Data-Informed Global Sparseness in Attention Mechanisms for Deep Neural Networks

Ileana Rugina[*, 1], **Rumen Dangovski**[*, 1], **Li Jing** [1], **Preslav Nakov**[2], **Marin Soljačić** [1]
(*) equal contribution
[1] Massachusetts Institute of Technology
[2] Mohamed bin Zayed University of Artifcial Intelligence, UAE
{irugina, rumenrd, ljing, soljacic}@mit.edu
preslav.nakov@mbzuai.ac.ae

Abstract

Attention mechanisms play a crucial role in the neural revolution of Natural Language Processing (NLP). With the growth of attention-based models, several pruning techniques have been developed to identify and exploit sparseness, making these models more efficient. Most efforts focus on hard-coding attention patterns or pruning attention weights based on training data. We propose Attention Pruning (AP), a framework that observes attention patterns in a fixed dataset and generates a global sparseness mask. AP saves 90% of attention computation for language modeling and about 50% for machine translation and GLUE tasks, maintaining result quality. Our method reveals important distinctions between self- and cross-attention patterns, guiding future NLP research. Our framework can reduce both latency and memory requirements for any attention-based model, aiding in the development of improved models for existing or new NLP applications. We have demonstrated this with encoder and autoregressive transformer models using Triton GPU kernels and make our code publicly available at https://github.com/irugina/AP

Keywords: attention mechanism, sparse computational graphs, lottery ticket hypothesis

1. Introduction

Given enough computational power, the scalability of the attention mechanism (Bahdanau et al., 2015; Hermann et al., 2015; Vaswani et al., 2017) will allow for building ever larger Natural Language Processing (NLP) models with billions of parameters (Shoeybi et al., 2019; Radford et al., 2019; Raffel et al., 2020; Brown et al., 2020; Chowdhery et al., 2022; Smith et al., 2022; Fedus et al., 2022; Du et al., 2022; Anil et al., 2023).

While impressive, these advances also pose a responsibility to the Natural Language Processing (NLP) community to interpret the behavior of the hundreds of attention heads in a single model, and potentially to reduce the number of computations. Responding to this challenge, previous work has taken pioneering steps to discover and to explain the sparseness in the attention patters (Vig and Belinkov, 2019; Clark et al., 2019; Kovaleva et al., 2019; Yeh et al., 2023; Ruscio et al., 2023; Kobayashi et al., 2023; Biderman et al., 2023; Zhang et al., 2023; Li et al., 2023). Here, we argue that as the number of heads grows in the range of thousands, automatic measures would be needed to discover and to impose sparseness to such models.

In this paper we introduce a simple task-agnostic data-informed pruning method for attention mechanisms: *Attention Pruning*. We train Transformer-based models and we analyze the *global* observed attention patterns, averaged over all input sequences in the train set, in order to identify and to remove weak connections between input tokens.

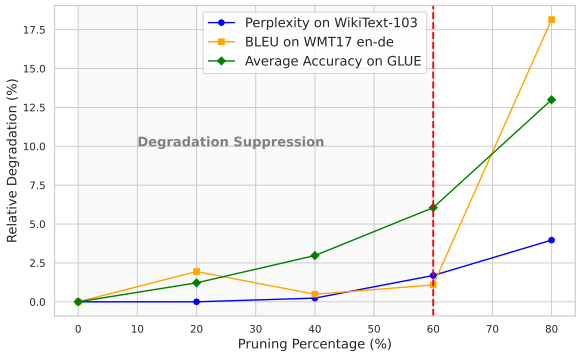


Figure 1: Attention pruning maintains performance and reduced attention computations. Pruning can often enable more efficient and interpretable models with only a modest decrease in performance.

Following Frankle and Carbin (2019), we then retrain these models, enforcing sparseness through masking, and we demonstrate that attention mechanisms incorporate extraneous connections between the input tokens: we obtain comparable performance while using sparse attention patterns for NLP tasks such as language and sequence-to-sequence (seq2seq) modelling, as well as prediction on GLUE tasks. Figure 1 summarizes the impact of using our pruning method on standard NLP tasks.

These global sparseness patterns could help improve both interpretability and inference-time computational efficiency for widely-used attention mod-

LREC-COLING 2024, pages 4392–4403
20-25 May, 2024.

els. Our contributions are as follows:

- We present a novel pruning method for attention patterns, focusing on the model's computational graph, not attention weights, and showcase both theoretical and empirical gains.
- Our data-informed, global method prunes attention while retaining information and closely preserving original results.
- As an application-agnostic method, we investigate pruning impacts on language and seq2seq modeling, including GLUE task predictions.
- In seq2seq experiments, we examine attention pruning in encoder self-attention, decoder self-attention, and encoder–decoder attention, highlighting key differences.

2. Related Work

There are several fruitful directions for research focused on improving the computational efficiency and the interpretability of the attention mechanism. Sparseness plays a central role in all of them, as simple attention mechanisms inherently scale quadratically with sequence length and assign non-zero correlation between any two input tokens.

One line of research is that of reducing computational complexity of attention using insights provided by empirical observations of patterns. Child et al. (2019) introduced two sparse matrix factorizations that reduce the computational complexity from $O(N^2)$ to $O(N\sqrt{N})$. Kitaev et al. (2020) created a sparse attention mechanism with $O(N \log N)$ computational complexity, which is achieved by using local sensitivity hashing to cluster tokens that should attend to each other and then only computing attention within tokens from the same chunks. More directly related to our work is that of Beltagy et al. (2020) and Zaheer et al. (2020) who looked directly at sparsifying the attention patterns rather than at the underlying matrix factorization, and reduced the computational complexity of attention from $O(N^2)$ to $O(N)$ using GPU kernels (Gray et al., 2017). A key difference between these approaches and ours is that we do not impose any a priori restrictions on the type of attention patterns we can generate.

Another successful approach has been to adapt low-rank matrix approximation methods to simplify attention computations. Xiong et al. (2021) adapted Nyström's method (Baker, 1979). Wang et al. (2020) leveraged the Johnson–Lindenstrauss lemma to introduce projections and to improve complexity. In contrast to this line of work our contribution is easier to implement because (*i*) we do not require architectural modifications and we only change a few lines of code in practice, and (*ii*) we do not require optimizing the numerical stability of any mathematical methods and we introduce very few and simple hyper-parameters.

Correia et al. (2019) and Peters et al. (2019) explored directly incorporating sparseness into Transformer models through the choice of activation functions and introduced the α-entmax functions. This encompasses both softmax for $\alpha = 1$ and sparsemax (or projections onto the probability simplex) for $\alpha = 2$. For any $\alpha > 1$ α-entmax is sparse. Peters et al. (2019) provided an efficient implementation and experimented with $\alpha = 1.5$. We leverage global attention masks, rather than creating a sparse attention pattern for each key-value pair, and we manage both to provide quantifiable speed guarantees and to achieve higher sparseness in practice.

There has been a lot of research on understanding over-parameterization and on developing methods that make BERT models faster and more lightweight (Ganesh et al., 2020). Previous work has found that different attention heads encode similar patterns and hence these heads are not always all necessary (Michel et al., 2019; Kovaleva et al., 2019; Voita et al., 2019), and thus good performance can be achieved by removing entire attention heads at test time. Sajjad et al. (2020) pruned entire Transformer layers at a time and again obtained good performance while removing a large percentage of the model's parameters. Our pruning method takes a more fine-grained approach and prunes individual connections rather than entire heads or layers, and thus it could be used in conjunction with the above-mentioned methods.

3. Attention Pruning

3.1. Scaled Dot-Product Attention

Attention mechanisms are used to learn connections (correlations) between two sequences of lengths N and M, respectively.

Transformer models use the scaled dot-product attention introduced in (Vaswani et al., 2017), which takes as input three matrices: a matrix $\mathbf{Q} \in \mathbb{R}^{M \times d_k}$ composed of query vectors $q \in \mathbb{R}^{d_k}$, a matrix $\mathbf{K} \in \mathbb{R}^{N \times d_k}$ composed of key vectors $k \in \mathbb{R}^{d_k}$, and a third matrix $\mathbf{V} \in \mathbb{R}^{N \times d_v}$, which groups value vectors $v \in \mathbb{R}^{d_v}$.

The scaled dot-product outputs a transformation of the sequence of length M governed by the values associated with other sequence's tokens, as well as the relative strength of the connection between any two tokens from the two sequences:

$$\mathbf{A} = \mathrm{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V}$$

The general sequence-to-sequence Transformer model uses three types of attention layers: two self-

attention mechanisms for the encoder and for the decoder, respectively, as well as a third encoder–decoder attention, which we will call *cross-attention*. Additionally, each attention layer is a multi-headed attention mechanisms that concatenates several instances of the dot-product attention described above.

3.2. Our Pruning Algorithm

Tailor attention to the dataset. The attention mechanism (Vaswani et al., 2017) creates a matrix with NM connections where N is the length of a source sequence, i.e., its number of tokens, and M is the length of the target sequence. In practice, in order to compute high-level representations of sources and targets, a portion of these connections is ignored by pre-trained models, which could be observed with visualization tools on pre-trained Transformer-based models (Vig and Belinkov, 2019; Clark et al., 2019; Kovaleva et al., 2019). More specifically, there are entries in the attention matrix whose values are close to zero *with high probability*, and thus there is no correlation between the corresponding tokens. We would like to prune such connections, thus reducing the noise coming from such entries.

For each possible tuple (type, l, m), where type is either a *self-attention encoder*, a *self-attention decoder* or an *encoder–decoder*, $l \in \{1, \dots L\}$ is the layer in the Transformer, $m \in \{1, \dots H\}$ is the index of the head within the Transformer layer, we calculate the corresponding *average attention matrix* $\overline{\mathbf{A}}^{(\text{type},l,m)}$ from the corresponding attention matrices $\mathbf{A}_i^{(\text{type},l,m)}$ generated by source-target pairs in the training set as follows:

$$\overline{\mathbf{A}}^{(\text{type},l,m)} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{A}_i^{(\text{type},l,m)}, \tag{1}$$

where n is the size of the training set. The summation is element-wise.

We introduce a single additional hyper-parameter p: the percentage of entries we want to prune in each multi-head attention mechanism. We would prune an entry in the attention matrix if almost all of the source-target pairs in the training dataset are below a threshold $\tau^{(type,l)}$, given by the p percentile of $\left[\overline{\mathbf{A}}^{(\text{type},l,1)}, \dots, \overline{\mathbf{A}}^{(\text{type},l,H)}\right]$. In other words, the global percentage p defines a layer-dependent pruning threshold $\tau^{(type,l)}$, which corresponds to the p percentile of all attention entries within that layer, across all heads. The pruning mask $\mathbf{M}_p$, which depends on p, is computed as follows:

$$\mathbf{M}_p^{(\text{type},l,m)} = (-\infty) \cdot \left[\!\!\left[\overline{\mathbf{A}}^{(\text{type},l,m)} < \tau^{(type,l)} \right]\!\!\right].^{1} \tag{2}$$

We use the Iverson bracket notation, where the comparison operation yields a matrix, whose entries are 1 if the comparison is satisfied, and 0 otherwise (using broadcasting for the scalars).

The mask modifies the calculation of the attention matrix as follows:

$$\widetilde{\mathbf{A}}_{i,p}^{(t)} = \text{softmax}\left(\frac{\mathbf{Q}_i^{(t)} \mathbf{K}_i^{(t)\top} + \mathbf{M}_p^{(t)}}{\sqrt{d_k}} \right) \mathbf{V}_i^{(t)},$$

where $t = (\text{type}, l, m), \mathbf{Q}_i^{(\text{type},l,m)} \in \mathbb{R}^{N \times d_k}$, $\mathbf{K}_i^{(\text{type},l,m)} \in \mathbb{R}^{M \times d_k}$, $\mathbf{V}_i^{(\text{type},l,m)} \in \mathbb{R}^{N \times d}$, d_k and d are hidden dimensions. Note that the only difference between $\widetilde{\mathbf{A}}_{i,p}^{(\text{type},l,m)}$ and $\mathbf{A}_i^{(\text{type},l,m)}$ is that in the calculation for the latter we replace $\mathbf{M}_p^{(\text{type},l,m)}$ with the zero matrix. Also, note that our pruning mechanism only reduces the computational graph, and *does not affect the number of parameters of the model*.

An interpretation of inducing the mask $\mathbf{M}_p$ onto our computational graph is that we constrain the inductive bias of the attention mechanism by adding the inductive bias of *whether a pair of source–target positions correlate*, which we learn from the training dataset by inspecting the outputs of a pre-trained model. Observing whether a pair of positions correlate in the training dataset would allow us to infer a positional inductive bias from the dataset itself that we can incorporate during training. Therefore, we propose the following algorithm.

Attention Pruning (AP). Choose a model F with attention matrices in its computational graph and a percentage p for pruning. Then follow the steps below:

1. Train the model F, initialized with weights $\boldsymbol{\theta}$, on the training set $\mathcal{D} = \{(x_i, y_i)\}_{i=1,\dots,n}$ via validation on the $\mathcal{D}_{\text{valid}}$ split to obtain optimized weights $\boldsymbol{\theta}^*$ and then compute the accuracy (or any other desired evaluation measure)

2. For each possible tuple (type, l, m), calculate the average $\overline{\mathbf{A}}^{(\text{type},l,m)}$ from the attention matrices

   $$\mathbf{A}_i^{(\text{type},l,m)} \equiv \mathbf{A}_i^{(\text{type},l,m)}(x_i; \boldsymbol{\theta}^*)$$

   for $i = 1, \dots, N$, i.e., generated by each training example in $\mathcal{D}$, using equation 1.

[1] We implement this by adding a large negative value before applying softmax; this value depends on the codebase.

p (%)	Perplexity
0	24.157
20	24.157
40	24.214
60	24.566
80	25.115
90	26.011

Table 1: Attention Pruning for Transformer-XL-base trained on WikiText-103 can sparsify 90% of its attention patterns, while maintaining good performance.

3. For each possible tuple (type, l, m), calculate the mask $\mathbf{M}_p^{(\text{type},l,m)}$ from $\overline{\mathbf{A}}^{(\text{type},l,m)}$ using equation 2, then obtain a new model F' from F by replacing the attention matrices with $\widetilde{\mathbf{A}}_{i,p}^{(\text{type},l,m)}$ as in Section 3.
4. Train F' on $\mathcal{D}$ and compute the accuracy $a_{\text{test},p}^{\text{pruned}}$ on $\mathcal{D}_{\text{test}}$.

Our method is inspired by the Lottery Ticket Hypothesis (Frankle and Carbin, 2019). However, except from sharing Step 1, our AP method diverges significantly from other methods inspired by the Lottery Ticket Hypothesis, such as (Yu et al., 2020), as we study sparseness not in the parameters of the neural network, but in the attention patterns of the model on a fixed dataset. Such a distinction is important, because while the sparseness of the weight matrices is usually not interpretable, the sparseness of the attention patterns is, as it has been observed in the literature (Kovaleva et al., 2019; Raganato et al., 2020; Beltagy et al., 2020).

Given our method and the above motivation we now ask the following: (1) Can AP reveal a global sparseness for attention patterns? (2) Can that sparseness be deployed for efficient inference? (3) Can AP yield an insight into the sparseness properties of different attention types? We answer positively all these questions in the following experimental sections.

4. Language Modelling Experiments

We first test pruning attention matrices on the WikiText-103 (Merity et al., 2017) language modelling task. We use the Transformer-XL base architecture (Dai et al., 2019), which adds recurrence to Transformer models by caching self-attention hidden states between text segments.

We can see in Table 1 and Figure 1(a) that we can prune over 80–90% of the attention entries and still maintain good performance on language modelling tasks. Figure 2 shows Transformer XL's prune masks, averaged over all its layers and heads, when using $p = 30\%$ or $p = 90\%$. We speculate that

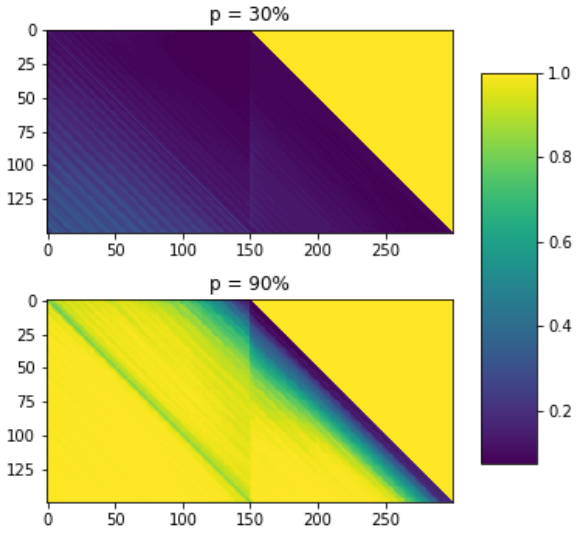


Figure 2: Transformer-XL pruning masks (binary valued) averaged over all layers and attention heads for $p \in \{30\%, 90\%\}$. AP prunes entries in the left half (attention to past sequences) more aggressively than the conventional self-attention entries in the right half. Note that the right half also has an auto-regressive mask.

attention pruning performs so well here because it enables Transformer-XL to pay attention to long sequences only when the distant past is actually relevant.

5. Machine Translation Experiments

We further test our method on encoder-decoder Transformer models using various translation tasks and the implementation by Ott et al. (2019). In all experiments, we use half-precision floating point (FP16).We run two sets of experiments on the WMT17 English–German (en-de) and the IWSLT14 German–English (de-en) machine translation tasks following the default base Transformer architectures from Ott et al. (2019) (`transformer_iwslt_de_en` and `transformer_wmt_en_de`, respectively), and we evaluate on the best model, measured by the highest validation-split BLEU score in the case of IWSLT14 de-en, and by the lowest validation loss in the case of WMT17 en-de.

We prune all three types of attention — *self-attention-encoder*, *self-attention-decoder* and *encoder-decoder*—, and we run five experiments per translation task, for $p \in \{20, 40, 50, 60, 80\}$, for both IWSLT14 de-en and WMT17 en-de.

Table 2 summarizes the pruned models performance for the two translation tasks. We can see that AP performs significantly worse here than it did for the language modeling task above. This suggest that a finer-grained analysis on pruning

p (%)	0	20	40	60	80
IWSLT14	34.94	32.08	24.16	14.18	5.00
WMT17	26.73	23.04	4.03	1.28	0.30

Table 2: BLEU scores results from the IWSLT14 de-en and WMT17 en-de translation tasks. We can see that pruning all types of attention mechanism leads to a fast drop in performance.

p (%)	Self-Enc	Self-Dec	Cross
0		34.94	
20	34.53	34.94	33.50
40	33.70	34.94	24.38
50	33.56	35.08	22.60
60	33.68	34.91	15.08
80	33.67	34.90	6.39

Table 3: Results on the IWSLT14 de-en translation task when pruning each type of attention mechanism. Pruning cross-attention connections sharply hurts the model performance.

p (%)	Self-Enc	Self-Dec	Cross
0		26.73	
20	26.21	26.73	21.68
40	26.60	26.73	3.72
50	26.60	26.51	2.46
60	26.15	26.61	1.60
80	26.11	23.56	0.69

Table 4: Results obtained on the WMT17 en-de translation task when pruning each type of attention mechanism. Pruning up to 80% of encoder self-attention connections results in a minimal drop of model performance.

specific attention types, with which we proceed below.

5.1. Attention Type Analysis

We now look at whether the three types of attention mechanism require specialized pruning. In order to isolate the effects of AP, we first prune only one of the three types of attention mechanisms. The results are summarized in Tables 3 and 4.

For both datasets, we clearly see the same trend: the Transformer models are more sensitive to pruning cross-attention patterns than to removing self-attention, which is in line with what was reported in previous work (You et al., 2020). Figures 3a and 3c represent the average attention patterns observed on the IWSLT14 de-en dataset for cross-attention and the encoder's self-attention mechanism, respectively. A direct comparison between the two suggests that cross-attention mechanisms

p (%)	IWSLT14 de-en	WMT17 en-de
0	34.94	26.73
20	34.92	26.21
40	33.70	26.60
50	33.68	26.19
60	33.64	26.44
80	33.81	21.88

Table 5: BLEU scores for both translation tasks when pruning self-attention mechanisms. These are more robust under AP, even when the encoder and the decoder attentions are simultaneously pruned.

are more brittle, which is in part because they exhibit variable context windows. This makes sense since in this case the queries and the keys are generated from different sequences.

5.2. Pruning Self-Attention Only

We prune the two types of self-attention patterns (*self-attention-encoder* and *self-attention-decoder*) for $p \in \{20, 40, 50, 60, 80\}$. The results are shown in Table 5.

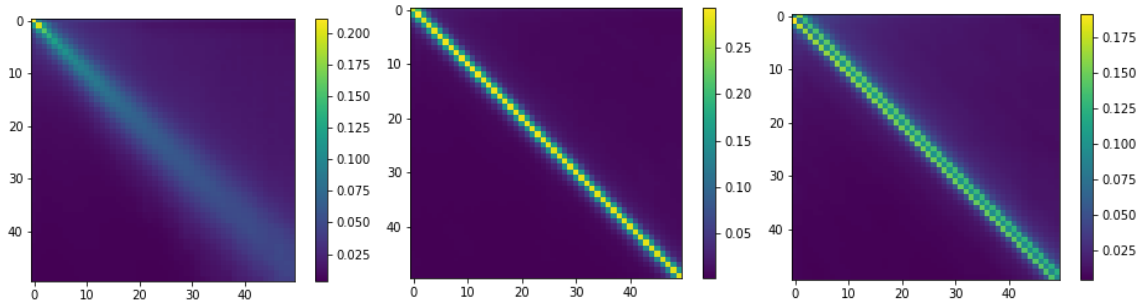
In agreement with 5.1, we find that we can prune large percentage of self-attention connections while maintaining good BLEU scores. The average attention patterns among all encoder self-attention heads for IWSLT14 are shown in Figure 3c. We can see that they are indeed sharper than in the case of cross-attention, and we show that a fixed window encodes the relevant contextual information for processing an input token. Moreover, we observe that numerous attention heads encode very similar patterns, in agreement with (Voita et al., 2019).

The results for WMT17 en-de (see Figure 1(b), Table 5) are particularly encouraging. We can prune over 60% of the self-attention entries, while losing less than 1 BLEU point absolute. This suggests that AP is particularly robust when used with large datasets, possibly because we use summary statistics.

6. Sparse Normalization

One disadvantage of our pruning method is that we use a single pruning percentage p for all attention patterns within a Transformer layer. Translation models are sequence-to-sequence models from a source fragment of length M to a target fragment of length N, where M and N are *not invariant* across a dataset. Because of this, we either leave extraneous noisy connection between tokens in shorter sequences or we lose important information when modeling longer sequences.

This problem would be ameliorated if we masked attention mechanisms according to pat-



(a) (b) (c)

Figure 3: IWSLT14 de-en train dataset attention patterns: (a) cross-attention with variable context window, (b) encoder self-attention with 1.5-entmax activation for sharper patterns, and (c) encoder self-attention with constant context window.

p (%)	BLEU score			
	Enc	Dec	Cross	Enc+Dec
0		34.93		
20	34.38	34.93	32.02	34.55
40	34.76	34.93	24.92	34.60
50	34.33	34.93	16.03	34.37
60	33.77	35.05	10.97	33.91
80	33.17	34.93	4.38	33.21

Table 6: Experiments on IWSLT14 de-en using 1.5-entmax. We can see that pruning self-attention mechanisms maintains good performance at higher sparseness percentages than those induced by 1.5-entmax alone. This is in agreement with the trend observed in Section 5.1, where we saw that pruning cross-attention patterns yields a sharp drop in performance.

terns sharper than those produced by the softmax activation function. Therefore, we turn to the α-entmax activation, which was used in (Correia et al., 2019) and (Peters et al., 2019), and which encourages sparseness in the attention patterns. We repeated all IWSLT14 de-en experiments from Section 5 using $\alpha = 1.5$. Figure 3b shows the average observed attention pattern among the encoder self-attention heads in a model that uses 1.5-entmax as its normalizing activation function. We can see that they are indeed sharper than the analogous ones, which we presented in Figure 3c.

Table 6 demonstrates how attention pruning performs in conjunction with 1.5-entmax on the IWSLT14 de-en translation task. We would like to note that the behavior in 5.1 regarding cross-attention's lack of robustness to pruning is even more pronounced now.

7. AP with BERT on GLUE

In order to analyze the sparseness of BERT on GLUE tasks (Wang et al., 2019), we study the vanilla BERT architecture `bert-base-cased`, and we apply our AP method on top of it. We train on the standard GLUE tasks with searched hyper-parameters listed in Appendix B.

In Table 7 and Figure 1(c), we present the results of our experiments. Generally, we observe that we can perform attention pruning on BERT heads, while maintaining the performance. For example, AP with $p = 20$ loses only 1.00 points absolute on average compared to the baseline $p = 0$ model. Another important observation is that, for some GLUE tasks, we can prune even more than 50% of the attention weights while maintaining competitive scores, e.g., for MNLI, QNLI, QQP, SST-2, and SST-B. For the remaining tasks —RTE, MRPC and CoLA—, we maintain reasonable performance by pruning a sizable fraction of the attention weights.

8. Why Use Tailored Attention Masks

We explore the importance of using attention masks tailored to specific datasets by comparing against two other scenarios: (*i*) prune random entries in attention patterns, and (*ii*) prune using an attention mask learned on a different dataset.

8.1. Machine Translation

We prune self-attention mechanisms in encoder–decoder models trained with softmax on the IWSLT14 translation tasks for $p \in \{20, 40, 50, 60, 80\}$ using either random or out-of-distribution attention masks, and we compare to the baseline results above. For the out-of-distribution experiments, we generate attention masks on IWSLT14 en-de. Tables 8 and 9 show the results. We note that data-informed masks outperform random pruning by a large

Accuracy (%)										
p (%)	MNLI-m	MNLI-mm	QNLI	QQP	RTE	SST-2	MRPC	CoLA	STS-B	Average
0	84.07	83.44	90.91	87.51	65.7	91.97	88.77	57.78	88.39	82.06
20	84.00	83.42	89.72	86.37	64.44	91.11	87.12	55.99	87.34	81.06
40	83.42	83.70	88.76	84.73	62.82	89.91	84.45	52.33	86.48	79.62
50	83.32	82.87	87.81	83.84	62.09	89.05	83.08	48.56	85.28	78.43
60	82.54	81.98	87.19	83.10	61.37	88.82	82.04	45.05	81.70	77.09
80	79.29	78.64	82.37	81.32	57.22	84.52	78.57	34.80	65.89	71.40
90	75.40	75.23	77.23	77.45	49.46	80.56	79.41	20.28	51.39	65.16

Table 7: BERT on GLUE. Attention Pruning reduces the attention computations by tens of percentage points, while maintaining comparable performance. We report Spearman correlation for STS-B, F1 for QQP and MRPC, and accuracy for the rest. The reported results are the median from five reruns on Dev.

p (%)	0	20	40	50	60	80
AP	34.94	34.92	33.70	33.68	33.64	33.81
Rand.	34.94	32.51	30.66	27.96	26.56	5.93

Table 8: Results for Machine Translation with IWSLT14 de-en task. Pruning random attention entries yields sharp drop in performance as the pruning percentage p increases. AP yields up to 80% sparseness, while maintaining good performance.

p (%)	0	20	40	50	60	80
base	30.57	30.37	27.61	27.28	27.51	27.46
ood	30.57	30.44	27.46	27.60	27.56	27.46

Table 9: Results from the IWSLT14 en-de translation tasks. For simple attention patterns, such as those in self-attention mechanism for translation tasks, AP is robust under distributional shifts.

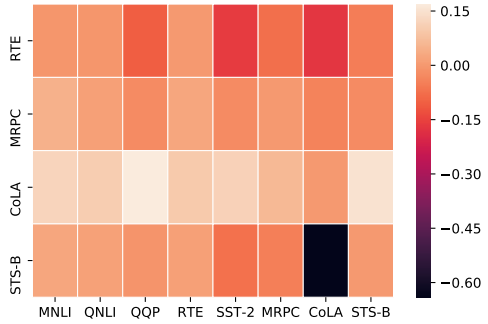


Figure 4: Relative accuracy when a GLUE task (indicated in the rows: STS-B, CoLA, MRPC, RTE) is trained with AP ($p = 40$) using the attention patterns of GLUE tasks (indicated in the columns: all GLUE tasks). The relative accuracy is computed so that the in-domain experiment is zero, and the out-of-domain experiments show deviations in accuracy.

margin, especially as the percentage p increases. However, using attention masks gathered for the other translation direction does not meaningfully influence our results. We speculate that this is because we only look at self-attention patterns, which in the case of translation are very sharp and exhibit a constant context window.

8.2. BERT

We perform out-of-domain experiments by training GLUE tasks with AP using attention patterns from all GLUE tasks. In Figure 4, we show the relative difference in accuracy for our experiments on four GLUE datasets. Note that the columns for SST-2 and CoLA show significantly negative relative accuracy, which means that the attention patterns from these datasets are not useful for the majority of the GLUE tasks. This is in line with the nature of the datasets: SST-2 and CoLA are two tasks in GLUE that do not involve pairs of sentences. Particularly detrimental to the performance is using CoLA masks on the STS-B dataset (black entry in Figure 4). Our experiment in the figure is for $p = 40$, but we observed a similar pattern for a variety of pruning percentages.

9. Towards Efficient Hardware Implementation of Attention Pruning

9.1. MACs

In order to quantify the computational advantage we obtain from pruning, we estimate the number of Multiply-Accumulate Operations, or MACs for short (Randell, 1971), executed by a simple attention mechanism during the forward pass of the model.

Let the input be a tensor $x \in \mathbb{R}^{B \times N \times d}$, where B is the batch size, N is the number of tokens in

Kernel	Pruning	Efficiency	
	p (%)	Time (s)	Memory (GB)
Pytorch	0	95.80	6.24
Triton	0	95.41	6.85
Triton	90	86.44 (↓9.4%)	5.00 (↓27%)

Table 10: Proof-of-concept empirical gains from using AP inference on SQuAD. Memory tracks the GPU.

a sequence, and d is the number of embedding dimensions. In Appendix A we show that with AP we achieve a reduction of

$$\text{fraction of MACs} = \frac{4d + (2-p)N}{4d + 2N}. \quad (3)$$

This is particularly helpful for large $N \gg d$, suggesting that AP might be computationally beneficial in applications with long input sequences such as summarization or question answering.

9.2. Bert Benchmark

We demonstrate empirical gains using block-sparse GPU kernels. Gray et al. (2017) first implemented efficient sparse matrix multiplication operations and applied them to both dense and convolutional layers. Child et al. (2019) extended this work to attention mechanisms with certain pre-determined sparseness patterns. Tillet et al. (2019) introduced Triton, a language and compiler used to generate optimized GPU code that allows for higher design flexibility than Pytorch. We use the DeepSpeed[2] implementation of sparse attention, which requires efficient sampled dense-dense and sparse-dense multiplications as well as softmax operations.

Since in order to obtain performance gains we need to use sparseness patterns structured around blocks, as a proof-of-concept, we turn our attention to question answering applications and apply AP to the Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 2016) and use a blocksize of 16. We choose SQuAD for hardware benchmarking as the sequences are longer (up to 384 tokens) than those in the GLUE benchmark. We fine-tune BERT on a single GeForce GTX 1080 GPU for two epochs with a learning rate of 3e-5 and a batch size of 4. The length of the sequences is capped at 384 tokens. At test time, we use a batch size of 256.

The results in Table 10 suggest that we can empirically reduce both inference time and GPU memory consumption using AP and sparse kernels. At the same time, we do not compromise performance noticeably: Pytorch and Triton kernels for $p = 0$ yield 81.02 Exact and 88.63 F1 scores, while AP with Triton for $p = 90$ yields 79.62 Exact and 87.32 F1 scores. We would like to underscore the 27% reduction in memory, in particular, as memory limitations usually prevent utilization of attention-based models when hardware is constrained. Thus AP is especially promising in this context.

9.3. Llama2 7B Benchmark

We evaluate our pruning method on the Llama2 7B model using the Wikitext-2 dataset, with a focus on language modeling [3]. We use a context length and stride of 4096. Attention pattern statistics are gathered from the training split, and evaluation is performed on the test split. Due to computational constraints, we do not fine-tune the model weights. Given this limitation, we concentrate on moderate pruning percentages. This approach demonstrates how memory improvements can be scaled to recent large language models even with low batch sizes; all experiments are conducted with a batch size of one.

	Pytorch	Triton
Pruning %	0	60
Memory (GB)	19.51	16.88
Perplexity	6.48	6.72

Table 11: Memory efficiency in Llama2 7B model: Pruning cuts forward pass memory from 19.51 GB to 16.88 GB, over a 13.92 GB baseline to load the model, with minimal perplexity increase.

We summarize our results in Table 11. Loading the model onto the GPU with half-precision weights requires 13.92 GB of memory. We managed to reduce the additional memory requirements for a forward pass from 5.58 GB to 3.07 GB, achieving a reduction of over 40%, with only a minimal increase in perplexity.

10. Conclusion and Future Work

We introduced Attention Pruning, a novel method that leverages data-informed sparseness for pruning attention. Through controlled experiments on a variety of tasks (from language modeling to machine translation and GLUE tasks predictions), we showed that our method prunes most computations using pre-computed attention patterns while maintaining, or even improving, performance. Our application to seq2seq tasks enabled us to study attention patterns in self- and cross-attention, revealing key distinctions between the two.

In future work, we aim to assess our method on other models, NLP tasks, and datasets of various

[2] https://www.deepspeed.ai

[3] https://github.com/irugina1/llama-attention-pruning

sizes, as well as optimize Attention Pruning for existing hardware. We believe co-design approaches for efficient sparse kernels and their successful utilization can enhance Attention Pruning's scalability. To encourage further co-design efforts, we are releasing our code to the research community.

11. Limitations

Our study required extensive experiments across three types of tasks, language modeling, machine translation and BERT fine-tuning. If one wants to reproduce our analysis from scratch they would need to rely on extensive GPU compute. However, we will release our code, in hopes to alleviate the work of researchers who would like to perform a similar study.

12. Ethics Statement

In this research, we introduce a novel method for pruning attention mechanisms in natural language processing models. While our primary goal is to reduce computational complexity and enhance model efficiency, we recognize the potential ethical implications of our work.

First, it is important to note that our developed pruning technique can lead to the creation of more efficient and less resource-intensive models. This has positive implications for sustainability, as it could help reduce the energy consumption and environmental footprint of large-scale NLP models.

However, we also acknowledge that more efficient models might contribute to the development and deployment of increasingly powerful NLP systems, which could be misused for malicious purposes, such as disinformation campaigns or automated harassment. Therefore, we encourage the research community to adopt responsible practices in the development and deployment of NLP models with attention pruning and to continuously evaluate the potential risks and societal impact of these technologies.

Moreover, as our pruning method is applied to pre-trained models, the potential biases embedded in the training data could still persist. Therefore, we emphasize the importance of addressing and mitigating biases in NLP models and training data to ensure that the resulting systems do not perpetuate harmful stereotypes or unfair treatment of certain social groups.

Finally, we commit to transparently sharing our code, as mentioned earlier, and findings to encourage further collaboration within the research community, fostering an open and responsible approach to the development and improvement of attention pruning techniques.

Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. 2023. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*.

Dzmitry Bahdanau, KyungHyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations*, ICLR '15, San Diego, CA, USA.

Christopher T. H. Baker. 1979. *Numerical Integration in the Treatment of Integral Equations*, pages 44–53. Birkhäuser Basel, Basel.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document Transformer. *arXiv preprint arXiv:2004.05150*.

Luisa Bentivogli, Ido Dagan, Hoa Trang Dang, Danilo Giampiccolo, and Bernardo Magnini. 2009. The fifth pascal recognizing textual entailment challenge. In *Proceedings of Text Analysis Conference*, TAC '09, Gaithersburg, Maryland, USA.

Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. 2023. Pythia: A suite for analyzing large language models across training and scaling. *arXiv preprint arXiv:2304.01373*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared 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 M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher 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. *arXiv preprint arXiv:2005.14165*.

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.

Z. Chen, H. Zhang, X. Zhang, and L. Zhao. 2018. Quora question pairs.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the Second BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, ACL '19, pages 276–286, Florence, Italy.

Gonçalo M. Correia, Vlad Niculae, and André F. T. Martins. 2019. Adaptively sparse transformers. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, EMNLP-IJCNLP '19, pages 2174–2184, Hong Kong, China.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, EMNLP '19, Florence, Italy.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, et al. 2022. Glam: Efficient scaling of language models with mixture-of-experts. In *International Conference on Machine Learning*, pages 5547–5569. PMLR.

William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *The Journal of Machine Learning Research*, 23(1):5232–5270.

Jonathan Frankle and Michael Carbin. 2019. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. In *Proceedings of the 7th International Conference on Learning Representations*, ICLR '19, New Orleans, LA, USA.

Prakhar Ganesh, Yao Chen, Xin Lou, Mohammad Haris Ali Khan, Yin Yang, Deming Chen, Marianne Winslett, Hassan Sajjad, and Preslav Nakov. 2020. Compressing large-scale transformer-based models: A case study on bert. arXiv prerint arXiv:2002.11985.

Scott Gray, Alec Radford, and Diederik P. Kingma. 2017. GPU kernels for block-sparse weights.

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Proceedings of the Annual Conference on Neural Information Processing Systems: Advances in Neural Information Processing Systems 28*, pages 1693–1701.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient Transformer. In *Proceedings of the 8th International Conference on Learning Representations*, ICLR '20, Virtual Conference.

Goro Kobayashi, Tatsuki Kuribayashi, Sho Yokoi, and Kentaro Inui. 2023. Feed-forward blocks control contextualization in masked language models. *arXiv preprint arXiv:2302.00456*.

Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. 2019. Revealing the dark secrets of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4365–4374, Hong Kong, China. Association for Computational Linguistics.

Yiran Li, Junpeng Wang, Xin Dai, Liang Wang, Chin-Chia Michael Yeh, Yan Zheng, Wei Zhang, and Kwan-Liu Ma. 2023. How does attention work in vision transformers? a visual analytics attempt. *IEEE Transactions on Visualization and Computer Graphics*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *Proceedings of the 5th International Conference on Learning Representations*, ICLR '17, Toulon, France.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Teaching machines to read and comprehend. In *Proceedings of the Annual Conference on Neural Information Processing Systems: Advances in Neural Information Processing Systems 32*, NIPS '19, pages 14014–14024.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, NAACL-HLT '19, pages 48–53, Minneapolis, Minnesota.

Ben Peters, Vlad Niculae, and André F. T. Martins. 2019. Sparse sequence-to-sequence models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, ACL '19, pages 1504–1519, Florence, Italy.

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

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Alessandro Raganato, Yves Scherrer, and Jörg Tiedemann. 2020. Fixed encoder self-attention patterns in transformer-based machine translation. In *Findings of the Association for Computational Linguistics*, EMNLP '20, pages 556–568, Virtual Conference.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. *arXiv e-prints*, page arXiv:1606.05250.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text.

B. Randell. 1971. Ludgate's analytical machine of 1909. *The Computer Journal*, 14:317–326.

Valeria Ruscio, Valentino Maiorca, and Fabrizio Silvestri. 2023. Attention-likelihood relationship in transformers. *arXiv preprint arXiv:2303.08288*.

Hassan Sajjad, Fahim Dalvi, Nadir Durrani, and Preslav Nakov. 2020. Poor man's bert: Smaller and faster transformer models. *arXiv preprint arXiv:2004.03844*.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-LM: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*.

Shaden Smith, Mostofa Patwary, Brandon Norick, Patrick LeGresley, Samyam Rajbhandari, Jared Casper, Zhun Liu, Shrimai Prabhumoye, George Zerveas, Vijay Korthikanti, et al. 2022. Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model. *arXiv preprint arXiv:2201.11990*.

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.

Philippe Tillet, H. T. Kung, and David Cox. 2019. Triton: An intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, MAPL 2019, page 10–19, New York, NY, USA. Association for Computing Machinery.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the Annual Conference on Neural Information Processing Systems: Advances in Neural Information Processing Systems*, NIPS '17, pages 5998–6008, Long Beach, CA, US.

Jesse Vig and Yonatan Belinkov. 2019. Analyzing the structure of attention in a transformer language model. In *Proceedings of the Second BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, ACL '19, pages 63–76, Florence, Italy.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, ACL '19, pages 5797–5808, Florence, Italy.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *Proceedings of the 7th International Conference on Learning Representations*, ICLR '19, New Orleans, LA, USA.

Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-attention with linear complexity.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference.

Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. 2021. Nyströmformer: A nyström-based algorithm for approximating self-attention.

Catherine Yeh, Yida Chen, Aoyu Wu, Cynthia Chen, Fernanda Viégas, and Martin Wattenberg. 2023. Attentionviz: A global view of transformer attention. *arXiv preprint arXiv:2305.03210*.

Weiqiu You, Simeng Sun, and Mohit Iyyer. 2020. Hard-coded Gaussian attention for neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7689–7700, Online. Association for Computational Linguistics.

Haonan Yu, Sergey Edunov, Yuandong Tian, and Ari S. Morcos. 2020. Playing the lottery with rewards and multiple languages: lottery tickets in RL and NLP. In *Proceedings of the 8th International Conference on Learning Representations*, ICLR '20, Virtual Conference.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2020. Big bird: Transformers for longer sequences. In *Advances in Neural Information Processing Systems*, volume 33, pages 17283–17297. Curran Associates, Inc.

Haojie Zhang, Mingfei Liang, Ruobing Xie, Zhenlong Sun, Bo Zhang, and Leyu Lin. 2023. Better pre-training by reducing representation confusion. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 2280–2291.

A. Estimation of the MACs in Equation 3

In order to process the input x, we need to follow this sequence of operations: (*i*) Compute the *key*, *query*, and *value* matrices; each of these is obtained by applying a fully connected transformation $\mathbb{R}^d \rightarrow \mathbb{R}^d$ and in total requires $3BNd^2$ MAC operations; (*ii*) Compute attention scores P; we need to compute Bh matrix multiplications of the form $X_1 \times X_2$ between matrices $X_1 \in \mathbb{R}^{N \times d_k}$ and $X_2 \in \mathbb{R}^{d_k \times N}$; this takes BhN^2d_k MACs. (*iii*) Compute attention patterns: we need to calculate Bh matrix products of the form $X_1 \times X_2$ between matrices $X_1 \in \mathbb{R}^{N \times N}$ and $X_2 \in \mathbb{R}^{N \times td_k}$; this costs us BhN^2d_k MACs; (*iv*) Apply a fully connected layer to compute the attention mechanism output, which takes BNd^2 MACs. Now let us consider what happens when a fraction p ($0 \leq p \leq 1$) of the entries in the matrix $P \in \mathbb{R}^{B \times h \times N \times N}$ are pruned: in step 3 we only need to do a fraction $1 - p$ of the work we used to and therefore, we achieve reduction of

$$\text{fraction of MACs} = \frac{4d + (2-p)N}{4d + 2N}.$$

B. Details about GLUE

We train on the following GLUE tasks: MNLI (Williams et al., 2018), QQP (Chen et al., 2018), STS-B (Cer et al., 2017), SST-2 (Socher et al., 2013), RTE (Bentivogli et al., 2009), MRPC (Dolan and Brockett, 2005), QNLI (Rajpurkar et al., 2016), CoLA (Warstadt et al., 2019). For each task, we fine-tune the BERT model on a single GPU for three epochs with a learning rate of 2e-5 and a batch size of 32. The maximum length of the input sequences is 128. Since the datasets are small, we run each of our experiments with five different random seeds in order to be able to capture the mean and the standard deviation of the results.