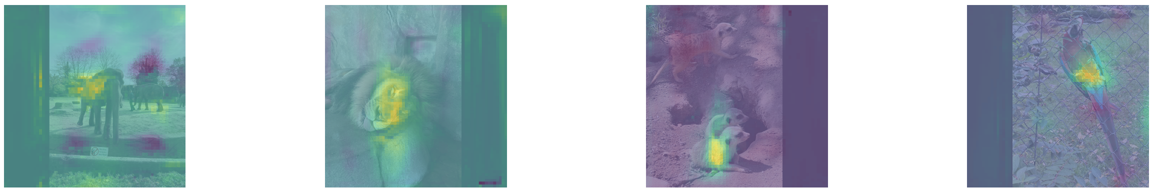

Figure 1: Qualitative examples of the image that the decoder tokens attend to. Images are shown with padding. It can be seen that the attentions focuses at small image spots.

by *"bl no weights SA dec"* and *"bl no weights CA dec"*. They do not use weights for their self-attentions or cross-attentions, respectively, in the decoder. The last two rows show the results using the proposed attention; the second last row for the self-attention of the decoder, the last row for the cross-attention. Here, the networks with modified cross-attention need to be compared, and the two networks with modified self-attention. As can be seen, the proposed algorithm achieves results superior to the corresponding baselines.

We provide several ablations in table 2. The first row shows the result optimizing the l_2-norm in the cross-attention of the decoder. The other two rows correspond to a vanilla ADMM and the proposed efficient ADMM.

4.2. Qualitative Results

We provide qualitative examples of the results of the proposed algorithm in Fig. 1. They were created by a network in which the cross-attention in the decoder is replaced. The attention maps stem from one of the 100 tokens in the decoder that cross-attend to the image maps and are subsequently decoded into object location and classifications. The images used here are not from MSCOCO. It can be seen that the tokens focus their attention onto small spots of the images. It can also be seen that a small part of the attention draws information from the padded regions, see Darcet et al. [4].

5. Discussion

Standard transformers require tokens to be multiplied by weight matrices for their attention mechanism. These weights can be so large that they do not fit into the fast memory of the compute units. In particular on mobile GPUs, this memory is so limited, that only a few tokens can be multiplied with a small piece of each weight matrix, thereby performing the matrix-vector products in tiny segments. The required data transfer exceeds the limited bandwidth between the slow main memory and the fast compute devices, hence the total latency increases. While other works compress the data, we propose to eliminate the matrices from the attention algorithms. We show that attention can be performed by solving optimization problems. To increase the speed of those operations and allow for auto-differentiation, we propose several techniques to reduce the required memory. The experimental evaluation shows that our algorithms compare favorably with the baselines.

References

[1] Guangsheng Bao, Zebin Ou, and Yue Zhang. Gemini: Controlling the sentence-level summary style in abstractive text summarization. In *Conference on Empirical Methods in Natural Language Processing*, 2023.

[2] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision (ECCV)*, 2020.

[3] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[4] Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. In *International Conference on Learning Representations*, 2024.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, 2019.

[6] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021.

[7] Paul Henderson and Vittorio Ferrari. End-to-end training of object class detectors for mean average precision. In *arXiv*, 2017.

[8] Pierre-David Letourneau, Manish Kumar Singh, Hsin-Pai Cheng, Yunxiao Shi, Dalton Jones, Matthew Harper Langston, Shizhong Han, Hong Cai, and Fatih Porikli. Padre: A unifying polynomial attention drop-in replacement for efficient vision transformer. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[9] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision (ECCV)*, 2014.

[10] OpenAI, Josh Achiam, et al. GPT-4 technical report. Technical report, OpenAI, 2023.

[11] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. Technical report, MetaAI, 2023.

[12] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[13] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations (ICLR)*, 2018.

[14] Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim. Graph transformer networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.