SPARSE TOKEN TRANSFORMERS WITH ATTENTION BACK TRACKING

Anonymous authors
Paper under double-blind review

ABSTRACT

Despite the success of Transformers in various applications from text, vision, and speech domains, they are yet to become standard architectures for mobile and edge device applications due to their heavy memory and computational requirements. While there exist many different approaches to reduce the complexities of the Transformers, such as the pruning of the weights/attentions/tokens, quantization, and distillation, we focus on token pruning, which reduces not only the complexity of the attention operations, but also the linear layers, which have non-negligible computational costs. However, previous token pruning approaches often remove tokens during the feed-forward stage without consideration of their impact on later layers' attentions, which has a potential risk of dropping out important tokens for the given task. To tackle this issue, we propose an attention back-tracking method that tracks the importance of each attention in a Transformer architecture from the outputs to the inputs, to preserve the tokens that have a large impact on the final predictions. We experimentally validate the effectiveness of the method on both NLP and CV benchmarks, using Transformer architectures for both domains, and the results show that the proposed attention back-tracking allows the model to better retain the full models' performance even at high sparsity rates, significantly outperforming all baselines. Qualitative analysis of the examples further shows that our method does preserve semantically meaningful tokens.

1 INTRODUCTION

Transformers have achieved huge success in various application domains such as natural language processing (NLP) and computer vision (CV), obtaining state-of-the-art performances on a variety of tasks, and are now considered the de facto standard architectures for a number of domains. However, Transformers require over a few tera-flops per entry (Devlin et al., 2019; Dosovitskiy et al., 2021b) to compute, which is orders of magnitude larger than the computational cost for previous CNN and RNN architectures (Tan & Le, 2019; Bahdanau et al., 2015). To reduce such computational burdens of Transformer models, previous works explored model compression methods such as distillation (Wang et al., 2020; Jiao et al., 2020), quantization (Han et al., 2015; Frankle & Carbin, 2018), and pruning (Zaheer et al., 2020; Guo et al., 2019; Kim et al., 2022; Rao et al., 2021; Wang et al., 2021). Pruning approaches for Transformers mostly aim to remove unnecessary model weights (Han et al., 2015; Frankle & Carbin, 2018), or the attentions (Zaheer et al., 2020), which could achieve a linear reduction in complexity. Token pruning can reduce the complexity of attentions and fully connected layers simultaneously by removing less relevant tokens for the target task (Kim et al., 2022; Goyal et al., 2020; He et al., 2021). How can we decide which tokens to prune then? Previous studies either compute the importance score of each input token as the average attention scores (Goyal et al., 2020; Kim et al., 2022) (Figure 1), or learn the importance score of each token with an additional neural network (NN) at each layer (Rao et al., 2021; He et al., 2021).

However, previous token pruning methods have the following problem: they prune the tokens in the input sequence without explicitly evaluating the importance of each token on the **final sequence representation** and **prediction tasks**. This is because all existing works (Kim et al., 2022; Wang et al., 2021; Goyal et al., 2020; Rao et al., 2021; He et al., 2021) compute the importance score of an input token at each attention layer while doing the forward pass from the input to the final output (feed-forward token pruning, Figure 1). Thus, they may prune out important tokens at earlier layers that are important for the final representation as well as the task loss. For example, in Figure 1, the

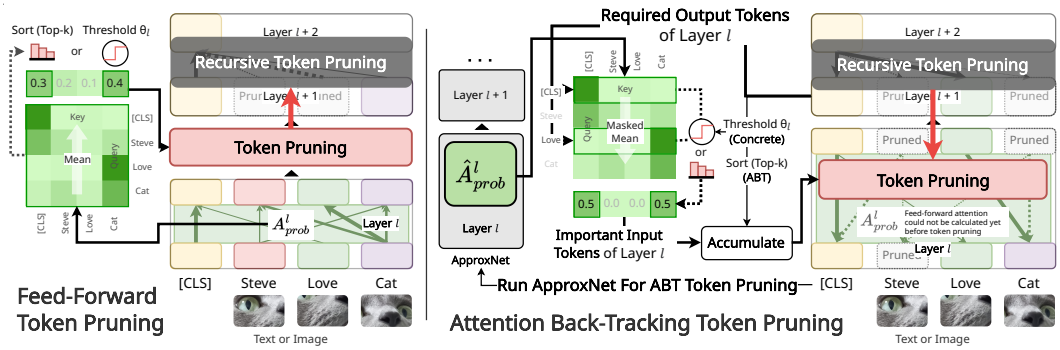


Figure 1: **Concepts.** Comparison of token pruning method between feed-forward and attention back-tracking. The thickness of arrows represents the weight of attention probability. The left side is token pruning in a feed-forward pass. The right side is token pruning with attention back-tracking.

token `Love` in the layer $l+1$ is pruned by the feed-forward method, although the token has a high attention probability in the representation at the final layer $l+2$ that is used by the classification task.

To tackle such a limitation of conventional feed-forward token pruning methods, we propose an **Attention back-tracking** method for computing the importance score of each input token by their importance on the required output tokens (e.g.: sequence representation token, in last layer) and the task performance. As illustrated in Figure 1 (right), we take a backward pass from the last layers' token representations to input tokens, pruning the tokens by their importance score of each input token by accumulating its attention score. By doing so, we are able to better select and keep the important tokens that will minimally affect the output and the prediction. We name our novel token pruning method as **S**parse **T**oken **T**ransformer with **A**ttention **B**ack-**T**racking (**STTABT**).

However, one challenge here is that such backtracking requires us to know the attention score of each token before feed-forwarding the input sequence to the model. To handle the issue, we introduce a lightweight attention approximation network trained with knowledge distillation called **ApproxNet**. Moreover, to actually prune the tokens, we need to decide which tokens to be retained based on the importance scores. The top-k method, by design, can select a predefined number of tokens (Goyal et al., 2020), but requires sorting the tokens by their importance scores, which is not differentiable. The thresholding method, which prunes the tokens with scores under the threshold value (Kim et al., 2022) does not require sorting, but setting the threshold can be tricky since it is difficult to know how many tokens will be remained in advance with the given threshold. To remedy such issues in both methods, we propose a learnable and smooth threshold function named **Concrete masking**, inspired by Concrete Dropout (Gal et al., 2017). Specifically, we jointly train the threshold value for each layer and importance score which is computed by attention back-tracking, with the task objective to find the thresholds that can minimize the task loss.

Our method is generally applicable to Transformer encoders for any domains, and we validate our method on text and image classification tasks. For text classification, we validated the proposed token pruning methods on GLUE (Wang et al., 2019) benchmark with BERT (Devlin et al., 2019). For image classification, we validate it on ImageNet-1K (Deng et al., 2009) benchmark with DeiT (Touvron et al., 2021). The experimental results show that our model works surprisingly well even at very high pruning rates, for which baselines show significant performance degeneration. For example, on a GLUE benchmark task (QQP), our method obtains 45.54% increased token sparsity (17.15% to 9.34%) compared to the baseline with the same accuracy. Moreover, our method obtains very high token sparsity (18.7%), keeping only half of the tokens retained by DynamicViT (37.5%), with only 0.8% accuracy loss on ImageNet-1K benchmark. Our contributions can be summarized as follows:

- We propose a novel token pruning method for Transformers based on attention back-tracking, which considers the importance of each token on the final representations as well as the task loss based on the approximated attention from the distilled model.
- We propose Concrete masking, which automatically learns the pruning rate for each layer with Concrete dropout. (Gal et al., 2017).
- Our token pruning is generally applicable to any Transformers, and we validated it on trained Transformers from both NLP (BERT) and CV (DeiT) domains, on GLUE and ImageNet benchmark,

whose results show that it achieves a significantly better accuracy-computation tradeoff. (Kim et al., 2022; Rao et al., 2021).

2 Related Work

Attention Pruning for Transformers Since the literature on the general neural network compression techniques such as quantization and weight pruning is vast, we discuss compression techniques specifically targeting Transformers (Vaswani et al., 2017). Due to the heavy computational cost of the self-attention, several recent works have focused to reduce its time complexity from $\mathcal{O}(N^2)$ to about $\mathcal{O}(N)$ given the N length of the sequence by pruning the attentions in the self-attention layers. BigBird (Zaheer et al., 2020) is one such method, and it utilizes set of sparse attentions, that are both local and random, while using few global attentions, to deal with longer sequences. Similarly, Star-Transformer (Guo et al., 2019) leverages the concept of a virtual relay node and reduces the number of fully-connected attentions with the radial and ring connections. However, these prior works do not consider the content of the input sequence, as they make use of static sparsity patterns. To remedy this, some recent works aim to dynamically sparsify attentions of the Transformer considering the input data. Specifically, Reformer (Kitaev et al., 2020) divides the tokens into several groups through Locally Sensitive Hashing (LSH) and calculates attentions by sparsely connecting them. Routing Transformer (Roy et al., 2021) uses the k-means clustering algorithm to cluster input tokens.

Token Pruning for Transformers However, in most datasets used in Natural Language Processing (NLP), such as text classification (Wang et al., 2019), input sequences with with excessively long lengths are quite rare, as the length of most sequences is limited to 256 tokens (Kim et al., 2022). Moreover, fully connected layers require significant computations that are non-negligible, although they have been relatively overlooked. Therefore, some recent works (Kim et al., 2022; Goyal et al., 2020) focus on reducing the number of tokens by selecting the most important tokens for the given task, to reduce the computational cost of both linear and self-attention layers. Notably, PoWER-BERT (Goyal et al., 2020) progressively eliminates tokens based on their importance scores, computed from the attentions at each layer of the Transformer encoder, and Learned Token Pruning (LTP) (Kim et al., 2022) aims to learn to threshold the importance scores to prune the token. However, both methods prune the tokens during the feed-forward pass, assuming that important input tokens are also important output tokens. Therefore, they may prune out tokens that are important for the output vector at earlier layers. Compared to them, our method STTABT can consider the impact of each token on the output vector and the prediction task by attention back-tracking, thanks to our novel lightweight attention approximation network.

3 Sparse Token Transformer with Attention Back Tracking

We start by briefly describing the Transformer model that our method focuses on. Then, we propose two components for our Sparse Token Transformer with Attention Back Tracking (STTABT): *Attention Back Tracking (ABT)* which obtains the importance score for each token with an attention approximation network and *Concrete masking* for dynamically learning the token-wise threshold for each input. STTABT is a post-hoc token pruning method and assumes that the target Transformer encoder has previously been optimized on the training dataset for the target task. Our goal is to lower the inference cost of the given pretrained Transformer by pruning the tokens at each layer.

3.1 Transformer

For concepts used in later sections, we define the self-attention layer — the fundamental element of the Transformer architecture (Vaswani et al., 2017) — in this section. Let us denote the attention score and probability at l-th Transformer layer as follows:

$$\boldsymbol{A}^l_{score} = \frac{\boldsymbol{Q}^l {\boldsymbol{K}^l}^\top}{\sqrt{d_k}}, \qquad {\boldsymbol{A}^l}_{prob} = \text{softmax}({\boldsymbol{A}^l}_{score}), \tag{1}$$

where $\boldsymbol{Q}^l \in \mathbb{R}^{L \times d_k}$ is the query matrix, $\boldsymbol{K}^l \in \mathbb{R}^{L \times d_k}$ is the key matrix, d_k is the dimension of query and key matrix, and $l \in [1, L]$ is the layer index.

3.2 ATTENTION BACK-TRACKING (ABT)

We now describe the details of our method, Attention Back Tracking (ABT). The fundamental idea of ABT is to prune the tokens based on their importance scores that measure their impacts on the final sequence representation for task prediction, by back-tracking the attentions. Specifically, as shown in Figure 1 (right), we start with the tokens with higher attention probabilities from the last layer, and backtrack their accumulated attention probabilities to the first layer of the Transformer encoder consisting of L layers with regards to the token for the sequence classification task (specifically, [CLS] token). However, to apply such a technique into practice, we must feed-forward the input sequence through all of the Transformer layers in order to compute attention scores beforehand. Thus a straightforward implementation of the technique could be quite inefficient because it will require us to perform forward pass of the model twice for each input.

Attention Approximation Network To alleviate the cost, we use an *attention approximation network* (ApproxNet), a distilled Transformer encoder that approximates the attention probabilities of the original network with a significantly less number of parameters. We perform knowledge distillation based on the distillation loss from TinyBERT (Jiao et al., 2020) to build the ApproxNet. Specifically, we leverage embedding based distillation loss $\mathcal{L}_{embd}$, hidden state based distillation loss $\mathcal{L}_{hidn}$, attention-based distillation loss $\mathcal{L}_{attn}$, and prediction-layer distillation loss $\mathcal{L}_{pred}$ for the distillation. Especially, for the attention-based distillation, we use the KL divergence loss from MiniLM (Wang et al., 2020) instead of the mean squared error loss as follows:

$$\mathcal{L}_{attn} = \frac{1}{HT^2L} \sum_{l=1}^{L} \sum_{h=1}^{H} \sum_{j=1}^{T} \sum_{i=1}^{T} (\boldsymbol{A}^l_{prob} * (\log \boldsymbol{A}^l_{prob} - \log \hat{\boldsymbol{A}}^l{}_{prob}))_{h,j,i}, \tag{2}$$

where $\boldsymbol{A}^l_{prob} \in \mathbb{R}^{H \times T \times T}$ denotes l-th layer attention probability of the original network (teacher), $\hat{\boldsymbol{A}}^l_{prob} \in \mathbb{R}^{H \times T \times T}$ denotes l-th layer attention probability of the ApproxNet (student), H is the number of heads for multi-head self-attention layers, T is the length of input sequence, and L is the number of layers in the Transformer network.

Algorithm 1: Update token mask from the output token indices

```
def UpdateTokenMaskOfLayer(Â^l_prob, i^{l+1}, s^{l+1}, p, k)
    // Reduce attention heads and gather required
       output tokens' attention probabilities
       with previous importance score.
    A ← gather(mean(Â^l_prob, dim=0), i^{l+1}, dim=0)
    a ← mean(diag(s^{l+1})A * p + A * (1 − p), dim=0)
    i^l ← unique(cat(i^{l+1}, topk(a, k)))
    s^l ← gather(a, i^l)
    s^l ← s^l / sum(s^l)
    return i^l, s^l
```

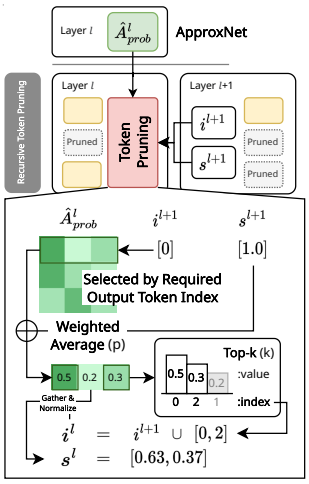


Figure 2: Computation flow of update token mask of whole Transformer encoder layers with attention back-tracking by Algorithm 1.

Token Pruning with Attention Back-tracking on Approximated Attention Then, using the approximated attentions from the ApproxNet, we can now prune the tokens at each layer of the main network. In Algorithm 1, we define the pruning function UpdateTokenMaskOfLayer, which updates the mask at each Transformer layer l for token pruning. The input arguments of the pruning function are the approximated attention probability $\hat{\boldsymbol{A}}^l_{prob}$ of l-th layer from the ApproxNet, the token mask $\boldsymbol{i}^{l+1} \in \mathbb{Z}^{o_{l+1}}$, and the importance scores of the tokens $\boldsymbol{s}^{l+1} \in \mathbb{R}^{o_{l+1}}$ from the upper layer $l+1$, where $l = 1$ is the first layer of the Transformer model and o_{l+1} is the number of selected tokens in $(l+1)$-th layer. $\boldsymbol{i}^L$ is initialized with indices of required tokens, which is [CLS] token in the classification task. $\boldsymbol{s}^L \in \mathbb{R}^{o_T}$ is initialized as the one vector $\mathbf{1}/o_T$ where o_t is the number of output tokens, which is generally a single token. $p \in (0, 1) \subset \mathbb{R}$ determines how much importance scores should be propagated from the upper layer. If $p = 0$, the importance scores from the upper layer will

be completely ignored when pruning the tokens in l-th layer. $k \in \mathbb{Z}$ determines how many tokens should be selected in case of using top-k selection method, and we could easily change the number of tokens to retain at each layer, by adjusting k. The function returns the index of remaining input tokens at l-th layer and their importance scores given those arguments.

3.3 Concrete Masking

The top-k tokens at each layer of the Transformer model may still be selected using the previous scheme's static value of k, but in order to regulate the pruning based on the input, a mechanism for dynamically altering the pruning ratio is also necessary. Therefore, previous work, such as LTP (Kim et al., 2022) proposed to learn a layer-wise threshold for token pruning. In detail, during fine-tuning, LTP keeps tokens that have larger scores than the threshold, which is jointly optimized with the network parameters. However, it can be challenging to control the number of tokens left at each layer when the importance score is relatively sparse and the majority of score values are close to zero or very small values. To address the problem, we suggest a novel dynamic thresholding method named as *Concrete Masking*, inspired by the learnable dropout function idea in Concrete dropout (Gal et al., 2017).

Let us consider a single row of one of multi-head attention score matrix $\boldsymbol{a}_t^l = \boldsymbol{A}_{score}{}_{h,t}^{l} \in \mathbb{R}^T$ in this section for brevity, where h is the head index and i is the row index. We further assume that the attention scores follow a normal distribution. We transform $\boldsymbol{a}_t$ into the uniform distribution where $\boldsymbol{a}_{t,i} \in [0, 1]$ so that the importance score becomes uniformly distributed. Formally,

$$\tilde{\boldsymbol{a}}^l = \Phi(\frac{\boldsymbol{a}^l - \mu_{\boldsymbol{a}}}{\sigma_{\boldsymbol{a}}}), \qquad \tilde{\boldsymbol{A}}^l = \begin{bmatrix} — & \tilde{\boldsymbol{a}}_1^l & — \\ & \cdots & \\ — & \tilde{\boldsymbol{a}}_T^l & — \end{bmatrix} \tag{3}$$

where Φ is the cumulative distribution function of the standard normal distribution, $\mu_{\boldsymbol{a}} = \sum_{t=1}^{T} \boldsymbol{a}_t^l / T$ is the mean of $\boldsymbol{a}^l$, and $\sigma_{\boldsymbol{a}} = \sqrt{\sum_{t=1}^{T} (\boldsymbol{a}_t^l - \mu_{\boldsymbol{a}})^2 / T}$ is the standard deviation of the $\boldsymbol{a}^l$. Then, we apply above transformation to all the rows of the predicted attention score matrix from ApproxNet $\hat{\boldsymbol{A}}_{score}^l$ to acquire $\tilde{\boldsymbol{A}}_l$ where all of rows are uniformly distributed. Similar to the line 3 of Algorithm 1, we then propagate the transformed importance score $\tilde{\boldsymbol{a}}$ as follows:

$$\boldsymbol{b}^l = \frac{\sum_{t=1}^{T} \left(\left(p \mathrm{diag}(\boldsymbol{s}^{l+1}) + 1 - p \right) \mathrm{diag}(\boldsymbol{m}^{l+1}) \tilde{\boldsymbol{A}}^l \right)_{t,:}}{\sum \boldsymbol{m}^{l+1}}, \quad \boldsymbol{s}^l = \left(1 - \epsilon \frac{\boldsymbol{b}^l}{\max \boldsymbol{b}^l} + \epsilon \right) \tag{4}$$

where the token mask $\boldsymbol{m}^{l+1} \in (0, 1)^T \subset \mathbb{R}^T$ is the Concrete mask vector. Then, the updated Concrete token mask $\boldsymbol{m}^t$ is defined as follows (Gal et al., 2017):

$$\boldsymbol{m}^l = \max \left(\boldsymbol{m}^{l+1}, \mathrm{sigmoid} \left(\frac{\log p_{\theta^l} - \log(1 - p_{\theta^l}) + \log \boldsymbol{s}^l - \log(1 - \boldsymbol{s}^l)}{\tau} \right) \right) \tag{5}$$

where *sigmoid* is the sigmoid function, τ is the temperature hyperparameter, and $p_{\theta^l} = \mathrm{sigmoid}(\theta^l)$ where θ^l is a learnable parameter and initialized with θ_{init}. During training, the model learn the proper value of θ^l with the soft mask function applied to each layer. After training for reproducing the base accuracy with sufficient epochs, we change the soft mask into a hard mask, then fine-tune the parameters to fit the hard mask as in LTP (Kim et al., 2022). For hard masking, we threshold the soft Concrete token mask $\boldsymbol{m}^l$ using a set threshold, usually 0.5.

To train our Concrete masking mechanism, we need regularization terms in addition to the task loss. First of all, we add the regularization term $\mathcal{L}_p$, which prevents θ^l from deviating too far from the initialized value and another regularization term $\mathcal{L}_{mask}$ so that the average token retention ratio stay around $p_{\theta_{init}}$ as follows:

$$\mathcal{L}_p = \lambda_p \sum_{l=1}^{L} (\theta^l - \theta_{init}^l)^2, \quad \mathcal{L}_{mask} = \lambda_{mask} \left(\left(\frac{1}{TL} \sum_{l=1}^{L} \sum_{t=1}^{T} \boldsymbol{m}_t^l \right) - p_{\theta_{init}} \right)^2 \tag{6}$$

where λ_p and λ_{mask} are hyperparameters.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

To show the general applicability of our method, STTABT, we validate it on both Natural Language Processing (NLP) and vision tasks. In order to evaluate our method, we primarily compare the performance of our method against those of the baselines at the same **token retention ratios**, which is a metric that measures how many tokens are retrained after pruning. We use the average token retention ratio values across all the layers of the target Transformers. To accurately measure the efficiency of our method, even considering the overhead for computing the ApproxNet for attention approximation, we also report the performance as a function of FLOPs in Section 5.

4.1.1 PRUNING TRANSFORMER-BASED TRAINED LANGUAGE MODELS

Dataset and metric. We use nine datasets from GLUE (Wang et al., 2019) benchmark for the text classification and use the BERT$_{base}$ (Devlin et al., 2019) as the base model. For evaluation metric, we use the metrics from the original paper (Wang et al., 2019). **Baselines.** We compare STTABT against two notable token pruning baselines – the manual top-k method and Learned Token Pruning (LTP) (Kim et al., 2022). The manual top-k method is the feed-forward token pruning which keeps the k_l tokens with highest importance scores for each l-th layer. LTP is a baseline with a learnable threshold, which requires to fine-tune the whole parameters to learn the appropriate threshold value for each input. We match the training settings of STTABT and LTP, then follow the hyperparameter search strategies described in (Kim et al., 2022). For the comparison against LTP, we use STTABT with Concrete masking strategy (§3.3), while use top-k pruning with our method (§3.2) when comparing against the manual top-k baseline.

4.1.2 PRUNING VISION TRANSFORMERS

Dataset and metric. We use ImageNet-1k (Deng et al., 2009) for the image classification experiment, using the Data-efficient Image Transformer (DeiT) (Dosovitskiy et al., 2021a; Touvron et al., 2021) as the base Vision Transformer model. **Baselines.** We use DynamicViT (Rao et al., 2021) as the primary baseline. We follow the experimental setups, including data preprocessing and training pipeline of DynamicViT for all experiments on the image classification task.

4.2 EXPERIMENTAL RESULTS ON NLP TASKS

Attention back-tracking. We measure the performance of our attention back-tracking (ABT) method with the ApproxNet (§3.2). We experiment with different k in Algorithm 1 to vary the average token retention ratios. The experimental results in Figure 3 show that the our method significantly outperforms the manual top-k with the same average token retention ratio. Furthermore, we find that using the ApproxNet (blue lines) obtains similar performance to using the attention from the full network (green lines). These results support that our attention backtracking strategy can identify more important tokens than the feed-forward pruning methods do, which compute the importance scores and perform pruning with forward passes.

Concrete masking. In Figure 3, we further present the performance evaluations with or without the Concrete masking, to examine its effect. One of the advantages of our model with Concrete masking is that the user can flexibly adjust the token retention ratio by changing θ_{init} value. In contrast, the user cannot easily adjust the token retention ratio of LTP since the method does not support the control of the token and relies on regularization on the mask, which is the token retention ratio with a regularization scaling factor. For the lower token retention ratio of LTP, we must increase the scaling factor, then the regularization term becomes significantly bigger than task loss at some point. As a result, the LTP token retention ratio converges to zero or an extremely low ratio which is not appropriate to deal with task loss. For instance, in QNLI, MRPC, CoLA, and SST-2 of Figure 3, LTP baseline cannot cover the low token retention ratio, whereas our method can.

At relatively high token retention ratios around 30-50%, both STTABT with concrete masking and LTP work well, although our method still outperforms LTP in most cases. However, at lower than 30% token retention ratios, our method with Concrete masking show significantly superior performance over LTP, better preserving the accuracy of the full model (Please see the results on MNLI, QNLI,

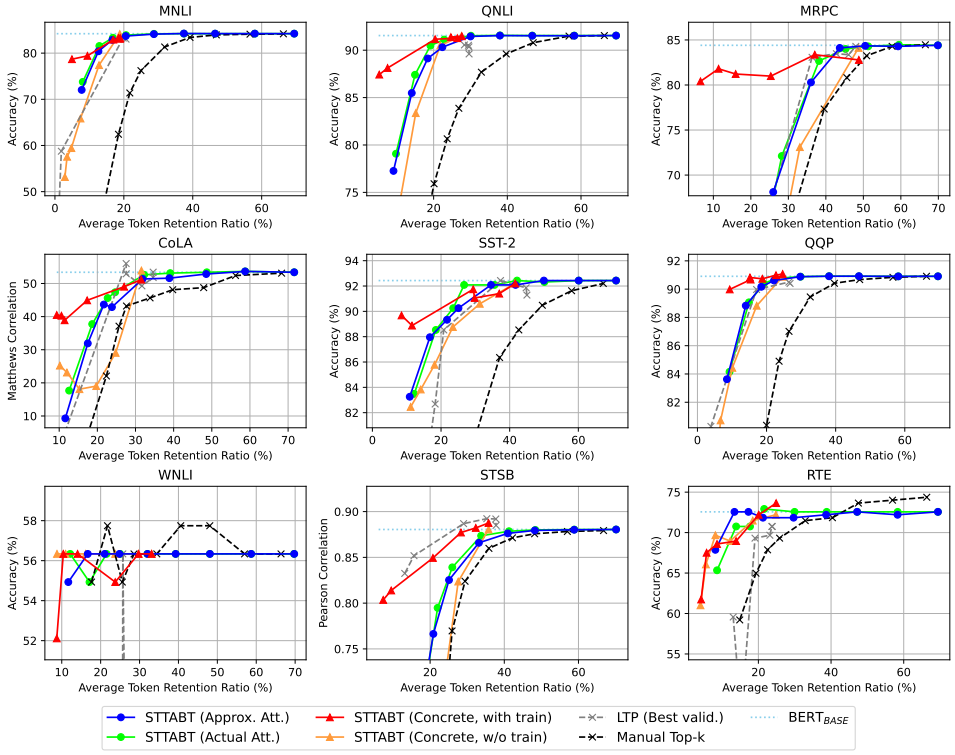


Figure 3: Experimental results with varying average token retention ratio for our method STTABT (by ApproxNet with factor 4) and baselines on GLUE and BERT$_{base}$. Our results with top-k pruning (blue, green) are compared against the manual top-k method (black dot). In contrast, results with Concrete masking (red, orange) are compared to the LTP (grey dot).

MRPC, CoLA, SST-2, QQP, and RTE). To examine which tokens are pruned with our method and LTP, we further visualize the pruning mask at each layer of BERT$_{base}$ in Figure 6. We observe that LTP keeps most of the important tokens, but unimportant tokens as well such as dots in Figure 6. On the other hand, our method with Concrete masking prunes useless tokens such as dots at much earlier layers than LTP, yielding higher sparsity in total.

The Concrete masking method have significantly higher accuracy in most of cases in GLUE benchmark than ABT without Concrete masking and baselines. The Concrete masking method works especially well at high pruning rates, while have competitive performance to baselines at low pruning rates too.

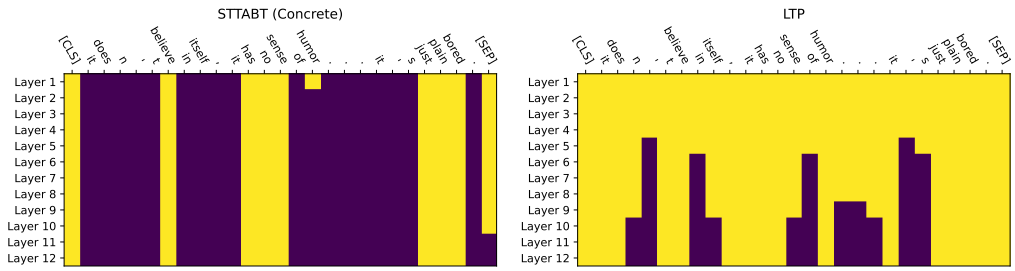


Figure 4: Token pruning visualization between our Concrete masking and LTP layer by layer. Layer l means the input token mask on the layer l. Yellow color indicates the remained input tokens in each layer. The input texts are randomly selected from The Stanford Sentiment Treebank (SST-2) (Wang et al., 2019). Similar accuracy of trained LTP and our Concrete masking models are used ($\approx 89\%$).

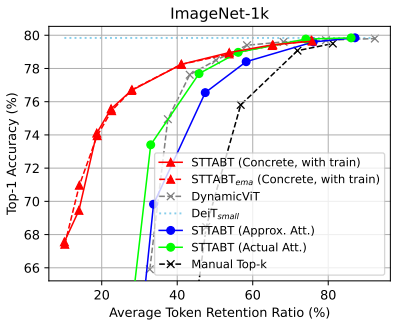


Figure 5: Top-1 accuracy plot varying the average token retention ratio for our method and baselines on ImageNet-1k.

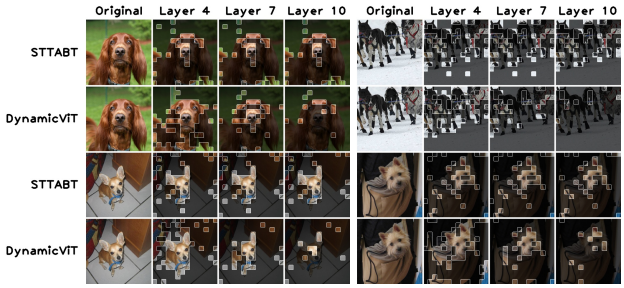


Figure 6: The visualization of token pruning with STTABT (Concrete) and DynamicViT. Models with similar accuracy are used (Ours@f4: 74.1%, DynamicViT: 74.9%).

Table 2: Relative FLOPs of various overhead sources of our methods. The FLOPs results are calculated with a sequence length of 128. We randomly generate token retention ratios for each layer for simulation. Fused MulAdd for matmul is not considered in FLOPs calculation.

	BERT$_{base}$	**ApproxNet**			**Manual Top-k**			**Attention Back-Tracking**			**Concrete Masking**		
		Reduce Factor			Token Retention Ratio			Token Retention Ratio			Token Retention Ratio		
		4	8	16	0.2	0.4	1.0	0.2	0.4	1.0	0.2	0.4	1.0
FLOPs	22.4G	1.5G	436.9M	140.6M	559.5K	1.1M	2.8M	597.4K	1.2M	2.8M	9.2M	18.5M	46.1M
Relative	100%	6.86%	1.95%	0.63%	0.00%	0.00%	0.01%	0.00%	0.01%	0.01%	0.04%	0.08%	0.21%

4.3 Experimental Results on Computer Vision Tasks

The experimental results show that our Attention back-tracking (ABT) performs competitively to DynamicViT even without Concrete masking, on computer vision tasks, and our full model with Concrete masking significantly outperforms Dynamic ViT 5, especially at lower token retention ratios. For example, our method with Concrete masking achieves $74.1\%_{(-5.8\%)}$ accuracy with 18.7% token retention ratio, but DynamicViT requires 37.5% token retention ratio with $74.9\%_{(-5.0\%)}$ accuracy. Moreover, since our ABT requires training ApproxNet only once regardless of the target token retention ratio, it has a huge advantage in training cost than DynamicViT. The DynamicViT used twice more tokens than our Concrete masking, but the accuracy improvement is only 0.8%. Even considering attention approximation overhead, our model performs better than DynamicViT since the overhead is only about $1.95 - 6.86\%$ of the base model, as shown in Table 2.

We visualize the pruned tokens at different layers of the vision Transformers in Figure 6. The visualization shows that our model with Concrete masking concentrates on the important tokens in an earlier layer than DynamicViT. Moreover, our model preserves more important tokens while DynamicViT attends to a larger number of tokens, including unimportant ones. For instance, DynamicViT selects tokens with other attributes at Layer 4, while our model keeps only the tokens related to Chihuahua from the beginning of the model, thanks to the attention back-tracking in Figure 6.

5 Computational Efficiency of STTABT

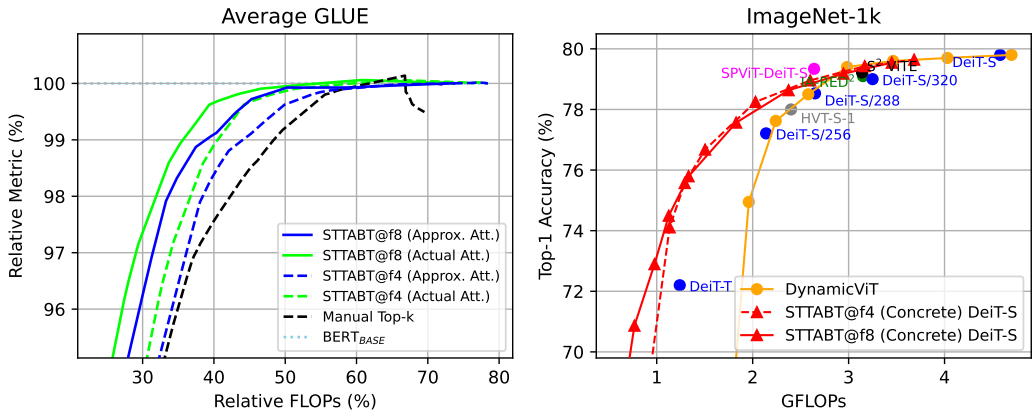
5.1 FLOPs Comparision

For the GLUE benchmark, we compare the attention back-tracking (ABT) method and the manual top-k method with respect to FLOPs. In Figure 7a, we compare the performance with respect to FLOPs over the test set. We find that the ABT performs better than the manual top-k baseline on most datasets.

Table 1: Approximation error of ApproxNet depending on initializing methods.

	Init.	**Random**		**Distilled**	
Dataset	**Factor**	2	4	4	8
Average GLUE	**MSE**	0.0112	0.0112	**0.0013**	0.0018
	KL Div.	1.6430	1.6450	**0.2019**	0.2981

The ABT results look shifted toward the right than the Figure 3, because the computational cost includes overheads such as computing the mask update function (Algorithm 1) and the ApproxNet. We examine the relative overheads of each additional component in Table 2. We see that ApproxNet



(a) (b)

Figure 7: (a) FLOPs comparison between attention back-tracking with top-k and manual top-k baseline on BERT$_{base}$. (b) FLOPs comparison between our method with concrete masking and baselines on ViT. All plotted models except DeiT-T are based from DeiT$_{small}$ for fair comparison. `STTABT@f`x means the attention back-tracking by ApproxNet with factor x.

is the major bottleneck, as the ApproxNet with factor 4 requires about 6.86% of the BERT$_{base}$ FLOPs, which is significantly larger than the manual top-k (0.01%). However, the overhead is more convincing with higher reduction factor of 8 (1.95%) than 4. Therefore in Figure 7a, the ABT results with factor 8 shows better result than 4.

We further compare the performance of STTABT with different reduction factors of the ApproxNet. Table 1 shows that the ApproxNet performs accurately approximations of the actual attention scores with a lower reduction factor, thus results in better preformance. However, when considering the accuracy-efficiency tradeoff, the performance of the model with the reduction factor of 4 are often worse than the one with the reduction factor of 8.

In Figure 7b, we show the accuracy over FLOPs comparison of our model against baselines on ImageNet-1k (Deng et al., 2009), using DeiT$_{small}$ as the base model to prune. The baselines include DynamicViT (Rao et al., 2021) is used as the primary baseline, DeiT$_{small}$ (Touvron et al., 2021), SPViT (He et al., 2021), IA-RED^2 (Pan et al., 2021a), S^2 ViTE (Chen et al., 2021), and HVT (Pan et al., 2021b). We observe that the fine-tuned DeiT$_{small}$ model pruned by our Concrete masking significantly outperforms the fine-tuned DeiT$_{tiny}$ model with the same FLOPs trained from scratch. DynamicViT with DeiT$_{small}$, on the other hand, results in a model with much lower performance than DeiT$_{tiny}$. This is good news for practitioners, as this suggests that we may not need to train Transformers at different scales from scratch, to deploy the model to systems or applications with different computational budgets.

6 Conclusions

In this paper, we propose a sparse token Transformer architecture based on a novel token pruning method, Sparse Token Transformer with Attention Back Tracking (STTABT), with two novel ideas – Attention Back-Tracing (ABT) and Concrete masking. In particular, the ABT method enables our pruning method to back-track the important tokens for classification from the topmost layer to the lower layers, while approximating the attentions using a lightweight approximation network (ApproxNet). Then, we propose a novel thresholding method named as Concrete masking, which adapts the idea from Concrete dropout to learn the appropriate token-wise pruning threshold given the desired amount of tokens to be retained. We evaluate the performance of our method on classification tasks from NLP and CV domains, by pruning tokens from the trained Transformer encoder-based language models and Vision Transformers. Our method shows a better accuracy-efficiency trade-off and yields Transformers with significantly higher accuracy for the same ratio of tokens retained, or at the same FLOPs even when considering the computational overhead from the ApproxNet. Qualitative analysis of the tokens retained by STTABT shows that it preserves relatively more important tokens while pruning out less relevant ones, compared to previous methods.

REPRODUCIBILITY STATEMENT

We will introduce code construction and data collection in this section for reproducibility.

Model Implementation. First we construct our model with PyTorch (Paszke et al., 2019) and Huggingface (Wolf et al., 2020). We modify `bert-base-uncased` from Huggingface Models (Wolf et al., 2020). **ApproxNet.** We create ApproxNet (§3.2, `trainer/glue_base.by`) with the pretrained models. The approximation result is shown in Section A. **ABT.** We implement Algorithm 1 on top of `bert-base-uncased` (Devlin et al., 2019) (`models/sparse_token.py`). **LTP, Concrete masking.** Then, we modify forward path to support LTP (Kim et al., 2022) and fix `ApproxSparseBertModel` to implement Concrete masking (§3.3).

NLP training. The training for NLP tasks, we implemented our own training loop (`trainer/concrete_trainer.py`) and hyper-parameter tuning settings inherits LTP hyper-parameter settings (Kim et al., 2022) (`main/ltp_glue_plot.py`, `main/concrete_glue_plot.py`). We use the GLUE dataset from Huggingface Dataset (Wolf et al., 2020).

Vision training. We start from DynamicViT implementation (Rao et al., 2021) including data loader, training loop, training loss, and evaluation codes (`main/vit_concrete_end2end.py`). Therefore, we did not apply hyper parameter grid search on ViT pipeline. And we remove hard token mask training stage, and we use larger $\lambda_{mask} = 100$ than NLP task, because of lack of padding. Also, we did not use prediction distillation loss for training ApproxNet for ViT. The training dataset is ImageNet-1k (ILSVRC2012) (Deng et al., 2009). We replace the Transformer encoder in ViT model (Dosovitskiy et al., 2021b) into our model code. Our PyTorch model is modified to apply ViT changes from Transformer encoder (Dosovitskiy et al., 2021b).

REFERENCES

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1409.0473.

Tianlong Chen, Yu Cheng, Zhe Gan, Lu Yuan, Lei Zhang, and Zhangyang Wang. Chasing sparsity in vision transformers: An end-to-end exploration. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 19974–19988, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/a61f27ab2165df0e18cc9433bd7f27c5-Abstract.html.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009. ISSN 1063-6919. doi: 10.1109/cvpr.2009.5206848.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pp. 4171–4186. Association for Computational Linguistics, 2019. URL https://doi.org/10.18653/v1/n19-1423.

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. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021a. URL https://openreview.net/forum?id=YicbFdNTTy.

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. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021b. URL https://openreview.net/forum?id=YicbFdNTTy.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Training pruned neural networks. *CoRR*, abs/1803.03635, 2018. URL http://arxiv.org/abs/1803.03635.

Yarin Gal, Jiri Hron, and Alex Kendall. Concrete dropout. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 3581–3590, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/84ddfb34126fc3a48ee38d7044e87276-Abstract.html.

Saurabh Goyal, Anamitra Roy Choudhury, Saurabh Raje, Venkatesan T. Chakaravarthy, Yogish Sabharwal, and Ashish Verma. Power-bert: Accelerating BERT inference via progressive word-vector elimination. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 3690–3699. PMLR, 2020. URL http://proceedings.mlr.press/v119/goyal20a.html.

Qipeng Guo, Xipeng Qiu, Pengfei Liu, Yunfan Shao, Xiangyang Xue, and Zheng Zhang. Star-transformer. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pp. 1315–1325. Association for Computational Linguistics, 2019. URL https://doi.org/10.18653/v1/n19-1133.

Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks. *CoRR*, abs/1506.02626, 2015. URL http://arxiv.org/abs/1506.02626.

Haoyu He, Jing Liu, Zizheng Pan, Jianfei Cai, Jing Zhang, Dacheng Tao, and Bohan Zhuang. Pruning self-attentions into convolutional layers in single path. *arXiv preprint arXiv:2111.11802*, 2021.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. Tinybert: Distilling BERT for natural language understanding. In Trevor Cohn, Yulan He, and Yang Liu (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pp. 4163–4174. Association for Computational Linguistics, 2020. URL https://doi.org/10.18653/v1/2020.findings-emnlp.372.

Sehoon Kim, Sheng Shen, David Thorsley, Amir Gholami, Woosuk Kwon, Joseph Hassoun, and Kurt Keutzer. Learned token pruning for transformers. In Aidong Zhang and Huzefa Rangwala (eds.), *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*, pp. 784–794. ACM, 2022. URL https://doi.org/10.1145/3534678.3539260.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=rkgNKkHtvB.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=Byj72udxe.

Bowen Pan, Rameswar Panda, Yifan Jiang, Zhangyang Wang, Rogerio Feris, and Aude Oliva. Ia-red ^{2}: Interpretability-aware redundancy reduction for vision transformers. *Advances in Neural Information Processing Systems*, 34:24898–24911, 2021a.

Zizheng Pan, Bohan Zhuang, Jing Liu, Haoyu He, and Jianfei Cai. Scalable visual transformers with hierarchical pooling. *CoRR*, abs/2103.10619, 2021b. URL https://arxiv.org/abs/2103.10619.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. Dynamicvit: Efficient vision transformers with dynamic token sparsification. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 13937–13949, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/747d3443e319a22747fbb873e8b2f9f2-Abstract.html.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient content-based sparse attention with routing transformers. *Trans. Assoc. Comput. Linguistics*, 9:53–68, 2021. URL https://doi.org/10.1162/tacl_a_00353.

Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 6105–6114. PMLR, 2019. URL http://proceedings.mlr.press/v97/tan19a.html.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 10347–10357. PMLR, 2021. URL http://proceedings.mlr.press/v139/touvron21a.html.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=rJ4km2R5t7.

Hanrui Wang, Zhekai Zhang, and Song Han. Spatten: Efficient sparse attention architecture with cascade token and head pruning. In *IEEE International Symposium on High-Performance Computer Architecture, HPCA 2021, Seoul, South Korea, February 27 - March 3, 2021*, pp. 97–110. IEEE, 2021. URL https://doi.org/10.1109/HPCA51647.2021.00018.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

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. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, Online, October 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.emnlp-demos.6.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontañón, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: Transformers for longer sequences. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/c8512d142a2d849725f31a9a7a361ab9-Abstract.html.

APPENDIX

A EXPERIMENTAL DETAILS

A.1 DETAILS ON ATTENTION APPROXIMATION NETWORK

	Init.	**Random**		**Interpolate**		**Distilled**	
Dataset	**Factor**	**2**	**4**	**2**	**4**	**4**	**8**
MNLI	**MSE**	0.0087	0.0087	0.0086	0.0086	**0.0013**	0.0018
	KL Div.	1.7503	1.7520	1.7174	1.7097	**0.2393**	0.3423
QNLI	**MSE**	0.0052	0.0052	0.0052	0.0052	**0.0009**	0.0012
	KL Div.	1.8026	1.8078	1.7826	1.7750	**0.2666**	0.3866
MRPC	**MSE**	0.0052	0.0052	0.0052	0.0052	**0.0007**	0.0011
	KL Div.	1.9680	1.9751	1.9484	1.9406	**0.2248**	0.3740
CoLA	**MSE**	0.0339	0.0338	0.0337	0.0334	**0.0032**	0.0042
	KL Div.	1.0845	1.0810	1.0709	1.0629	**0.1223**	0.1644
SST-2	**MSE**	0.0167	0.0167	0.0166	0.0165	**0.0019**	0.0025
	KL Div.	1.4762	1.4744	1.4622	1.4528	**0.1899**	0.2606
QQP	**MSE**	0.0099	0.0099	0.0099	0.0098	**0.0014**	0.0020
	KL Div.	1.5960	1.6020	1.5756	1.5679	**0.2269**	0.3149
WNLI	**MSE**	0.0070	0.0070	0.0070	0.0069	**0.0005**	0.0009
	KL Div.	1.5982	1.5977	1.5752	1.5688	**0.1375**	0.2211
STSB	**MSE**	0.0097	0.0097	0.0096	0.0096	**0.0009**	0.0012
	KL Div.	1.5230	1.5266	1.4991	1.4915	**0.1561**	0.2258
RTE	**MSE**	0.0048	0.0048	0.0048	0.0048	**0.0007**	0.0010
	KL Div.	1.9883	1.9881	1.9678	1.9593	**0.2533**	0.3930
Average GLUE	**MSE**	0.0112	0.0112	0.0112	0.0111	**0.0013**	0.0018
	KL Div.	1.6430	1.6450	1.6221	1.6143	**0.2019**	0.2981

Table A.1: The ApproxNet performance results over GLUE benchmark with various initialization setting.

In this section, we provide additional details on the attention approximation networks (ApproxNet). We train the ApproxNet following the instructions in Section 3.2. As we mentioned, the architecture of the ApproxNet is identical with the original network such as BERT (Devlin et al., 2019) and Vision Transformer (ViT) (Touvron et al., 2021), except for the dimension of hidden and intermediate hidden unit size. We downsize the architecture by reducing the hidden unit dimension with different reducing factors. We use 4 and 8 as the reducing factors in all experiments of the paper. We then train the ApproxNet with the knowledge distillation (Wang et al., 2020; Jiao et al., 2020). Specifically, we use the embedding based distillation loss $\mathcal{L}_{embd}$, hidden state based distillation loss $\mathcal{L}_{hidn}$, attention-based distillation loss $\mathcal{L}_{attn}$, and prediction-layer distillation loss $\mathcal{L}_{pred}$ to train the ApproxNet for BERT. For Vision Transformer, we do not use the prediction-layer distillation loss.

In Table A.1, we present the approximation error between the attention probability in the original network and the ApproxNet. We observe that the distillation helps the ApproxNet to approximate almost identical attention probability of the original network. We also observe that the smaller reduce factor leads to lower attention approximation error. However, we also find that the ApproxNet with a larger reducing factor also accurately approximate the attention probability of the original network.

To boost the training speed for ApproxNet on downstream tasks, we first distill the pre-trained Transformer encoders prior to the distillation of the fine-tuned network on each task. This procedure is not necessary if the downstream task is large enough or there is no downstream task such as ImageNet-1k. For instance, we directly train ApproxNet by setting the fine-tuned ViT as the teacher network with ImageNet-1k dataset. However, for experiments on GLUE benchmark, we first train the ApproxNet by setting the original pre-trained BERT as the teacher network with the Wikitext-103 dataset (Merity et al., 2017) for 200 epochs. Then, we finetune the ApproxNet by setting the fine-tuned BERT on each task as the teacher network with each task dataset for 30$\sim$50 epochs.

B ADDITIONAL EXPERIMENTAL RESULTS

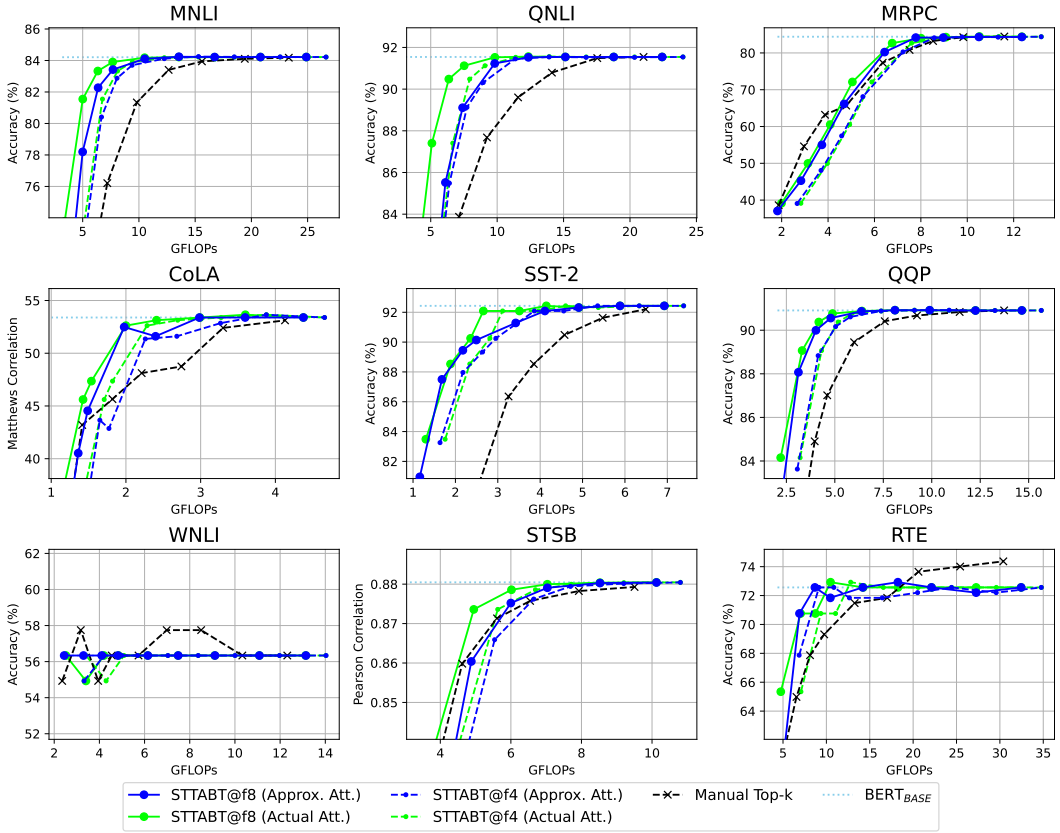


Figure B.1: FLOPs comparison between attention back-tracking with top-k and manual top-k baseline on all of datasets in GLUE benchmark. `STTABT@f`x means the attention back-tracking by ApproxNet with factor x.

B.1 ADDITIONAL RESULTS ON GLUE EXPERIMENTS WITH FLOPS

In Figure B.1, we present evaluation results on all GLUE benchmark with regards to FLOPs. We average every data point across 9 datasets in Figure B.1 to plot Figure 7a in the main paper.

B.2 ADDITIONAL VISUALIZATION EXAMPLES

In Figure B.2 and B.3, we visualize the additional visualization of the token pruning on each layer with STTABT (Concrete) and baselines (Kim et al., 2022; Rao et al., 2021). We further confirm that our method prunes more tokens throughout all of transformer layers than baselines in both sentence and image classification tasks.

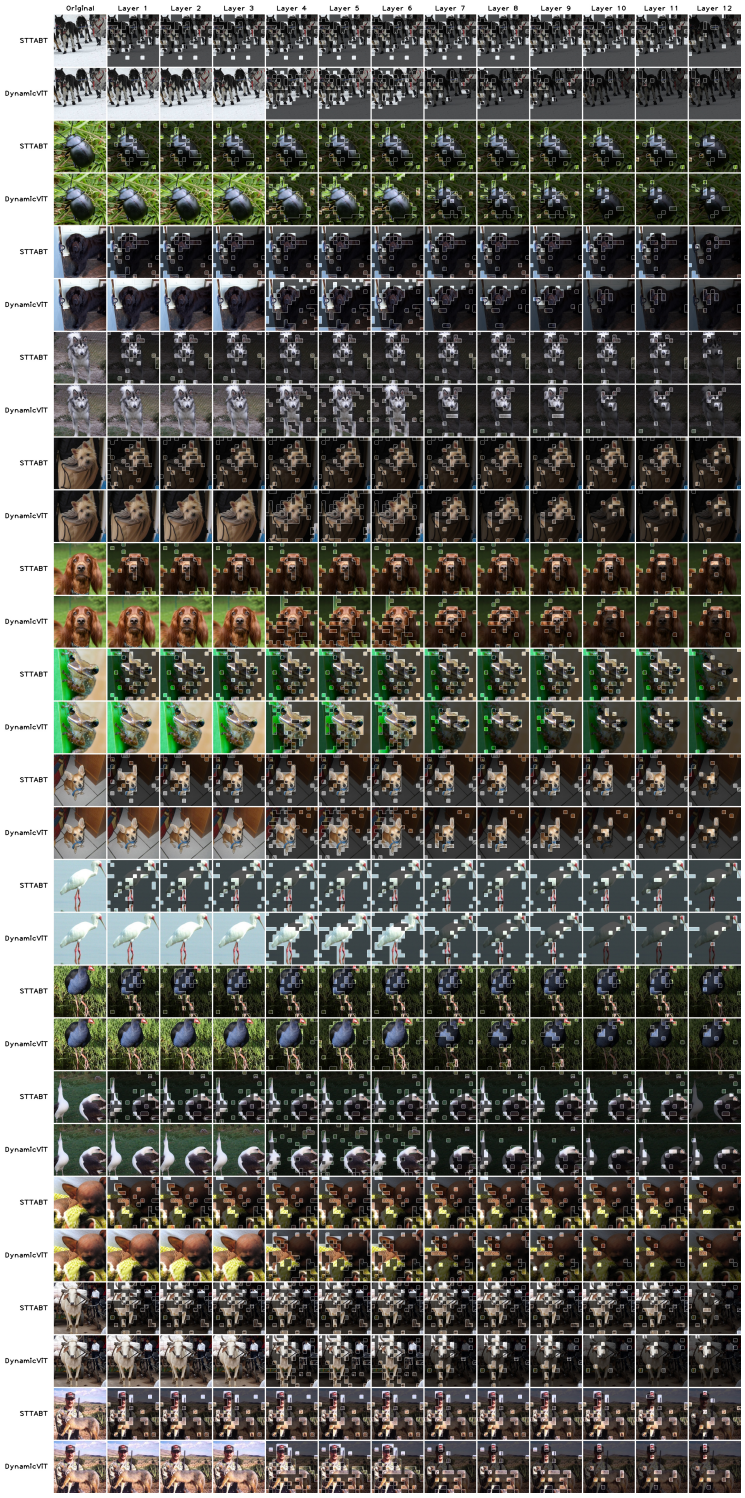


Figure B.2: More visualization examples of the token pruning on each layer with STTABT (Concrete) and DynamicViT on ImageNet-1k dataset.

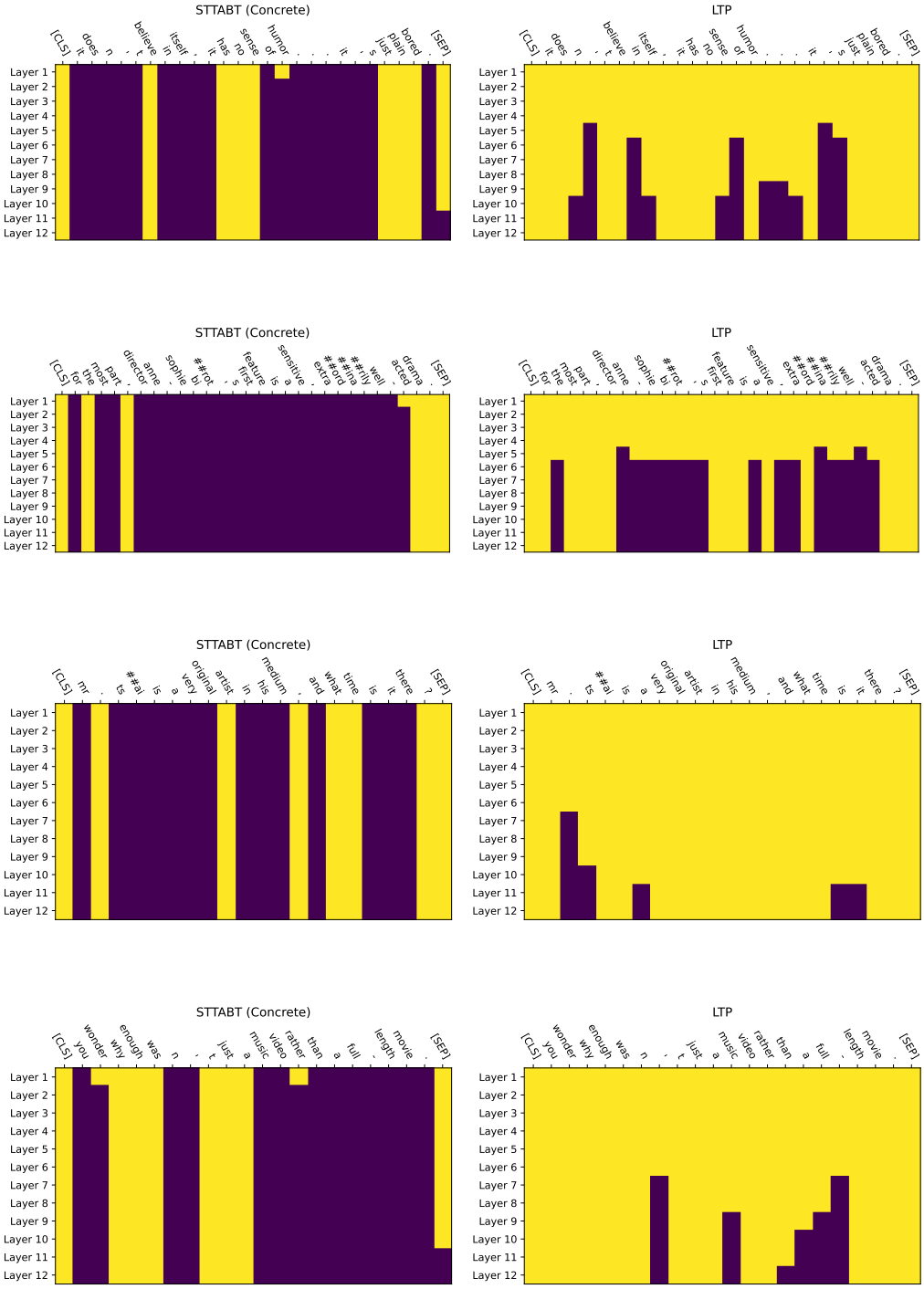


Figure B.3: More visualization examples of the token pruning on each layer with STTABT (Concrete) and LTP on SST-2 dataset.