AdaPrune: Pruning Transformer with Sparse Regularization

Anonymous ACL submission

Abstract

The key component of transformer architecture is the multi-head self-attention (MHA) and feed forward neural network (FFN). In this paper, we observe that, across many applications, MHA component is nonsymmetric and FFN component is sparse within transformer architecture. Leveraging this observation, we propose a new method, AdaPrune, to utilize sparse regularization to conduct structure pruning in MHA and FFN modules. This method selects task-specific valuable heads in multi-head attention modules and effective blocks in feed forward layers during the fine-tuning stage, while maintaining the original performance of full transformer model. Extensive experiments show that AdaPrune can achieve competitive performance on these tasks while significantly reduce the computation cost.

1 Introductions

Transformer models (Vaswani et al., 2017) have proven its overwhelming performance among various natural language processing (NLP) problems such as neural machine translation (Liu et al., 2020; Ott et al., 2018), natural language understanding (Devlin et al., 2019; Liu et al., 2019) and automatic speech recognition (Wang et al., 2020a). Despite becoming the fundamental architecture for many applications, transformer model is hard to deploy in practice due to the high computation cost. This is attributed to the multi-head self-attention module (MHA) and feed-forward neural network (FFN) inside the standard transformer architecture (Child et al., 2019; Wu et al., 2020). On the one hand, MHA has quadratic complexity in compute and memory with respect to sequence length of inputs (Vaswani et al., 2017). On the other hand, FFN contributes most of computation FLOPS as it normally consists of a large hidden dimensions (e.g. FFN contains more than 50% of parameters for GPT-3 model families (Brown et al., 2020)).

Figure 1: Trade-off illustration between performance (Y-axis) and inference cost (X-axis) of Standard Prune and AdaPrune for RoBERTa base model.

There are several lines of research work focusing on improving the efficiency of transformer architecture: (i) optimizing self-attention mechanism (Tay et al., 2020) such as introducing sparsity (Child et al., 2019) and low-rank projection (Wang et al., 2020b; Choromanski et al., 2020) into attention computation; (ii) knowledge distillation from large pre-trained transformer models to smaller counterpart such as DistilBERT (Sanh et al., 2020); (iii) mixed precision such as training and deploying quantized model (Ott et al., 2019); (iv) model pruning such as removing some heavy component of the transformer model to speed up the inference (Frankle and Carbin, 2019; Chen et al., 2020).

In this paper, we propose a new angle for improving transformer efficiency. We first provide an analysis across a range of NLP classification and translation tasks, to show that, in the standard transformer architecture, (i) if we define the self-attention mechanism as a functional mapping, it is nonsymmetrical across each layer and each heads. (ii) feed forward neural network is extremely sparse. In details, we observe that more than 97% tensors from FFN is equal to or proximate to zero after the activation.

Leveraging this observation, we proposed a new
method to improve transformer efficiency, named AdaPrune, which utilizes sparse regularization to conduct structured pruning. Our method consists of three steps as described in Section 3: fine-tuning with sparse regularization on in-domain task, prune model (not fine-tuned) based on component rank in MHA and FFN obtained from first step, and fine-tune pruned model on the same target task. We apply our methodology on pre-trained language model such as RoBERTa (Liu et al., 2019) and investigate the performance-efficiency tradeoff on GLUE benchmark. As shown in Fig. 1, AdaPrune provides average 2% improvement in GLUE benchmark compared to the simple structured prune, which remove each heads and FFN blocks randomly. We believe this method is complementary and can be combined with the existing various transformer efficiency techniques.

2 Motivation

In this section, we will introduce the standard transformer architecture and two important observations inside the multi-head attention and feed forward neural network inside transformer.

2.1 Transformer Architecture

Each transformer layer (Vaswani et al., 2017) includes two important module: MHA and FFN. The MHA can be described as

\[
\text{MHA}_i(X_i) = \text{Concat}[\text{head}_i(X_i)]_{i=1}^H \mathbf{W}_{\text{out}},
\]

where \(X_i\) is the hidden representations in \(i\)th layer, \(H\) is number of heads in each layer. The query, key and value are produced via linear mapping \(\mathbf{W}_q^i, \mathbf{W}_k^i\) and \(\mathbf{W}_v^i\). The FFN is defined as

\[
\mathbf{Y}_i = \text{ReLU}(\mathbf{W}_{\text{fc1},i}\text{MHA}_i(X_i) + b_{\text{fc1},i}),
\]

\[
\mathbf{C}_i = \mathbf{W}_{\text{fc2},i}^T \mathbf{Y}_i + b_{\text{fc2},i},
\]

where \(\mathbf{W}_{\text{fc1},i}, \mathbf{W}_{\text{fc2},i}^T \in \mathbb{R}^{d_{\text{hidden}} \times d_{\text{hidden}}}. \) In the sequel, we will omit the subscript \(i\).

2.2 Sparsity and Non-symmetry

Existing work (Tsai et al., 2020) shows that MHA and FFN are the most time consuming modules in transformer architecture, which contributes more than 95% FLOPS. Therefore, we investigate the characteristics of MHA and FFN in each transformer layer to see if we can prune the architecture into a compact size. In MHA, we define the model parameters \(\mathbf{W}_q^i, \mathbf{W}_k^i \) and \(\mathbf{W}_v^i\) as functional mappings and investigate the distribution of spectral norm across different heads and layers. For example, we fine-tune RoBERTa model (Liu et al., 2019) on the Stanford Sentiment Treebank.
Binary task (Socher et al., 2013). Then we extract weights from the MHA module as shown in Fig. 2. The heatmap reveals that, the spectral norm of $W^q_l$, $W^k_l$ and $W^v_l$ varies across different heads and different layers. We hypothesis that this phenomenon is due to different token distribution of input instances across different tasks. It provides the possibility to remove the task-specific heads with smaller spectral norm. Furthermore, we investigate the sparsity of the output from FFN across all layers. In short, there are two linear layers in each transformer layer $W_{fc1}$ and $W_{fc2}$. Usually, activation (e.g. GeLU/ReLU) is applied to the output from Y. We explore the distribution shift in both classification task (GeLU is applied) and translation task (ReLU is applied). Fig. 3 illustrates the distribution change of the output from $W_{fc1}$ before and after activation. For example, in the translation task, we observe more than 97% values after ReLU activation are zero. This implies corresponding linear projections from $W_{fc1}$ for these negative outcome values are redundant and can be further removed.

3 Methodologies

In this section, we present the main algorithmic component of AdaPrune, a sparse regularization-based structure pruning strategy. It compresses the transformer architecture through attention heads pruning and block pruning in feed forward networks, and takes three steps to compress the transformer architecture. The main procedure is listed in Algorithm 1.

Fine-tuning with Sparse Regularization: We first add sparse regularization to task loss (e.g. cross-entropy loss in classification task) during the fine-tuning stage. Suppose we have $L$ layers, the loss is defined as

$$\text{Loss} = L_{\text{task}} + \lambda \left( \sum_{i=1}^{L} \|\text{MHA}_i\| + \|\text{FFN}_i\| \right)$$

$$\|\text{MHA}_i\| = \sum_{i=1}^{H} \|W^q_i\| + \|W^k_i\| + \|W^v_i\|$$

In the above, sparse regularization is added across each layer and each head with a single scaling factor $\lambda$. Note that we partition the matrix $W_{fc1}$ and $W_{fc2}$ of FFN into $M \times N$ blocks, and calculate the norm of each block separately. Then the norm of FFN in transformer layer $l$ is defined as

$$\|\text{FFN}_l\| = \sum_{i=1}^{M} \sum_{j=1}^{N} \|W_{fc1i,j}\| + \|W_{fc2i,j}\|$$

Norm-based Pruning: After fine-tuning with sparse regularization, we use the norm $\|W^q_l\| + \|W^k_l\| + \|W^v_l\|$ of each head as the proxy for head importance, and remove the unimportant heads based on the pre-defined norm threshold $\epsilon_m$. Similarly, importance of blocks in feed forward networks are calculated based on the norm of corresponding blocks, and unimportant blocks are removed based on threshold $\epsilon_f$. As a result, we could control the strength of pruning by simply tuning the thresholds $\epsilon_m$ and $\epsilon_f$. We will store the pruning information in set $M_T$.

Fine-tune Pruned Model: Based on head and block index stored in $M_T$, we will prune pre-trained language model $M$ to obtain a smaller pre-trained language model $M_{pruned}$. Then we further fine-tune the pruned model $M_{pruned}$ on the same target tasks.

4 Experiments

In our experiments, we use the RoBERTa-base model (Liu et al., 2019) and evaluate proposed method on various tasks with three pruning settings: MHA pruning, FFN pruning and combination of the MHA & FFN pruning. For each setting, we have explored two pruning percentages (33% or 50%), which means that we remove 33% or 50% parameters in the corresponding module. We evaluate the performance of AdaPrune in the GLUE benchmark (Wang et al., 2019). We also compare our proposed method with two baseline methods:

```
Algorithm 1 AdaPrune

1: procedure ADAPRUNE($D_{train}, M$)
2: $M_T = \text{AdaTrain}(D_{train}, M, \lambda)$
3: $M_{pruned} = \text{Prune}(M_T, M, \epsilon_m, \epsilon_f)$
4: for epoch in epochs do
5:     for $X_{train_i}, Y_{train_i}$ in $D_{train}$ do
6:         pred = $M_{pruned}(X_{train_i})$
7:         loss = loss(pred, $Y_{train_i}$)
8:     Backpropagation(loss)
9: end for
10: end for
11: return $M_T^{pruned}$
12: end procedure
```
### Table 1: Results of the RoBERTa-base model with different pruning methods. The pruning configuration includes the number of heads in MHA of each layer and the dimension of $W_{fc1}$ in the FFN. For example, $H=12$ means there are 12 heads in each transformer layer and $D=3072$ indicate the two feed forward layers are of size $[768, 3072]$ and $[3072, 768]$.

<table>
<thead>
<tr>
<th>Method</th>
<th>IMDB (Acc)</th>
<th>SST-2 (Acc)</th>
<th>CoLA (Acc)</th>
<th>MNLI (Acc)</th>
<th>QNLI (Acc)</th>
<th>QQP (Acc)</th>
<th>STS (Pear./Sp.)</th>
<th>AVG</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Standard Prune</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$H=8, D=3072$</td>
<td>92.7</td>
<td>93.5</td>
<td>83.6</td>
<td>86.2</td>
<td>91.7</td>
<td>91.5</td>
<td>89.4/89.1</td>
<td>89.7</td>
</tr>
<tr>
<td>$H=6, D=3072$</td>
<td>91.9</td>
<td>93.2</td>
<td>82.2</td>
<td>85.7</td>
<td>90.9</td>
<td>91.4</td>
<td>88.8/88.5</td>
<td>89.1</td>
</tr>
<tr>
<td>$H=12, D=2048$</td>
<td>91.8</td>
<td>93.1</td>
<td>78.1</td>
<td>84.6</td>
<td>91.8</td>
<td>91.3</td>
<td>89.5/89.1</td>
<td>88.9</td>
</tr>
<tr>
<td>$H=12, D=1536$</td>
<td>91.2</td>
<td>92.1</td>
<td>73.6</td>
<td>85.0</td>
<td>90.1</td>
<td>91.1</td>
<td>89.6/89.3</td>
<td>87.8</td>
</tr>
<tr>
<td>$H=8, D=2048$</td>
<td>90.8</td>
<td>92.3</td>
<td>77.5</td>
<td>84.7</td>
<td>90.0</td>
<td>91.1</td>
<td>88.1/87.8</td>
<td>87.8</td>
</tr>
<tr>
<td>$H=6, D=1536$</td>
<td>89.3</td>
<td>90.6</td>
<td>71.5</td>
<td>83.0</td>
<td>87.4</td>
<td>90.6</td>
<td>83.7/83.8</td>
<td>85.0</td>
</tr>
<tr>
<td><strong>AdaPrune</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$H=8, D=3072$</td>
<td>93.0</td>
<td>91.5</td>
<td>83.9</td>
<td>86.8</td>
<td>92.1</td>
<td>91.6</td>
<td>89.8/89.5</td>
<td>90.2</td>
</tr>
<tr>
<td>$H=6, D=3072$</td>
<td>92.0</td>
<td>94.5</td>
<td>82.9</td>
<td>86.0</td>
<td>91.3</td>
<td>91.5</td>
<td>89.2/88.9</td>
<td>89.5</td>
</tr>
<tr>
<td>$H=12, D=2048$</td>
<td>92.9</td>
<td>94.5</td>
<td>83.2</td>
<td>87.2</td>
<td>92.6</td>
<td>91.8</td>
<td>89.6/89.4</td>
<td>90.1</td>
</tr>
<tr>
<td>$H=12, D=1536$</td>
<td>91.2</td>
<td>93.5</td>
<td>77.9</td>
<td>85.8</td>
<td>90.9</td>
<td>91.3</td>
<td>88.7/88.4</td>
<td>88.5</td>
</tr>
<tr>
<td>$H=8, D=2048$</td>
<td>91.3</td>
<td>93.8</td>
<td>80.7</td>
<td>86.0</td>
<td>91.6</td>
<td>91.6</td>
<td>86.4/86.2</td>
<td>88.5</td>
</tr>
<tr>
<td>$H=6, D=1536$</td>
<td>88.9</td>
<td>93.0</td>
<td>77.1</td>
<td>84.8</td>
<td>90.4</td>
<td>91.4</td>
<td>85.4/85.2</td>
<td>87.0</td>
</tr>
<tr>
<td>DistilBERT (Sanh et al., 2020)</td>
<td>92.8</td>
<td>91.3</td>
<td>51.3</td>
<td>82.2</td>
<td>89.2</td>
<td>88.5</td>
<td>84.5/85.0</td>
<td>83.1</td>
</tr>
<tr>
<td>RoBERTa (Liu et al., 2019)</td>
<td>93.7</td>
<td>95.1</td>
<td>83.1</td>
<td>87.5</td>
<td>93</td>
<td>91.9</td>
<td>91.3/90.9</td>
<td>90.8</td>
</tr>
</tbody>
</table>

Table 2: Summary of efficiency improvement with different pruning configurations in MHA and FFN. The first line is the baseline coming from the RoBERTa base model without pruning. Since GFLOPs depend on the size of the input feature, GFLOPs in the table are calculated based on the input tensor with batch size as 1, sequence length as 512 and embedding size as 768.

<table>
<thead>
<tr>
<th>Setup</th>
<th>GFLOPs</th>
<th>Time (ms)</th>
<th>Savings</th>
</tr>
</thead>
<tbody>
<tr>
<td>H=12, D=3072</td>
<td>96.6</td>
<td>809</td>
<td></td>
</tr>
<tr>
<td>H=8, D=3072</td>
<td>83.6</td>
<td>685</td>
<td>15.3%</td>
</tr>
<tr>
<td>H=6, D=3072</td>
<td>77.3</td>
<td>613</td>
<td>24.2%</td>
</tr>
<tr>
<td>H=12, D=2048</td>
<td>77.2</td>
<td>653</td>
<td>19.3%</td>
</tr>
<tr>
<td>H=12, D=1536</td>
<td>67.6</td>
<td>570</td>
<td>29.5%</td>
</tr>
<tr>
<td>H=8, D=2048</td>
<td>64.4</td>
<td>553</td>
<td>31.6%</td>
</tr>
<tr>
<td>H=6, D=1536</td>
<td>48.3</td>
<td>420</td>
<td>51.9%</td>
</tr>
</tbody>
</table>

**5 Conclusion**

In this work, we study the AdaPrune strategy with the goal of achieving both efficiency and effectiveness. We start with analysis on MHA and FFN modules to show our observation of weights distribution on these modules. Based on observation, we find that heads in MHA have different weights scale and we take that as indicator for importance. Additionally, we find output value from FC1 are cropped sharply at zero due to GeLU activation, which implies the unnecessary calculation exists in feed forward layers. Through experiments of applying AdaPrune to RoBERTa model, we prove the decent performance of AdaPrune against Standard Prune strategy.

Further, we benchmark the efficiency improvement for the prune model. We harvest the inference time based on Intel(R) Xeon(R) Platinum 8259CL CPU@2.50GHz for all kinds of parameter settings used in experiments. For each pruning setup, we pass the same test example with sequence length 512 as input for models and take the average inference time on 50 runs. The Table 2 shows the speed up varies from 15.36% to 48.09% with different prune setup.
References


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.


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.


Changhan Wang, Yun Tang, Xutai Ma, Anne Wu, Dmytro Okhonko, and Juan Pino. 2020a. fairseq s2t: Fast speech-to-text modeling with fairseq.


Zhanghao Wu, Zhijian Liu, Ji Lin, Yujun Lin, and Song Han. 2020. Lite transformer with long-short range attention.