Symmetric Dot-Product Attention for Efficient Training of BERT Language Models

Anonymous ACL submission

Abstract

Initially introduced as a machine translation model, the Transformer architecture has now become the foundation for modern deep learning architecture, with applications in a wide range of fields, from computer vision to natural language processing. Nowadays, to tackle increasingly more complex tasks, Transformer-based models are stretched to enormous sizes, requiring increasingly larger training datasets, and unsustainable amount of compute resources. The ubiquitous nature of the Transformer and its core component, the attention mechanism, are thus prime targets for efficiency research.

In this work, we propose an alternative compatibility function for the self-attention mechanism introduced by the Transformer architecture. This compatibility function exploits an overlap in the learned representation of the traditional scaled dot-product attention, leading to a symmetric with pairwise coefficient dot-product attention. When applied to the pre-training of BERT-like models, this new symmetric attention mechanism reaches a score of 79.36 on the GLUE benchmark against 78.74 for the traditional implementation, leads to a reduction of 6% in the number of trainable parameters, and reduces the number of training steps required before convergence by half.

1 Introduction

Since its introduction in 2017, the Transformer architecture powered by its scaled dot-product attention mechanism (Vaswani et al., 2017) has become the core component of modern deep-learning architectures and has enabled researchers to achieve breakthroughs in both natural language processing (NLP) and computer vision tasks such as language modelling (Brown et al., 2020), machine translation (Raffel et al., 2019), speech processing (Radford et al., 2022), and image recognition (Dosovitskiy et al., 2020). One of the many successes of the Transformer lies in its ability to operate and learn in an unsupervised setting from unstructured textual data, as well as its ability to handle complex and varied structures such as graphs, images, and sentences by increasing the model's number of layers. However, this trend has led to the emergence of machine learning models so enormous that the gap between the amount of compute resources available to many research groups and the amount needed to stay competitive is increasing year after year (Togelius and Yannakakis, 2023), and by training larger and larger models, brought deep-learning's energy consumption to unsustainable amounts (Thompson et al., 2021).

Efficient Transformer implementations are a popular area of research with many recent contributions on encoding and dense representation of tokens (Su et al., 2021), hardware-optimized implementation of attention (Dao et al., 2022), or Transformer implementations for long document processing (Beltagy et al., 2020). While the attention mechanism itself has been studied extensively (Niu et al., 2021), and several improvements to its computational complexity have been achieved (Kitaev et al., 2020; Zhou et al., 2021), it is still primarily computed via the dot-product between a *query* and a *key* (see Figure 1). Vaswani et al. (2017) highlight the difficulty of determining a proper compatibility function, and suggest that a more sophisticated compatibility function than dot product may be beneficial.

In this work, we propose alternative compatibility functions for the attention mechanism, i. e., the scaled dot-product attention mechanism. With this approach, we aim to improve the training efficiency of Transformer-based models and to reduce their resource consumption. We especially focus on the *self-attention* mechanism of BERT (Devlin et al., 2018), a Transformer-based encoder model.

Our contributions can be summarized as follows:

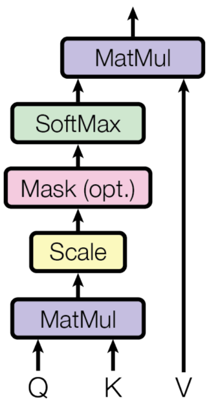


Figure 1: Scaled Dot-Product Attention (Vaswani et al., 2017)

- We introduce an alternative formula to replace the scaled dot-product attention (Section 2) that takes advantage of the underlying symmetric structure of attention, in order to reduce the number of parameters and improve the computational efficiency of the model.
- We benchmark our approach by training several BERT models on three attention mechanism setups as well as two different model sizes (Section 3).

We demonstrate that our new attention formula reduces the number of parameters of the model by 6%, and achieves a reduction of the number of training steps required for model convergence by 50% without sacrificing accuracy (Section 4). Finally, we discuss the effects of our proposed compatibility function on training efficiency, and situate our approach within the context of research on efficient Transformer-based models (Section 5).

2 Improving the Attention Mechanism

Modern Transformer-based models are neural networks that rely on the scaled dot-product attention mechanism introduced by Vaswani et al. (2017). We propose two variations of this mechanism: a symmetric dot-product and a symmetric with pairwise factors dot-product, that lead to a reduction in the number of parameters of the self-attention layer.

2.1 Scaled Dot-Product Attention

The scaled dot-product attention given by the following equation (Equation 1) is an operator on three input matrices, queries Q, keys K and values V. We focus on the dot product QK^T between queries Q and keys K, which is responsible for measuring the compatibility between tokens. The compatibility $\mathbf{A}$ is an operator on two input tokens ($x, y \in \mathbb{R}^h$), that computes the dot-product of the projections of x and y respectively through the operators $\mathbf{Q}$ and $\mathbf{K}$:

$$Attn(Q,K,V) = Softmax\left(\frac{\mathbf{QK^T}}{\sqrt{d}}\right)V \quad (1)$$

Given two linear operators $\mathbf{Q} : \mathbb{R}^h \to \mathbb{R}^d$ and $\mathbf{K} : \mathbb{R}^h \to \mathbb{R}^d$, we define a compatibility operator $\mathbf{A} : \mathbb{R}^h \times \mathbb{R}^h \to \mathbb{R}$, such that:

$$\mathbf{A}(x,y) = \mathbf{Q}(x) \cdot \mathbf{K}(y)^T \quad (2)$$

We challenge the necessity of using two different operators to compute the affinity of the self-attention encoder layer of the Transformer block. Since both $\mathbf{Q}$ and $\mathbf{K}$ are operators on the same token space, it is reasonable to assume that the representations they learn share some features. In that case, since the original expression (Equation 1) does not enforce any feature sharing, it may possess redundant parameters that will need to be learned twice.

We attempt to make this feature sharing property explicit in the compatibility operator expression, in order to remove redundant parameters, reduce overall model size, and improve convergence rate.

2.2 Symmetric Dot-Product attention

A simple way to make the feature sharing property explicit, is to enforce the following relation $\mathbf{Q} = \mathbf{K}$ between the two operators. This ensures that $\mathbf{Q}$ and $\mathbf{K}$ share features and results in the symmetric compatibility operator:

$$\mathbf{A}_{sym}(x,y) = \mathbf{Q}(x) \cdot \mathbf{Q}(y)^T \quad (3)$$

2.3 Pairwise Dot-Product Attention

One aspect that needs to be considered is the amount of features shared between the two operators. Complete overlap in terms of features may be detrimental to the overall performance of the attention mechanism, e. g., it could prevent the model to learn asymmetric relationships. Thus, we suggest the following compatibility operator where the amount of feature sharing is learned during training. To achieve this, we start with an operator $\mathbf{L}$ that will be shared, and we define operators $\mathbf{Q}$ and $\mathbf{K}$ as

Function	Expression	Parameters
original	$\mathbf{Q}(x)\mathbf{K}(y)^T$	$\mathcal{O}(3h^2)$
symmetric	$\mathbf{Q}(x)\mathbf{Q}(y)^T$	$\mathcal{O}(2h^2)$
pairwise	$\mathbf{Q}(x)S\mathbf{Q}(y)^T$	$\mathcal{O}(2h^2 + h/n)$

Table 1: Parameter count of the attention layer per compatibility function.

a composition of $\mathbf{L}$ with a base change, resulting in the following compatibility operator (Equation 5):

Given a linear operator $\mathbf{L} : \mathbb{R}^h \to \mathbb{R}^d$ and two square matrices $W_q, W_k \in \mathbb{R}^{d\times d}$, we define two linear operators $\mathbf{Q} : \mathbb{R}^h \to \mathbb{R}^d$ and $\mathbf{K} : \mathbb{R}^h \to \mathbb{R}^d$, such that:

$$\begin{aligned} \mathbf{Q}(x) &= \mathbf{L}(x) \cdot W_q \\ \mathbf{K}(x) &= \mathbf{L}(x) \cdot W_k \end{aligned} \tag{4}$$

Let $S \in \mathbb{R}^{d\times d}$ be the product $S = W_q \cdot W_k^T$, we define a compatibility operator $\mathbf{A} : \mathbb{R}^h \times \mathbb{R}^h \to \mathbb{R}$, such that:

$$\begin{aligned} \mathbf{A}(x, y) &= \mathbf{Q}(x) \cdot \mathbf{K}(y)^T \\ \mathbf{A}(x, y) &= \mathbf{L}(x) \cdot W_q \cdot W_k^T \cdot \mathbf{L}(y)^T \\ \mathbf{A}(x, y) &= \mathbf{L}(x) \cdot S \cdot \mathbf{L}(y)^T \end{aligned} \tag{5}$$

This operator can be interpreted as a weighted dot-product whose weights are stored in S, a matrix of pairwise factors. To make the expression consistent with the previously established expressions (Equation 2 and Equation 3), we relabel the $\mathbf{L}$ operator with the letter $\mathbf{Q}$, resulting in the following pairwise compatibility operator (Equation 6).

$$\mathbf{A}_{pair}(x, y) = \mathbf{Q}(x) \cdot S \cdot \mathbf{Q}(y)^T \tag{6}$$

2.4 Parameter Count

For a Transformer block of n heads, with input size h and attention size d, we give the parameter count formula for a complete block (with parameters from Q, K and V). We note that most Transformer implementations impose $d = h/n$.

As shown in Table 1, the symmetric compatibility operator uses two thirds of the original number of parameters. For the pairwise compatibility operator, the parameter count also depends on the number of attention heads, it converges towards 2/3 of the original number of parameters as the number of attention heads increases.

Config	Operator	Parameters
$BERT_{small}$	original	28,795,194
	symmetric	27,744,570 (3.65%)
	pairwise	27,875,642 (3.19%)
$BERT_{base}$	original	109,514,298
	symmetric	102,427,194 (6.47%)
	pairwise	103,017,018 (5.93%)

Table 2: Parameter count per model configuration and compatibility function (relative amount of parameters saved compared to the original). $Bert_{small}$: nlayers: 4, nheads: 8, hidden size: 512, intermediate size: 2048. $Bert_{base}$: nlayers: 12, nheads: 12, hidden size: 768, intermediate size: 3072.

In this section, we introduced two alternative compatibility functions for the attention mechanism, a symmetric dot-product operator and a symmetric with pairwise factors dot-product operator. In the following sections we will refer to them respectively as the *symmetric operator* and the *pairwise operator*, we will refer to the traditional scaled dot-product operator as the *original operator*.

3 Experiments

To evaluate the symmetric and pairwise operators against the original operator, we train and evaluate several Transformer-based encoder models, each using a different compatibility operator as part of the self-attention mechanism. The models are trained under the same conditions. First, we pre-train the models, because we want to measure the evaluation loss during training to see if our modifications have an impact on the training efficiency and the accuracy of the model. Then, we evaluate each model on the GLUE benchmark (Wang et al., 2019b) to evaluate the model's accuracy on relevant downstream tasks, such as, sentence acceptability (Warstadt et al., 2018), sentiment analysis (Socher et al., 2013), sentence similarity (Cer et al., 2017), and natural language inference (Williams et al., 2018; Rajpurkar et al., 2016). Finally, we select model checkpoints during training and evaluate those checkpoints on GLUE to measure the models' accuracy on downstream tasks during training.

3.1 Pre-Training Dataset

To pre-train our models, we select a subset of 30 million English documents from the OSCAR corpus (Abadji et al., 2022; Jansen et al., 2022) by applying content quality filters (See Appendix A).

Using OSCAR data instead of the BookCorpus (Zhu et al., 2015) and Wikipedia dumps is recommended for training BERT models (Geiping and Goldstein, 2023) and ensures that the amount of documents is large enough for single epoch training.

This training dataset is tokenized using the pretrained *bert-base-uncased* tokenizer (Devlin et al., 2018) and sentences are aggregated into groups of 512 tokens. After tokenization, the resulting dataset contains 137 million training samples, 70 billion tokens and 10,000 test samples.

3.2 Model Architectures

We prepare three variations of the BERT model (Devlin et al., 2018) using the original, the symmetric and the pairwise operators. We also train on two model sizes, *bert-small* and *bert-base*.

As shown in Table 2, the symmetric and pairwise operators lead to significant reduction in the number of parameters, 3.65% and 3.19% for the *bert-small* model, 6.47% and 5.93% for the *bert-base* model.

In the following sections, we refer to a *bert-base* model as $BERT_{base}$ when it uses the original operator, $BERT_{base,sym}$ or $BERT_{base,pair}$ when it uses the symmetric or pairwise operator respectively.

3.3 Pre-Training Setup

We follow the pre-training setup described by Devlin et al. (2018). The models are trained on a pure masked language modeling task with masking probability of 0.15 and batch size of 256 samples per training steps. Models are trained on 200,000 steps with a linear learning rate of 10^{-4} and learning rate warm-up during the first 10,000 steps. For the optimizer, we use Adam (Kingma and Ba, 2014) with weight decay, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-12}$, resulting in models pre-trained on 26 billion tokens. We measure evaluation cross-entropy loss during training to assess the training efficiency of our models.

3.4 Benchmark Fine-Tuning Setup

After pre-training, the models are fine-tuned and benchmarked on the GLUE dataset (Wang et al., 2019b) to assess their natural language understanding (NLU) capabilities. Each model is fine-tuned on the provided downstream task training dataset for 5 epochs, with a batch size of 16 and a linear learning rate of $1 \cdot 10^{-5}$. This benchmarking step is repeated on 5 downstream trials with different seeds. We measure individual task's scores, benchmark average and standard deviation across all trials. For each model, we measure: the combined F1 and accuracy on the Microsoft Research Paraphrase Corpus ***mrcp*** (Dolan and Brockett, 2005), Matthews correlation on the Corpus of Linguistic Acceptability ***cola*** (Warstadt et al., 2018), matched and mis-matched accuracy on the Multi-Genre Natural Language Inference Corpus ***mnli*** (Williams et al., 2018), accuracy on the Quora Question Pairs ***qqp***[1], accuracy on the Recognizing Textual Entailment dataset ***rte*** (Dagan et al., 2006; Bar Haim et al., 2006; Giampiccolo et al., 2007; Bentivogli et al., 2009), the combined Pearson and Spearman correlation on the Semantic Textual Similarity Benchmark ***stsb*** (Cer et al., 2017), accuracy on the Stanford Question Answering Dataset ***qnli*** (Rajpurkar et al., 2016), and accuracy on the Stanford Sentiment Treebank ***sst2*** (Socher et al., 2013). The Winograd schema challenge ***wnli*** task has been excluded from the evaluation following the recommendation of Devlin et al. (2018).

Compared to the original BERT setup or more recent compute optimized fine-tuning setups (Geiping and Goldstein, 2023), we choose to fine-tune for a longer time (5 epochs instead of 3) and with a lower learning rate ($1 \cdot 10^{-5}$ instead of $4 \cdot 10^{-5}$), to have a more stable fine-tuning experience and reduce the risk of lucky seeding. With this choice, we aim to have a fairer evaluation of the models.

3.5 Checkpoint Benchmarking

We want to evaluate how downstream accuracy evolves during pre-training. We extract checkpoints during training and evaluate them on the GLUE benchmark. Each checkpoint is fine-tuned and evaluated on GLUE using the previously established fine-tuning setup.

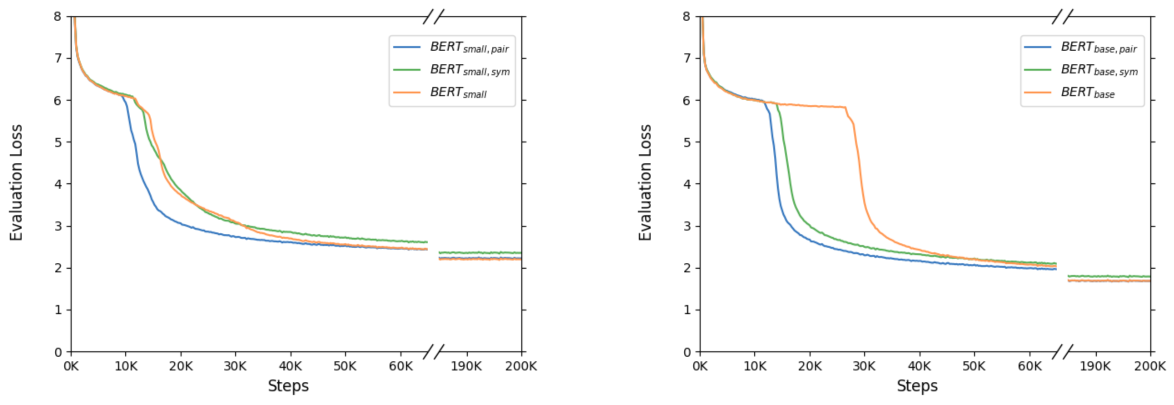
4 Results

In this section, we present the results of our experiments, the pre-training of our three variants (Figure 2), the scores they reach on the GLUE benchmark (Table 3) once fully trained and the evolution of the GLUE score during training (Figure 3).

4.1 Pre-Training Experiment

Figure 2b shows that the symmetric and pairwise variant converge much faster than the original vari-

[1]`https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs`



(a) $BERT_{small}$ (b) $BERT_{base}$

Figure 2: BERT pre-training evaluation loss. Models are trained for 200,000 steps, the evaluation loss is the cross-entropy loss. We observe that the models using the symmetric and pairwise operators converge faster than the original model.

ant for the $BERT_{base}$ model. The evaluation loss of the original variant remains on the initial plateau until step 25,000, when it sharply decreases. The symmetric variant remains on the initial plateau until step 13,500 and the pairwise variant until step 12,000. We also note that the original and pairwise variants will eventually reach the same evaluation loss plateau, while the symmetric variant remains above the two other variants with an additional absolute error of 0.1.

Comparing Figures 2a and 2b, we observe the impact of model size on training efficiency. When the model size increases, the original variant's initial plateau is expanded from step 12,000 to step 25,000, while the symmetric and pairwise variant were almost unaffected.

4.2 GLUE Benchmark Fine-Tuning

Table 3 shows that the pairwise variant performs better than the original variant with an increase of 0.6 points on the average GLUE score for both model sizes. The symmetric variant, however, is outperformed by the original variant in both cases, with a drop of 4 points on the average GLUE score. We also observe that both proposed variants have a lower standard deviation on the *bert-base* model.

4.3 GLUE Benchmarking Along Training Steps

Figure 3 shows that the improved training efficiency observed during pre-training translates to a faster convergence rate on the GLUE benchmark as well. The pairwise and original variants both reach a final average GLUE score of approximately

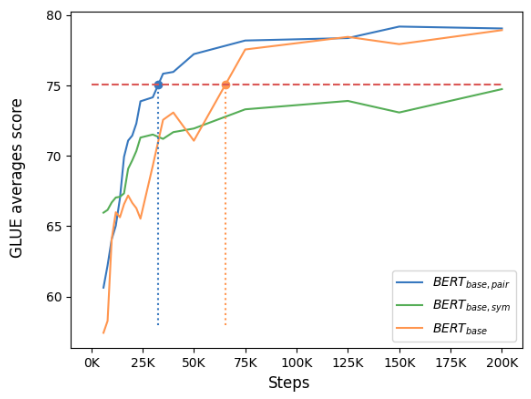


Figure 3: Average GLUE score over training steps. Checkpoints are sampled during training and evaluated on the GLUE benchmark. The red dashed line correspond to 95% of the final GLUE average score.

79. The pairwise variant achieves 95% (a score of 75) of its final value after 30,000 steps, the original variant reaches the same score after 65,000 steps.

We also observe a smoother evolution of the accuracy for the pairwise variant compared to the original variant. The experiment also highlights the performance drop of the symmetric variant when compared to the original variant.

5 Discussion

5.1 Pre-training Efficiency

During the pre-training experiment, we observed that both variants $BERT_{base,sym}$ and $BERT_{base,pair}$ outperformed the original variant $BERT_{base}$ in terms of convergence rate (they

Model	GLUE Score	mrpc	cola	mnli(m/mm)	qqp	rte	stsb	qnli	sst2
$BERT_{small}$	72.72 (0.07)	81.04	21.39	**77.16/77.76**	86.08	54.87	82.20	**85.45**	88.49
$BERT_{small,sym}$	69.61 (0.32)	76.81	10.29	75.25/75.72	85.13	55.74	77.83	82.79	86.90
$BERT_{small,pair}$	**73.38 (0.37)**	**82.34**	**24.21**	76.37/76.89	**86.67**	**56.25**	**84.13**	84.97	**88.60**
$BERT_{base}$	78.74 (0.63)	85.30	44.35	**81.66/82.07**	88.86	59.42	87.30	88.76	**90.92**
$BERT_{base,sym}$	74.82 (0.36)	78.36	35.22	78.66/79.05	87.70	53.43	84.47	86.90	89.56
$BERT_{base,pair}$	**79.36 (0.37)**	**87.83**	**46.91**	81.60/82.02	**88.89**	**60.58**	**86.88**	**88.78**	90.78

Table 3: Average GLUE scores Average score over the GLUE benchmark per model with individual task breakdown. $BERT_{base,pair}$ achieves the best **GLUE Score** of 79.36 with a standard deviation of 0.37, in comparison to $BERT_{base,pair}$ which achieve a **GLUE Score** of 78.74 with a standard deviation of 0.63.

initiated the learning and reached their respective plateau faster), for a *bert-base* model the convergence rate seems to be two times faster. However, $BERT_{base}$ and $BERT_{base,pair}$ ultimately met around the same evaluation loss, while $BERT_{base,sym}$ performed a little worse.

One obvious explanation for the improved convergence rate can be found in the reuse of the **Q** operator, this can impact convergence rate in three way:

- The accumulation of two loss gradients per forward/backward pass instead of a single one, resulting in an effect similar (but not exactly equivalent) to doubling the learning rate for the parameters of the **Q** operator.
- The reduction in the number of parameters.
- Sharing representation for both **Q** and **K** operators. If they do learn a subset of the same features, then enforcing a shared representation for both of them will reduce the amount of learning required.

These effects explain why both $BERT_{base,sym}$ and $BERT_{base,pair}$ converge much faster than $BERT_{base}$.

While converging faster than $BERT_{base}$, $BERT_{base,sym}$ did not reach the same evaluation loss. It is fair to assume that this is a modelling issue and not a size issue since $BERT_{base,pair}$ outperformed $BERT_{base,sym}$ with a similar number of parameters. Thus, we can conclude that symmetry is not a desired property of the compatibility function of the attention mechanism.

5.2 GLUE Benchmark

The evaluation of the three variants on the GLUE benchmark shows that $BERT_{base,pair}$ is more accurate than $BERT_{base}$, reaching an average score of 79.39 against 78.74 respectively. The evaluation also shows that the standard deviation of the average score across five trials is lower for both $BERT_{base,pair}$ and $BERT_{base,sym}$, with a standard deviation of 0.37 and 0.36 against 0.63 for $BERT_{base}$.

This confirms that the training efficiency improvement observed on the pre-training task translates to the fine-tuning task and leads to improvement on the downstream task's accuracy. With the added benefit of making the fine-tuning task more stable, as shown by the lower standard deviation.

We also note that the fairly small 0.1 difference in evaluation loss during training for $BERT_{base,sym}$ has translated to a 4 points accuracy drop on the evaluation benchmark, echoing our remark on the need to model asymmetric relationships.

With these results, we experimentally prove that our pairwise operator improves the training efficiency of Transformer-based models, leading to a faster convergence rate and overall lower training loss. These improvements also translate to downstream task benchmarks. Models using the pairwise compatibility operator are indeed more accurate than the ones using the original compatibility operator.

5.3 GLUE Evaluation During Pre-Training

Running the benchmark evaluation on our three models at several steps of the pre-training experiment shows that the training efficiency we observed translates well into downstream accuracy. Our $BERT_{base,sym}$ and $BERT_{base,pair}$ converge faster towards their respective final values, similarly to the training loss observed on the pre-training task. $BERT_{base}$ reaches 95% of its final value after 65,000 steps and $BERT_{base,pair}$ after 30,000 steps. While $BERT_{base}$ eventually catches up and improves on $BERT_{base,sym}$, $BERT_{base,pair}$ is consistently the better model.

This final experiment highlights the improved training efficiency induced by the pairwise compatibility operator. The faster convergence rate observed during pre-training is also observed on the downstream task evaluation, confirming the convergence rate improvement by a factor of two for the $BERT_{base,pair}$ model.

6 Related Work

While the Transformer architecture (Vaswani et al., 2017) popularized the use of the attention mechanism, and contributed to its adoption in the field of NLP, the attention mechanism was first introduced to NLP with recurrent neural networks applied to machine translation (Bahdanau et al., 2016). In this setting, the compatibility operator is a simple multi-layer perceptron with non-linear activation operating on the concatenation of inputs encoded by the recurrent neural network. This definition of the attention mechanism was then extended to other compatibility operator: Luong et al. (2015) mention the use of the dot-product between the recurrent neural network's hidden state, propose to explicitly integrate token positions into the compatibility operator, and even suggest the use of a general dot-product operator $score(h_t, \overline{h_s}) = h_t^T W \overline{h_s}$. Those initial influences have also been documented by Galassi et al. (2020) and Niu et al. (2021), where the general dot-product appears as a weighted dot-product between query and keys $f(q, K) = q^T W K$. Thus the pairwise compatibility operator we introduce is an evolution of the general dot-product, where we constrain it to a single and shared linear operator $\mathbf{Q}$ before applying the bilinear form of matrix S, resulting in the following operator $\mathbf{A}(x, y) = \mathbf{Q}(x) S \mathbf{Q}(y)^T$.

To the best of our knowledge, our work is the first application of the general dot-product with enforced symmetry to the self-attention mechanism of the Transformer architecture. While we focused on the compatibility operator, recent improvements have been made on other parts of the attention mechanism. Namely, He and Hofmann (2023) proposed to simplify the entire Transformer block by carefully removing components and achieved an impressive 15% weight reduction, while still relying on the traditional scaled dot-product.

7 Conclusion

In this work, we revisited the traditional scaled dot-product used in the Transformer self-attention mechanism. We challenged the use of two distinct operators to compute the dot-product between queries and keys, in favor of single shared operator and a weighted dot-product with pairwise factors. By doing so, we enforced a symmetric structure to the compatibility operator of the attention mechanism, reducing the number of parameters used in the Transformer layer by a third. As a result, when applied to BERT models, our pairwise compatibility operator reduces the overall number of parameters of the model by 6%, reduces the number of pre-training steps required by half and improves accuracy on the GLUE benchmark, making Transformer-based encoders more efficient, faster to train and lowering their resource requirements. We believe our work can be applied to other Transformer architectures like decoder and encoder-decoder models, as well as to other NLP tasks like machine translation and language modeling. And, more generally, to the concept of attention as a whole, where it would bring improvement in other fields such as computer vision.

For future work, we plan to evaluate the pairwise dot-product attention mechanism on larger models reaching into the billion parameters, and to evaluate our attention mechanism on other benchmarks, like SuperGLUE (Wang et al., 2019a) and SQuAD2.0 (Rajpurkar et al., 2018). We plan on implementing the pairwise compatibility operator for the cross-attention mechanism, and evaluating it on decoder and encoder-decoder tasks like language modeling and machine translation. Finally, we want to evaluate our pairwise dot-product attention not only on natural language processing tasks, but also on tasks from other fields, computer vision, time series forecasting and reinforcement learning.

Limitations

Our work focuses only on the application and evaluation of alternative compatibility functions for the self-attention mechanism of Transformer-based encoder models, benchmarked on NLU tasks. While our work has shown positive results on this specific use case, we cannot draw any conclusion on its application to decoder models and pure language modeling tasks, or encoder-decoder model and machine translation tasks. Those use cases rely on the cross-attention mechanism for which the shared representation we exploit with our pairwise compatibility operator may not be appropriate.

While we suggest that the $\mathbf{Q}$ and $\mathbf{K}$ operators learn a shared representation, we did not perform any analysis of the original scaled-dot product attention or of our pairwise dot-product attention. The parameter redundancy of multi-head attention models has been covered in Bian et al. (2021). However, to our knowledge the parameter redundancy between the query and the key operator of a single head has not been studied.

While our work showed positive improvements on the training efficiency of BERT-like models of fairly small sizes (100 million parameters), it is not enough to draw conclusions on its efficiency on very large models (e.g., 10 billion parameters).

We decided to benchmark our models on GLUE, as it is the most popular benchmark for NLU evaluation. However, this benchmark as been largely surpassed by modern machine learning models. For that reason, new benchmarks have been introduced, such as SuperGLUE (Wang et al., 2019a) or SQuAD2.0 (Rajpurkar et al., 2018).

Reproducibility Statement

All software related to our experiments with the attention mechanism is available at `anonymizedurl`. It uses the PyTorch (Paszke et al., 2017) and HuggingFace Transformer (Wolf et al., 2020) frameworks. The necessary steps to recreate the training dataset are documented at `anonymizedurl`, the dataset used for training is available at `anonymizedurl`.

References

Julien Abadji, Pedro Ortiz Suarez, Laurent Romary, and Benoît Sagot. 2022. Towards a cleaner document-oriented multilingual crawled corpus. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 4344–4355, Marseille, France. European Language Resources Association.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2016. Neural machine translation by jointly learning to align and translate.

Roy Bar Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. 2006. The second PASCAL recognising textual entailment challenge.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *CoRR*, abs/2004.05150.

Luisa Bentivogli, Ido Dagan, Hoa Trang Dang, Danilo Giampiccolo, and Bernardo Magnini. 2009. The fifth PASCAL recognizing textual entailment challenge.

Yuchen Bian, Jiaji Huang, Xingyu Cai, Jiahong Yuan, and Kenneth Church. 2021. On attention redundancy: A comprehensive study. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 930–945, Online. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada. Association for Computational Linguistics.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The PASCAL recognising textual entailment challenge. In *Machine learning challenges. evaluating predictive uncertainty, visual object classification, and recognising tectual entailment*, pages 177–190. Springer.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

William B Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the International Workshop on Paraphrasing*.

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. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, abs/2010.11929.

Andrea Galassi, Marco Lippi, and Paolo Torroni. 2020. Attention in natural language processing. *IEEE transactions on neural networks and learning systems*, 32(10):4291–4308.

Jonas Geiping and Tom Goldstein. 2023. Cramming: Training a language model on a single gpu in one day. In *International Conference on Machine Learning*, pages 11117–11143. PMLR.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third PASCAL recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*, pages 1–9. Association for Computational Linguistics.

Bobby He and Thomas Hofmann. 2023. Simplifying transformer blocks. *arXiv preprint arXiv:2311.01906*.

Tim Jansen, Yangling Tong, Victoria Zevallos, and Pedro Ortiz Suarez. 2022. Perplexed by Quality: A Perplexity-based Method for Adult and Harmful Content Detection in Multilingual Heterogeneous Web Data. *arXiv e-prints*, page arXiv:2212.10440.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. *CoRR*, abs/1508.04025.

Zhaoyang Niu, Guoqiang Zhong, and Hui Yu. 2021. A review on the attention mechanism of deep learning. *Neurocomputing*, 452:48–62.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch.

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. 2022. Robust speech recognition via large-scale weak supervision.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, abs/1910.10683.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. *CoRR*, abs/1806.03822.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Jianlin Su, Yu Lu, Shengfeng Pan, Bo Wen, and Yunfeng Liu. 2021. Roformer: Enhanced transformer with rotary position embedding. *CoRR*, abs/2104.09864.

Neil C. Thompson, Kristjan Greenewald, Keeheon Lee, and Gabriel F. Manso. 2021. Deep learning's diminishing returns: The cost of improvement is becoming unsustainable. *IEEE Spectrum*, 58(10):50–55.

Julian Togelius and Georgios N. Yannakakis. 2023. Choose your weapon: Survival strategies for depressed ai academics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019a. Superglue: A stickier benchmark for general-purpose language understanding systems. *CoRR*, abs/1905.00537.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019b. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2018. Neural network acceptability judgments. *CoRR*, abs/1805.12471.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le

Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. 2021. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 11106–11115.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *The IEEE International Conference on Computer Vision (ICCV)*.

A OSCAR Filters

To ensure good quality of our training dataset, we filter OSCAR dumps with the following rules:

- From the UT1 Blocklists project[2], we exclude the following categories:
  - "agressif"
  - "adult"
  - "cryptojacking"
  - "dangerous_material"
  - "phishing"
  - "warez"
  - "ddos"
  - "hacking"
  - "malware"
  - "mixed_adult"
  - "sect"
- We exclude documents whose *harmful perplexity score* is below 5.0 and above 100,000.
- Following recommendation from (Abadji et al., 2022), we exclude documents which have been flagged with quality warnings.

[2]http://dsi.ut-capitole.fr/blacklists/index_en.php