The Inhibitor: ReLU and Addition-Based Attention for Efficient Transformers (Student Abstract)

Rickard Brännvall[1,2]

[1]Computer Science Department, RISE Research Institutes of Sweden
[2]Machine Learning Group, Luleå University of Technology, Sweden

Background

Transformer models [1] have achieved great success in various ML tasks and are the foundation for modern large-scale language models. The architecture consists of repeated blocks of self-attention and feed-forward networks (FFNs).

In the simple decoder application, each self-attention block can consist of multiple attention heads. Each 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 the latent states *query*, *key* and *value*,

$$Q = XW_Q \tag{1}$$
$$K = XW_K \tag{2}$$
$$V = XW_V \tag{3}$$

by multiplication with weight matrices that are specific for each head.

Attention scores are calculated by passing the scaled dot-product of *query* and *key*,

$$Z_{ij} = QK^T/\sqrt{d} \tag{4}$$

through the Softmax function,

$$S_{ij} = \frac{\exp(Z_{ij})}{\sum_j \exp(Z_{ij})} \tag{5}$$

which is then used to form a weighted sum of *values* as output,

$$H = SV \tag{6}$$

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. Each Transformer block typically also has one or more layer-normalizations that stabilize the gradient flow.

Method

We propose to alternatively calculate *attention scores* as

$$Z_{ij} = \sum_k \frac{1}{\sqrt{d}} \left|Q_{ik} - K_{jk}\right| - \alpha \tag{7}$$

where α is a constant shift parameter. Now 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 a shifted Manhattan distance. In place of softmax multiplication, we subtract *attention scores* from *values* inside a ReLu function,

$$H_{ik} = \sum_j \left(V_{jk} - Z_{ij}^+\right)^+ \tag{8}$$

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.

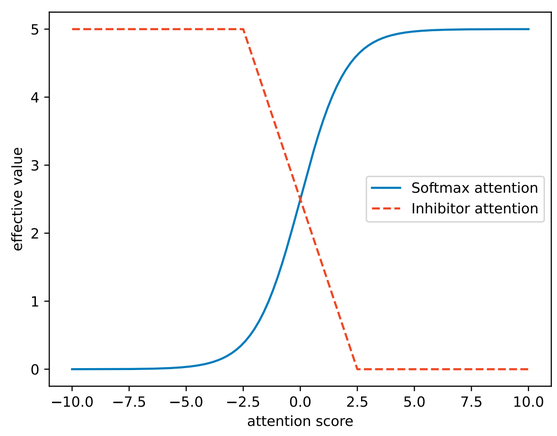

Figure 1: Caption for figure one.

Results

Benchmark comparisons. We carried out numerical experiments that trained Transformer models based on the Inhibitor and the conventional dot-product attention on four standard tasks. The **adding** problem was introduced by [2] to test long-term memory for RNNs. We also train one-layer Transformers for the **MNIST** handwritten digit recognition task [3] and **IMDB** movie-review sentiment analysis task [4], which are simple go-to benchmark task for image classification and text analysis, respectively. The final task, labeled **IAMW** in Table 1, uses the IAM Handwriting Database [5], which is a collection of handwritten words written by more than 700 writers.

	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). Differences are not significant at 95% confidence.

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. We used a shifted Inhibitor with $\alpha = 0.5$.

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.

	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).

For the plaintext experiment, we used integer 16-bit arithmetics implemented in the Rust programming language.The encrypted implementation uses the TFHE scheme as supported in the Concrete library for Python [6]. 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% confidence level.

Discussion

Many alternative formulations have been proposed, such as the Fastformer [7] based on additive attention, or the ReLUformer [8] that uses ReLU activation in place of Softmax. These architectures share elements with what is proposed in this note; however, they still apply attention by a matrix multiplication with the *value* matrix. Extending our earlier work with ReLU and addition-based gated RNNs [9], we here propose the Inhibitor attention mechanism, which is designed to not rely on variable-to-variable multiplication and Softmax activation.

The proposed Inhibitor mechanism allows straightforward quantization as equations 7 and 8 naturally support integer implementation. While experiment results are promising on simple training tasks, **for future work**, we examine performance under more challenging settings, for example, by pre-training a much larger Transformer model and testing on modern NLP tasks.