The Inhibitor: ReLU and Addition–Based Attention for Efficient Transformers

Preprint · October 2023

DOI: 10.13140/RG.2.2.23646.20805

1 author:

Rickard Brännvall
RISE Research Institutes of Sweden
31 PUBLICATIONS **188** CITATIONS

SEE PROFILE

The Inhibitor: ReLU and Addition-Based Attention for Efficient Transformers

Rickard Brännvall[1,2]
[1]Computer Science Department, RISE Research Institutes of Sweden
[2]Machine Learning Group, Luleå University of Technology, Sweden
rickard.brannvall@ri.se

September 2023

Abstract

To enhance the computational efficiency of quantized Transformers, we replace the dot-product and Softmax-based attention with an alternative mechanism involving addition and ReLU activation only. This side-steps the expansion to double precision often required by matrix multiplication and avoids costly Softmax evaluations but maintains much of the core functionality of conventional dot-product attention. It can enable more efficient execution and support larger quantized Transformer models on resource-constrained hardware or alternative arithmetic systems like homomorphic encryption. Training experiments on four common benchmark tasks show test set prediction scores comparable to those of conventional Transformers with dot-product attention. Our scaling experiments also suggest significant computational savings, both in plaintext and under encryption. In particular, we believe that the ReLU and addition-based attention mechanism introduced in this paper may enable privacy-preserving AI applications operating under homomorphic encryption by avoiding the costly multiplication of encrypted variables.

1 Introduction

Transformer models [1] have achieved great success in various machine-learning tasks and are the foundation for modern large-scale pre-trained Language Models such as BERT [2], RoBERTa [3], XLNet [4], Transformer-XL [5], and the GPT family [6, 7]. These models have enabled leaps in performance on many NLP tasks, as for example experienced by the millions that use the ChatGPT application [8]. Similarly, the Vision-Transformer (ViT) [9], and similar models outperform CNNs on many computer vision tasks such as image classification [10] and object detection [11].

The impressive gains in performance, however, come at a price. State-of-the-art Transformers are extremely large, having parameters in the hundreds

of billions. The requirements for memory and computational power have become prohibitive, not only for deployment on resource-constrained embedded systems, but sometimes even in data centers, and their considerable energy usage is widely discussed in society. Neural network quantization [12, 13, 14] is a method to address this problem, as it reduces memory usage by employing lower-bit precision for weight and activation tensors. This not only decreases inference time but also increases energy efficiency through the adoption of low-bit fixed-point arithmetic instead of floating-point operations [15]. Other approaches to improve Transformer's scalability and environmental sustainability include Pruning [16, 17], Knowledge Distillation [18, 19] and Neural Architecture Search [20]. In this work, we propose changes to the architecture's fundamental building blocks to take steps towards the same end.

Transformers use an attention mechanism to identify connections between different elements in a sequence, which conventionally takes the dot-product between the *query* and *key* matrices before passing the results through a Softmax function. Many alternative formulations have been proposed, such as the Fastformer [21] based on additive attention, or the ReLUformer [22] that uses ReLU activation in place of Softmax. These architectures share elements with what is proposed in this work, however, they still apply attention by a matrix multiplication with the *value* matrix. Extending our earlier work with ReLU and addition-based gated RNNs [23], we here propose the Inhibitor attention mechanism, which is designed to not rely on variable-to-variable multiplication and Softmax activation. These operations are more costly than constant-to-variable multiplications (also known as literal multiplication) and ReLU on many computing hardware. Even more so, they are particularly challenging operations under homomorphic encryption, an arithmetic system that permits calculation with encrypted variables without access to the secret key [24].

2 Method

The Transformer architecture consists of repeated blocks of self-attention and feed-forward networks (FFNs). In the simple decoder application, each block takes as input a sequence embedding, $X \in \mathbb{R}^{n \times d}$, where n is the length and d is the dimension. The embedding is transformed into *query*, *key* and *value*, by multiplication with weight matrices according to $Q = XW_Q$, $K = XW_K$, and $V = XW_V$. *Attention scores* are calculated by passing the scaled dot-product of *query* and *key* through the Softmax function,

$$S = \text{Softmax}\left(QK^T/\sqrt{d}\right) \tag{1}$$

which is then used to form a weighted sum of *values* as output, $H = SV$, exploiting that each row of S is normalized to sum to one. The FFN has two fully connected layers with weight matrix multiplication and ReLU activation,

$$H = (XW_1^T + b_1)^+ W_2 + b_2 \tag{2}$$

where W_1, W_2 are weight matrices and b_1, b_2 are bias vectors. We use the shorthand notation $x^+ = \max(0, x)$ for the ReLU function. Each Transformer block typically also has one or more layer normalizations [25] that stabilize the gradient flow and contribute to regularizing the training.

The Inhibitor. We propose to alternatively calculate *attention scores* as

$$Z_{ij} = \sum_k \frac{1}{\gamma} |Q_{ik} - K_{jk}| \tag{3}$$

where mixing between *query* and *key* is based on their absolute difference instead of dot-product. We have thus replaced the cosine (dot-prod) distance of conventional attention with the Manhattan distance. In place of Softmax multiplication, we subtract *attention scores* from *values* inside a ReLu function,

$$H_{ik} = \sum_j (V_{jk} - Z_{ij})^+ \tag{4}$$

such that entries of V for which Z are large are zeroed out and do not contribute to the sum. We call this mechanism *inhibition* as it is reminiscent of subtractive inhibition in biological neurons. FFN and normalization are left unchanged.

Shifted inhibition score. The Manhattan score of equation 3 can only take the value zero for rows where Q and V are identical. To make it easier to obtain a zero inhibition score, which allows entries of the *value* matrix, V, to pass unmodified through equation 4 we test using a slightly modified inhibition score, $Z' = (Z - \alpha)^+$, which applies a constant shift $\alpha \geq 0$ to the Manhattan score.

3 Results

Benchmark comparisons. We carried out numerical experiments that trained Transformer models based on the new Inhibitor and the conventional dot-product attention on four standard tasks. The aim was not to achieve SotA results but rather to examine if the Inhibitor mechanism would perform comparably on a set of familiar tasks, which is why we used simple set-ups without hyperparameter tuning. From Table 1, which reports the results, we note that for each task, the two alternative attention mechanisms score very similarly. Indeed, none of the differences are significant at 95% confidence (over at least 20 repetitions for each experiment). For experiments with the Inhibitor, we used a shifted score with $\alpha = 0.5$ and scaling factor $\gamma = \sqrt{d}$.

The **adding** problem was introduced by [26] to test long-term memory for RNNs. It takes two inputs: a sequence of random numbers and a two-hot sequence (which, in our experiments, each is 100 long). The ground truth is simply the dot-product of the two inputs as vectors, which, rather obviously, is not a challenging task for a conventional (dot-prod-based) Transformer. It is encouraging that the addition-based Inhibitor performs just as well on this task (which is hard for, e.g., simple RNNs to solve).

	Adding	MNIST	IMDB	IAMW
Dot-prod Attention	0.11%	98.2%	87.2%	17.9
Inhibitor Attention	0.12%	97.9%	87.3%	18.1

Table 1: The Inhibitor shows comparable performance to conventional attention for Transformers trained on four standard tasks (for mse, acc, acc, and edit distance, respectively).

We also train one-layer Transformers for the **MNIST** handwritten digit recognition task [27] and **IMDB** movie-review sentiment analysis task [28], which are simple go-to benchmark task for image classification and text analysis, respectively. Although far from SotA, we note that our simple models achieve decent accuracy for both attention mechanisms, where the differences in performance are not significant.

The final task, labeled **IAMW** in Table 1, uses the IAM Handwriting Database [29], which is a collection of handwritten words written by more than 700 writers. The model first applies two convolutional layers followed by a Transformer and uses Connectionist Temporal Classification (CTC) loss [30] as an endpoint layer to predict the text. Edit distance [31] was used as the evaluation metric. Again, the differences are small.

Scaling experiments. Next, we wanted to examine and compare the scaling properties for the proposed mechanism under two identified scenarios: i) quantization with integer arithmetics and ii) homomorphic encryption. Therefore, the two alternative attention mechanisms were implemented directly in low-level code rather than high-level ML libraries, where built-in optimizations for conventional models and design choices would bias a comparison.

For the plaintext experiment, we used integer 16-bit arithmetics implemented in the Rust programming language, which gives detailed low-level control over circuits and supports advanced time benchmarking through the Criterion package. The encrypted implementation uses the TFHE scheme [32] for Fully Homomorphic Encryption on the Torus as supported in the Concrete library for Python [33]. TFHE permits the evaluation of arbitrary single-variate activation functions under homomorphic encryption through the Programmable Bootstrap [34]. We here used considerably smaller networks where the embedding dimension was limited to size 2. The encrypted circuits were implemented for integers with up to 8-bit precision.

The results indicate that the proposed Inhibitor mechanism can have a significant advantage, with i) 30%–50% saving for the plaintext implementation on CPU as per Table 2, and ii) a factor 3–6 under encryption with TFHE as per table 3. Timing estimates are averaged over 20 repeated experiments and are significant at the 95%

	32	64	128	256
Dot-prod Attention	98.6 µs	330 µs	1.2 ms	4.48 ms
Inhibitor Attention	63.1 µs	178 µs	577 µs	2.5 ms

Table 2: Estimated plaintext execution time on CPU for four different sequence lengths (with fixed size single head).

	2	4	8	16
Dot-prod Attention	2.68 s	22.4 s	107 s	828 s
Inhibitor Attention	0.749 s	8.56 s	23.8 s	127 s

Table 3: Estimated encrypted execution time on CPU for four different sequence lengths (with fixed size single head).

Conclusions. The proposed Inhibitor mechanism allows straightforward quantization as equations 3 and 4 naturally support integer implementation. Our scaling experiments indicate that it has considerable potential for computational savings both for Transformers processing plaintext data as well as ciphertext.

While experiment results are promising on simple training tasks, *for future work*, it is necessary to examine performance under more challenging settings, for example, by pre-training a much larger Transformer model and testing on modern NLP and image recognition benchmark tasks.

Appendix

3.1 Signed Inhibitor

The inhibition mechanism described in equation 4 can only produce a non-negative attended value matrix H, while the conventional dot-product attention can also select from V that can have both positive and negative entries. We can rewrite inhibition to handle signed values simply as

$$H_{ik} = \sum_j \left(V_{jk}^+ - Z_{ij}\right)^+ + \sum_j \left(V_{jk}^- + Z_{ij}\right)^-, \tag{5}$$

such that the ReLU terms within the sum will attenuate or extinguish V_{jk} for large Z_{ij}, while V_{jk} will pass through unaltered for $Z_{ij} = 0$.

3.2 Implemention

A naive implementation of equation 3 and 4 relies on expanding the dimension of the broadcasted differences before passing the result through an abs or ReLU function and then summing over an inner dimension. This uses large amounts of memory and should be avoided.

The ML software that we used for this study supports fused operations for certain operations that avoid memory bloat and make the execution of the inhibition much more efficient on CPU and GPU hardware. Specifically, it supports calculating the pairwise distance between two tensors[1], which directly implements calculating the Manhattan distance of equation 3.

For the inhibition step of equation 4, we use a simple trick based on

$$x^+ = \frac{x + |x|}{2} \tag{6}$$

and write

$$\begin{aligned} H_{ik} &= \sum_j (V_{jk} - Z_{ij})^+ \\ &= \frac{1}{2}\sum_j V_{jk} - \frac{1}{2}\sum_j Z_{ij} + \frac{1}{2}\sum_j |V_{jk} - Z_{ij}| \end{aligned} \tag{7}$$

where the first two terms are simple sums that do not expand memory usage, and the last term can benefit from efficient fused pairwise Manhattan distance implementations.

Similarly, we rewrite for the signed inhibition of equation 5 to have

$$\begin{aligned} H_{ik} &= \sum_j \left(V^+_{jk} - Z_{ij}\right)^+ + \sum_j \left(V^-_{jk} + Z_{ij}\right)^- \\ &= \frac{1}{2}\sum_j V_{jk} + \frac{1}{2}\sum_j |V^+_{jk} - Z_{ij}| - \frac{1}{2}\sum_j |V^-_{jk} + Z_{ij}|, \end{aligned} \tag{8}$$

where we have, in addition to equation 6, used the relation

$$x^- = \frac{x - |x|}{2} \tag{9}$$

for the negative ReLU function to leverage a fused implementation.

References

[1] Ashish Vaswani et al. Attention is all you need. In *Advances in NeurIPS*, volume 30. Curran Associates, Inc., 2017.

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

[3] Yinhan Liu et al. Roberta: A robustly optimized bert pretraining approach. 07 2019.

[1] The `cdist` function in the PyTorch library for deep learning.

[4] Zhilin Yang et al. Xlnet: Generalized autoregressive pretraining for language understanding. 06 2019.

[5] Zihang Dai et al. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, Florence, Italy, July 2019. Association for Computational Linguistics.

[6] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[7] Tom Brown et al. Language models are few-shot learners. 05 2020.

[8] John Schulman et al. Chatgpt: Optimizing language models for dialogue. *OpenAI blog*, 2022.

[9] Alexey Dosovitskiy and other. An image is worth 16x16 words: Transformers for image recognition at scale. 10 2020.

[10] Ze Liu et al. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 03 2021.

[11] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. *End-to-End Object Detection with Transformers*, pages 213–229. 11 2020.

[12] Shuchang Zhou et al. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. 06 2016.

[13] Benoit Jacob et al. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, jun 2018.

[14] Raghuraman Krishnamoorthi. Quantizing deep convolutional networks for efficient inference: A whitepaper. 06 2018.

[15] Mark Horowitz. 1.1 computing's energy problem (and what we can do about it). volume 57, pages 10–14, 02 2014.

[16] Song Han et al. Learning both weights and connections for efficient neural networks. 06 2015.

[17] Song Han, Huizi Mao, and William Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. 10 2016.

[18] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. 03 2015.

[19] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. 10 2019.

[20] Minghao Chen, Houwen Peng, Jianlong Fu, and Haibin Ling. AutoFormer: Searching transformers for visual recognition. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, oct 2021.

[21] Chuhan Wu, Fangzhao Wu, Tao Qi, Yongfeng Huang, and Xing Xie. Fastformer: Additive attention can be all you need, 2021.

[22] Kai Shen, Junliang Guo, Xu Tan, Siliang Tang, Rui Wang, and Jiang Bian. A study on relu and softmax in transformer, 2023.

[23] Rickard Brännvall, Henrik Forsgren, Fredrik Sandin, and Marcus Liwicki. Relu and addition-based gated rnn, 2023.

[24] Craig Gentry. Computing arbitrary functions of encrypted data. *Communications of the ACM*, 53(3):97–105, mar 2010.

[25] Jimmy Ba, Jamie Kiros, and Geoffrey Hinton. Layer normalization. 07 2016.

[26] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, nov 1997.

[27] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010. http://yann.lecun.com/exdb/mnist/.

[28] Andrew L. Maas et al. Learning word vectors for sentiment analysis. In *Proc. ACL 2023*, Portland, Oregon, USA, June 2011.

[29] Urs-Viktor Marti and H. Bunke. The iam-database: An english sentence database for offline handwriting recognition. *IJDAR*, 5:39–46, 11 2002.

[30] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification. In *Proc. ICML 2006*, New York, NY, USA, 2006. ACM.

[31] Gonzalo Navarro. A guided tour to approximate string matching. *ACM Comput. Surv.*, 33(1):31–88, mar 2001.

[32] Ilaria Chillotti et al. TFHE: Fast fully homomorphic encryption over the torus. *Journal of Cryptology*, 33(1):34–91, apr 2019.

[33] Zama. Concrete: TFHE Compiler for Python programs, 2022. `https://github.com/zama-ai/concrete`.

[34] I. Chillotti et al. Programmable bootstrapping enables efficient homomorphic inference of deep neural networks. 2015.