AdapLeR: Speeding up Inference by Adaptive Length Reduction

Anonymous ACL submission

Abstract

Pre-trained language models have shown stellar performance in various downstream tasks. But, this usually comes at the cost of high latency and computation, hindering their usage in resource-limited settings. In this work, we propose a novel approach for reducing the computational cost of BERT with minimal loss in downstream performance. Our model dynamically eliminates less contributing tokens through layers, resulting in shorter lengths and consequently lower computational cost. To determine the importance of each token representation, we train a Contribution Predictor for each layer using a gradient-based saliency method. Our experiments on several diverse classification tasks show speedups up to 17x during inference time. We also validate the quality of the selected tokens in our method using human annotations in the ERASER benchmark. In comparison to other widely used strategies for selecting important tokens, such as *saliency* and *attention*, our proposed method has significantly less false positive rate in generating rationales.

1 Introduction

While large-scaled pre-trained language models exhibit stellar performances on various NLP benchmarks, their excessive computational costs and high inference latency have limited their usage in low-resource settings. In this regard, there have been various attempts at improving the efficiency of BERT-based models (Devlin et al., 2019), including knowledge distilation (Hinton et al., 2015; Sanh et al., 2019; Sun et al., 2019, 2020; Jiao et al., 2020), quantization (Gong et al., 2014; Shen et al., 2020; Tambe et al., 2021), weight pruning (Han et al., 2016; He et al., 2017; Michel et al., 2019; Sanh et al., 2020), and progressive module replacing (Xu et al., 2020). Despite providing significant reduction in model size, these techniques are generally static at inference time, i.e., they dedicate the same amount of computation to all inputs, irrespective of their difficulty.

A number of techniques have been also proposed in order to make efficiency enhancement sensitive to inputs. *Early exit* mechanism (Schwartz et al., 2020; Liao et al., 2021; Xin et al., 2020; Liu et al., 2020; Xin et al., 2021; Sun et al., 2021; Eyzaguirre et al., 2021) is a commonly used method in which each layer in the model is coupled with an intermediate classifier to predict the target label. At inference, a halting condition is used to determine whether the model allows an example to exit without passing through all layers. Various halting conditions have been proposed, including Shannon's entropy (Xin et al., 2020; Liu et al., 2020), softmax outputs with temperature calibration (Schwartz et al., 2020), trained confidence predictors (Xin et al., 2021), or the number of agreements between predictions of intermediate classifiers (Zhou et al., 2020).

Most of these techniques compress the model from the depth perspective (i.e., reducing the number of involved encoder layers). However, one can view compression from the width perspective (Goyal et al., 2020; Ye et al., 2021), i.e., reducing the length of hidden states. (Ethayarajh, 2019; Klafka and Ettinger, 2020). This is particularly promising as recent analytical studies showed that there are redundant encoded information in token representations (Klafka and Ettinger, 2020; Ethayarajh, 2019). Among these redundancies, some tokens relatively carry more task-specific information than others (Mohebbi et al., 2021), suggesting that only these tokens could be considered through the model. Moreover, in contrast to layer-wise pruning, token-level pruning does not sacrifice the model's capacity in complex reasoning (Sanh et al., 2019; Sun et al., 2019).

PoWER-BERT (Goyal et al., 2020) is one of the first such techniques which reduces inference time by eliminating redundant token representations through layers based on self-attention weight. Several studies have followed (Kim and Cho, 2021; Wang et al., 2021); However, they usually optimize a single token elimination configuration across the entire dataset, resulting in a static model. In addition, their token selection strategies are based on attention weights which can result in a sub-optimal solution (Ye et al., 2021). In this work, we introduce **Adap**tive **Le**ngth **R**eduction (**AdapLeR**), which instead of relying on attention weights, we train a set of Contribution Predictors (CP) to estimate tokens' saliency scores at inference. We show that this choice results in more reliable scores than attention weights in measuring tokens' contributions.

The most related study to ours is TR-BERT (Ye et al., 2021) which leverages reinforcement learning to develop an input-adaptive token selection policy network. However, as pointed out by the authors, the problem has a large search space, making it difficult for RL to solve. To mitigate this, they resorted to extra heuristics such as imitation learning (Hussein et al., 2017) for warming up the training of the policy network, action sampling for limiting the search space, and knowledge distillation for transferring knowledge from the intact backbone fine-tuned model. All of these steps significantly increase the training cost. Hence, they only perform token selection at two layers. In contrast, we propose a simple but effective method to gradually eliminate tokens in each layer throughout the training phase using a soft-removal function which allows the model to be adaptable to various inputs in a batch-wise mode. It is also worth noting above studies are based on top-k operations for identifying the k most important tokens during training or inference, which can be expensive without a specific hardware architecture (Wang et al., 2021).

In summary, our contributions are threefold:

- We couple a simple Contribution Predictor (CP) with each layer of the model to estimate tokens' contribution scores to eliminate redundant representations.
- Instead of an instant token removal, we gradually mask out less contributing token representations by employing a novel soft-removal function.
- We also show the superiority of our token selection strategy over the other widely used strategies by using human rationales.

2 Background

2.1 Self-attention Weights

Self-attention is a core component of the Transformers (Vaswani et al., 2017) which looks for the relation between different positions of a single sequence of token representations $(x_1, ..., x_n)$ to build contextualized representations. To this end, each input vector x_i is multiplied by the corresponding trainable matrices Q, K, and V to respectively produce query (q_i), key (k_i), and value (v_i) vectors. To construct the output representation z_i, a series of weights is computed by the dot product of q_i with every k_j in all time steps. Before applying a softmax function, these values are divided by a scaling factor and then added to an *attention mask* vector $\mathbf{m}$, which is zero for positions we wish to attend and $-\infty$ (in practice, -10000) for padded tokens (Vaswani et al., 2017). Mathematically, for a single attention head, the weight attention from token x_i to token x_j in the same input sequence can be written as:

$$\alpha_{i,j} = \underset{x_j \in \mathcal{X}}{\mathrm{softmax}} \left(\frac{q_i k_j^\top}{\sqrt{d}} + m_i \right) \in \mathbb{R} \qquad (1)$$

The time complexity for this is $O(n^2)$ given the dot product $q_i k_j^\top$, where n is the input sequence length. This impedes the usage of self-attention based models in low-resource settings.

While self-attention is one of the most white-box components in transformer-based models, relying on raw attention weights as an explanation could be misleading given that they are not necessarily responsible for determining the contribution of each token in the final classifier's decision (Jain and Wallace, 2019; Serrano and Smith, 2019; Abnar and Zuidema, 2020). This is based on the fact that raw attentions are being faithful to the local mixture of information in each layer and are unable to obtain a global perspective of the information flow through the entire model (Pascual et al., 2021).

2.2 Gradient-based Saliency Scores

Gradient-based methods provide alternatives to attention weights to compute the importance of a specific input feature. Despite having been widely utilized in other fields earlier (Ancona et al., 2017; Simonyan et al., 2013; Sundararajan et al., 2017; Smilkov et al., 2017), they have only recently become popular in NLP studies (Bastings and Filippova, 2020; Li et al., 2016; Yuan et al., 2019).

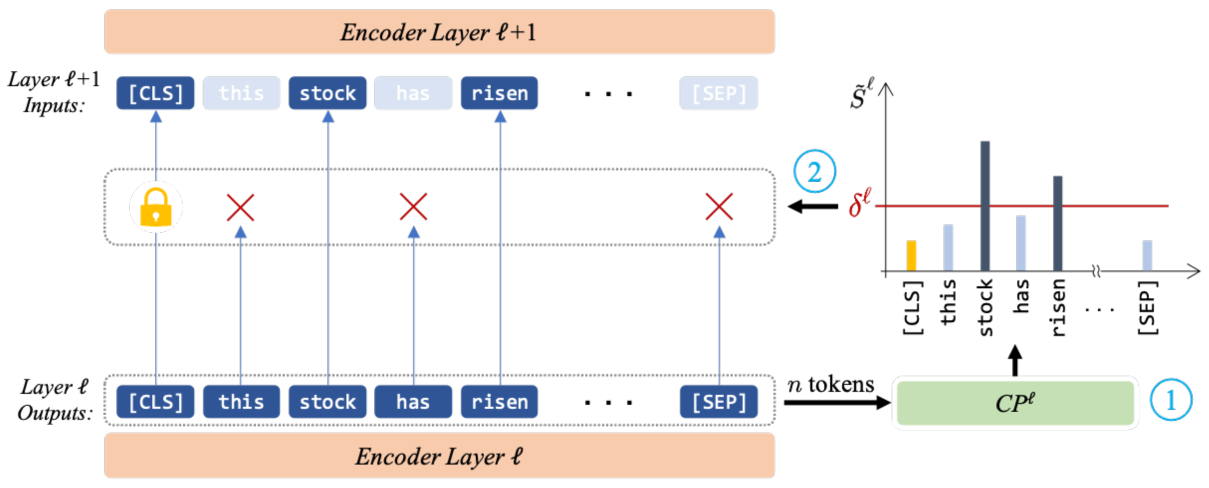

Figure 1: To reduce the inference computation, in each layer (1) the attribution score of the token representation is estimated and (2) based on a reduced uniform-level threshold ($\delta^\ell = \eta^\ell/n$) token representations with low importance score are removed. Since the final layer's classifier is connected to the [CLS] token and it could act as a pooler within each layer it is the only token that would remain regardless of its score.

These methods are based on computing the first-order derivative of the output probability y_c w.r.t. the input embedding h_i^0 (initial hidden states), where c could be true class label to find the most important input features or the predicted class to interpret model's behavior. After taking the norm of output derivatives, we get *sensitivity* (Ancona et al., 2017), which indicates the changes in model's output with respect to the changes in specific input dimensions. Instead, by multiplying gradients with input features, we arrive at *InputXGradient*, also known as *saliency*, which also considers the direction of input vectors to determine the most important tokens (Bastings and Filippova, 2020). Since these scores are computed for each dimension of embedding vectors, an aggregation method such as L2 norm or mean is needed to produce one score per input token (Atanasova et al., 2020a):

$$S_i = \| \frac{\partial y_c}{\partial h_i^0} \odot h_i^0 \|_2 \qquad (2)$$

3 Methodology

As shown in Figure 1, our approach relies on dropping low contributing tokens in each layer and passing only the more important ones to the next. Therefore, one important step is to measure the importance of each token. To this end, we opted for saliency scores which is a more reliable criterion in measuring token's contributions (Bastings and Filippova, 2020; Pascual et al., 2021). We will show in Section 5.1 results of a series quantitative analyses that supports this choice. In what follows, we first describe how we estimate saliency scores at inference time using a set of Contribution Predictors (CPs) and then we elaborate on how we leverage these predictors during inference (Section 3.2) and training (Section 3.3) phase.

3.1 Contribution Predictor

Computing gradients during inference is problematic in two ways. First, back-propagation computation prolongs inference time, which is contrary to our main goal. Second, computing gradients based on true labels is impossible given that no labels are available at inference time.

To circumvent these, we simply add a CP after each layer ℓ in the model to estimate contribution score for each token representation, i.e., $\tilde{S}_i^\ell$. The model then decides on the tokens that should be passed to the next layer based on the values of $\tilde{S}_i^\ell$. CP computes $\tilde{S}_i^\ell$ for each token using an MLP followed by a softmax activation function. We argue that, despite being limited in learning capacity, the MLP is sufficient for estimating scores that are more generalized and relevant than vanilla saliency values. We will present a quantitative analysis on this topic in Section 5.

3.2 Model Inference

Most BERT-based models consist of L encoder layers. The input sequence of n tokens is usually passed through an embedding layer to build the initial hidden states of the model h^0. Each encoder layer then produces the next hidden states using the ones from the previous layer:

$$h^\ell = \text{Encoder}_\ell(h^{\ell-1}) \quad (3)$$

In our approach, we eliminate less contributing token representations before delivering hidden states to the next encoder. Tokens are selected based on the contribution scores $\tilde{S}^\ell$ obtained from the CP of the corresponding layer ℓ. Based on the fact that the sum of these scores is one, we can define a lower cutoff threshold than uniform as: $\delta^\ell = \eta^\ell \cdot {}^1\!/_n$ with $0 < \eta^\ell < 1$ to distinguish important tokens. Tokens are considered important if their contribution score exceeds δ (which is a smaller value than the uniform score). Intuitively, a larger η provides a higher δ cutoff level, thereby dropping a larger number of tokens, hence, yielding more speedup. The value of η determines the extent to which we can rely on CP's estimations. In case the estimations of CP are deemed to be inaccurate, its impact can be reduced by lowering η. We train each layer's η^ℓ using an auxiliary training objective, which allows the model to adjust the cutoff value to control the speedup-performance tradeoff. Given that the final classification head uses the last hidden state of the [CLS] token, we preserve this token's representation in all layers, regardless of its contribution score. Also, since each input instance has a different computational path during token removal process, it is obvious that at inference time the batch size should be equal to one (single instance usage), similarly to other dynamic approaches (Zhou et al., 2020; Liu et al., 2020; Ye et al., 2021; Eyzaguirre et al., 2021; Xin et al., 2020).

3.3 Model Training

Training consists of three phases: initial finetuning, saliency extraction, and adaptive length retraining. In the first phase, we simply finetune the backbone model (BERT) on a given target task. We then extract the saliencies of three top-perfroming checkpoints from the finetuning process and compute the average of them to mitigate potential inconsistencies in saliency scores (cf. Section 2.2). The final step is to train a pre-trained model using an adaptive length reduction procedure. In this phase, a non-linear function gradually fades out the representations throughout the training process. Each CP is jointly trained with the rest of the model using the saliencies extracted in the previous phase alongside with the target task labels.

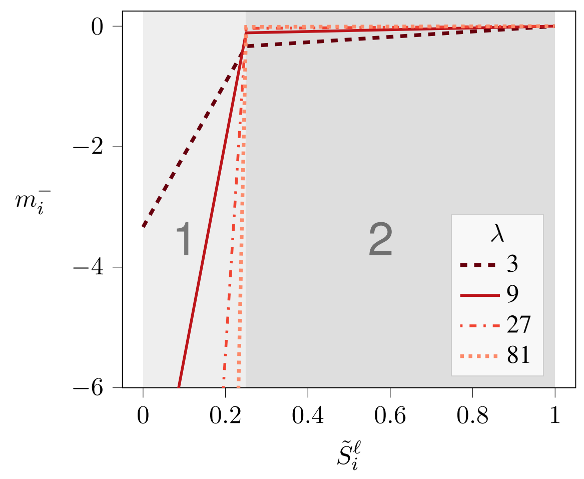

Figure 2: The soft-removal function plotted with $\lambda \in \{3, 9, 27, 81\}$ and $\delta^\ell = 0.25$. As λ increases, the removal region (1) gets steeper while the other zone (2), which is almost horizontal, approaches the zero level.

We also define a speedup tuning objective to determine the thresholds (via tuning η) to control the performance-speedup trade-off. In the following, we elaborate on the procedure.

Soft-removal function. If during training tokens are dropped similarly to the inference mode, we cannot leverage batch-wise training as the structure will vary with each example. Furthermore, due to the discrete nature of a hard removal, the effect of dropping tokens cannot be captured using a gradient back-propagation procedure. Hence, inspired by the padding mechanism of self-attention models (Vaswani et al., 2017) we introduce a new method that gradually masks out less contributing token representations. In each layer, after predicting contribution scores, instead of simply removing the token representations, we accumulate a negative mask to the attention mask vector M using a soft-removal function:

$$m_i^-(\tilde{S}_i^\ell) = \begin{cases} \lambda_{adj}(\tilde{S}_i^\ell - \delta^\ell) - \dfrac{\beta}{\lambda} & \tilde{S}_i^\ell < \delta^\ell \\ \dfrac{(\tilde{S}_i^\ell - 1)\beta}{(1-\delta^\ell)\lambda} & \tilde{S}_i^\ell \geq \delta^\ell \end{cases} \quad (4)$$

This function consists of two main zones (Figure 2). In the first term, the less important tokens with scores lower than the threshold (δ^ℓ) are assigned higher negative masking as they get more distant from δ. The slope is determined by $\lambda_{adj} = {}^\lambda\!/_\delta$, where λ is a hyperparameter that is increased exponentially after each epoch (e.g., $\lambda \leftarrow 10 \times \lambda$ after finishing each epoch). Increasing λ makes the soft-removal function stronger and more decisive in masking the representations. To avoid undergoing zero gradients during training, we define $0 < \beta < 0.1$ to construct a small negative slope (similar to the well known Leaky-ReLU of Maas et al. 2013) for those tokens with higher contributing scores than δ^ℓ threshold. Consider a scenario in which η^ℓ sharply drops, causing most of $\tilde{S}_i^\ell$ get over the δ^ℓ threshold. In this case, the non-zero value in the second term of Equation 4, which facilitates optimizing η^ℓ.

Training the Contribution Predictors. The CPs are trained by an additional term which is based on the KL-divergence of each layer's CP output with the extracted saliencies. The main training objective is a minimization of the following loss:

$$\mathcal{L} = \mathcal{L}_{\text{CE}} + \gamma \mathcal{L}_{\text{CP}} \tag{5}$$

Where γ is a hyperparameter which that specifies the amount of emphasis on the CP training loss:

$$\begin{aligned} \mathcal{L}_{\text{CP}} &= \sum_{\ell=0}^{L-1} D_{\text{KL}}(\hat{S}^\ell || \tilde{S}^\ell) \\ &= \sum_{\ell=0}^{L-1} (L-\ell) \sum_{i=1}^{N} \hat{S}_i^\ell \log(\frac{\hat{S}_i^\ell}{\tilde{S}_i^\ell}) \end{aligned} \tag{6}$$

Since S is based on the input embeddings, the [CLS] token usually shows a low amount of contribution due to not having any contextualism in the input. We hypothesize that this token acts similarly to a pooler and gathers information about the context of the input. Therefore, the token can potentially have more contribution as it passes through the model. To this end, we amplify the contribution score of [CLS] and renormalize the distribution ($\hat{S}^\ell$) with a trainable parameter θ^ℓ:

$$\hat{S}_i^\ell = \frac{\theta^\ell S_1^\ell \mathbf{1}[i=1] + S_i^\ell \mathbf{1}[i>1]}{\theta^\ell S_1^\ell + \sum_{i=2}^{n} S_i^\ell} \tag{7}$$

By this procedure, the next objective (discussed in the next paragraph) will have the capability of tuning the amount of pooling, consequently controlling the amount of speedup. Larger θ push the CPs to shift the contribution towards the [CLS] token and drop more tokens.

Speedup Tuning. In the speedup tuning process, we combine the cross-entropy loss of the target classification task with a length loss which is the expected number of unmasked token representations in all layers. Considering that we have a non-positive and continuous attention mask M, the length loss of a single layer would be the summation over the exponential of the mask values $\exp(m_i)$ to map the masking range $[-10000, 0]$ to a $[0 \text{ (fully masked/removed)}, 1 \text{ (fully retained)}]$ bound.

$$\begin{aligned} \mathcal{L}_{\text{SPD./PERF.}} &= \mathcal{L}_{\text{CE}} + \phi \mathcal{L}_{\text{LENGTH}} \\ \mathcal{L}_{\text{LENGTH}} &= \sum_{l=1}^{L} \sum_{i=1}^{n} \exp(m_i^\ell) \end{aligned} \tag{8}$$

In Equation 8, demonstrates how the length loss is computed inside the model and how its added to the main classification loss. During training, we assign a separate optimization process which tunes η and θ to adjust the thresholds and the amount of [CLS] pooling[1] alongside with the previous objective (CP Training).

The reason that this objective is treated as a separate problem instead of merging it with the previous one, is because in the latter case the CPs could be influenced by the length loss and try to manipulate the contribution scores for some tokens regardless of their real influence. So in other words, the first objective is to solve the task and make it explainable with the CPs, and the secondary objective builds the speedup using tuning the threshold levels and the amount of pooling in each layer.

4 Experiments

4.1 Datasets.

To verify the effectiveness of our proposed method on adaptive length reduction, we selected seven text classification datasets. In order to incorporate a variety of tasks, we utilized SST-2 (Socher et al., 2013) and IMDB (Maas et al., 2011) for sentiment, CoLA (Warstadt et al., 2019) for linguistic acceptability, MRPC (Dolan and Brockett, 2005) for paraphrase, AG's News (Zhang et al., 2015) for topic classification, MNLI (Williams et al., 2018) for NLI, and QNLI (Rajpurkar et al., 2016) for question answering. Evaluations are based on the test split of each dataset. For those datasets that are in the GLUE Benchmark (Wang et al., 2018), test results were acquired by submitting the test

[1]Since θ is not inside the main computational DAG of the model, we employed a dummy variable inside the model with frozen values equal to 1 and used that variable to compute the gradients.

Model	SST-2		IMDB		CoLA		MRPC		MNLI		QNLI		AG's news	
	Acc.	FLOPs	Acc.	FLOPs	MCC	FLOPs	F1.	FLOPs	Acc.	FLOPs	Acc.	FLOPs	Acc.	FLOPs
BERT	92.7	1.00x	93.8	1.00x	52.7	1.00x	87.5	1.00x	84.2	1.00x	90.3	1.00x	94.4	1.00x
PoWER-BERT	92.1	1.18x	92.2	1.70x	48.7	1.25x	88.0	1.07x	82.9	1.10x	89.7	1.23x	92.1	12.5x
TR-BERT	93.4	1.09x	93.2	2.90x	00.0	1.81x	81.9	1.16x	84.8	1.00x	89.0	1.09x	93.2	10.2x
AdapLeR	92.3	1.49x	91.7	3.21x	51.1	1.39x	87.6	1.27x	82.9	1.42x	89.3	1.47x	92.5	17.1x

Table 1: Comparison of our method (AdapLeR) with other baselines in seven classification tasks in terms of performance and speedup (FLOPs). For each dataset the corresponding metric has been reported (Matthew's: MCC, Accuracy: Acc., F1: F-1 Score). In the MNLI task, the speedup and performance values are the average of the evaluations on the matched and mismatched test sets.

predictions to the evaluation server. For IMDB and AG's News, results were computed based on the test set provided.

4.2 Experimental Setup

To compare our approach, we set our first baseline to be the pre-trained BERT (base-uncased) (Devlin et al., 2019) which is also the backbone model of our model and the other two baselines: TR-BERT and PoWER-BERT. For the latter two, we used the same implementations and suggested hyperparameters[2] to train these baselines. To fine-tune the backbone model we used similar hyperparameters over all tasks that are provided in Section B. The backbone model and our model implementation is based on the HuggingFace's Transformers library (Wolf et al., 2020). Trainings and evaluations were conducted on a dual 2080Ti 11GB GPU machine with multiple runs.

Hyperparameter Selection. Overall, we introduced several hyperparameters which are involved in the training process. However, the main two primary terms that are the most influential and have considerable effects on both the output performance and the speedup of the trained model are ϕ and γ. This makes our approach comparable to existing techniques (Goyal et al., 2020; Ye et al., 2021) which usually have two or three hyperparameters adjusted per task. While using grid search for these two terms, we kept other hyperparameters constant over all datasets. The final selected hyperparamters and more details are discussed in Section B.

FLOPs Computation. As we wish to determine the computational complexity of models independently of the operating environment (e.g., CPU/GPU), following Ye et al. (2021); Liu et al. (2020), we computed FLOPs, i.e., the number of floating-point operations (FLOPs) in a single inference procedure. To have a fair comparison, we computed FLOPs for PoWER-BERT in a single instance mode, described in Section A.

[2]Since some of the datasets were not used originally, we had to search the hyperparameters based on the given ranges

4.3 Results

The performance and speedup values of our proposed method and other baselines are presented in Table 1. We can observe that with a low performance gap in all tasks, our approach outperforms in terms of efficiency. It is noteworthy that the results also reveal some form of dependency on the type of tasks. Some tasks may need less amount of contextualism during inference and could be classified by using a fraction of the total amount of tokens. For instance, in AG's News, the topic of a sentence might be identified with a single token (e.g. Basketball $\rightarrow$ Topic: Sports).

Even though showing efficiency improvements in CoLA, MRPC and MNLI, it seems that adaptive length based methods would not be as capable as in other tasks. This problem might stem from the nature of the tasks themselves. For instance, linguistic acceptability could be compromised by the removal of some tokens, or detecting paraphrased sentences with removed tokens could buildup dissimilarities between the two sentences which leads to spurious results.

5 Analysis

In this section, we first conduct an experiments to support our choice of saliency scores as a supervision in measuring the importance of token representations. Next, we validate the behavior of Contribution Predictors in identifying most important tokens in an AdapLeR model.

	Movie Reviews		MultiRC	
Strategy	Acc.	FLOPs	Acc.	FLOPs
Full input	93.3	1	67.7	1
Human rationale	96.7	3.7	76.6	4.6
Saliency	**92.3**	3.7	**66.4**	4.4
Attention ALL	78.5	3.7	62.9	4.4
Attention [CLS]	70.3	3.7	63.7	4.4

Table 2: Accuracy and speedup when the most important input tokens during fine-tuning are computed based on attention and saliency strategies and human rationale (the upper bound). The bold values indicate the corresponding better strategy for each task (the closest performance to the upper bound).

5.1 Saliency vs. Attention

In dealing with token pruning, a natural question that might arise is what would be the most appropriate criterion for assessing the relative importance of tokens within a sentence? To arrive at an empirical and reliable upper bound in measuring token importance, we resort to human rationale. To this end, we used the ERASER benchmark (DeYoung et al., 2020), which contains multiple tasks for which important spans of the input text have been highlighted as supporting evidence (aka "rationale") by human. Among the benchmark tasks, we opted for two diverse classification tasks: Movie reviews (Zaidan and Eisner, 2008) and MultiRC (Khashabi et al., 2018). In the former task, the model predicts the sentiment based on multiple sentences. The MultiRC dataset contains a passage, a question, and multiple candidate answers, which is cast as a binary classification task of passage/question/answer triplets in ERASER.

In order to verify the reliability of human rationales, we fine-tuned BERT just on that rationales by excluding those tokens that are not highlighted in the input.[3] In Table 2, the first two rows show the performance score of BERT on target tasks with full tokens and only rationales in the input. We see that fine-tuning merely on rationales not only yielded less computation cost, but also resulted in higher performance when compared with the full input setting. Obviously, human annotations are not available for a whole range of downstream NLP tasks; therefore, this criteria is infeasible in practice and can only be viewed as an upper bound for evaluating different strategies in measuring token importance. We investigated the effectiveness of saliency and self-attention weights as two commonly used strategies for measuring the importance of tokens in pre-trained language models.

[3]During removing tokens from the input, we ensured that the positional information of the remaining tokens remained unchanged to mitigate discrepancies with the original input.

To compute these, we first fine-tuned BERT with all tokens in the input for a given target task. We then obtained saliency scores with respect to the tokens in the input embedding layer. This gives us two advantages. First, representations in this layer are non-contextualized, allowing us to measure the importance of each token individually. Second, the fact that the gradient passes from the end to the beginning of the model results in aggregated values for the relative importance of each token based on the entire model. Similarly, we aggregated self-attention weights across all layers of the model using a post-processed variant of attentions called *attention rollout* (Abnar and Zuidema, 2020), a popular technique in which each attention weight matrix in each layer is multiplied by the ones before it to form aggregated attention values. To assign an importance score to each token, we examined two different interpretation of attention weights. The first strategy is the one adopted by PoWER-BERT (Goyal et al., 2020) in which for each token we accumulate attention values from other tokens. Additionally, we measured how much the [CLS] token attends to each token in the input, a strategy which has been widely used in interpretability studies around BERT (Abnar and Zuidema, 2020; Chrysostomou and Aletras, 2021; Jain et al., 2020, *inter alia*). For a fair evaluation, for each sentence in the test set, we selected the top-k salient and attended words, with k being the number of words that are annotated as rationales.

Results in Table 2 show that fine-tuning on the most salient tokens outperforms that based on the most attended tokens. This denotes that saliency is a better indicator for the importance of tokens. Nonetheless, recent length reduction techniques (Goyal et al., 2020; Kim and Cho, 2021; Wang et al., 2021) have mostly adopted attention weights as their criterion for selecting important tokens as these weights are convenient to compute during the inference.

5.2 Contribution Predictor Evaluation

The goal of this section is to validate our Contribution Predictors in selecting the most contributed tokens and investigate whether our model is able

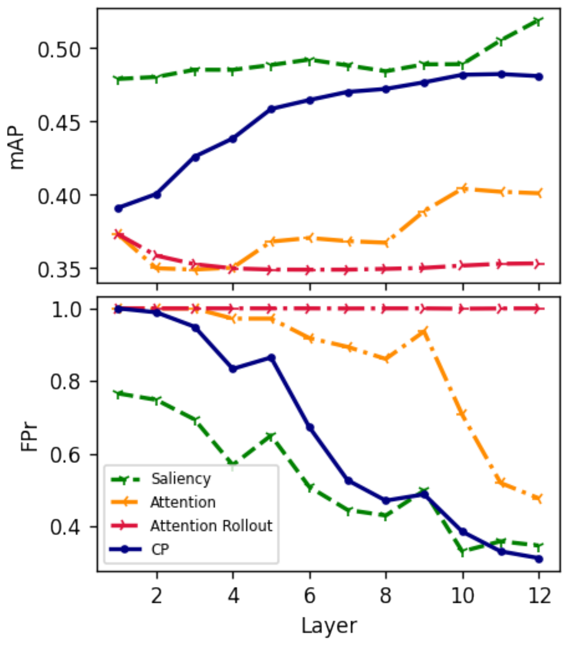

Figure 3: Evaluating our CP's output scores with other three alternative token importance scores in terms of agreement with human rationales.

to preserve rationales. For evaluation, we used two Average Precision (AP) and False Positive Rate (FPR) by comparing the remaining tokens to the human rationale annotations. The first metric measures whether the model assigns higher continuous scores to those tokens that are annotated by humans as rationales. Whereas, the intuition behind the second metric is how many irrelevant tokens are selected by the model to be passed to subsequent layers.

First, we fine-tuned the model on the Movie Review dataset and computed layer-wise raw attention, attention rollout, and saliency scores for each token representation. We also trained a model using our proposed approach and computed the output probability scores of CPs in each layer. Since human rationales are annotated at the word level, we sum the scores across tokens corresponding to each word to arrive at word-level importance scores. In addition to these soft scores, we used the threshold of δ^{ℓ} introduced in section 3.3 to reach a binary score indicating tokens selected in each layer. We used soft scores for computing AP and binary scores for computing FPR.

Figure 3 shows the agreement between human rationales and the selected tokens based on these two metrics. As we can see, in comparison to other widely used strategies for selecting important tokens, such as salinecy and attention, our Contribution Predictors has have significantly less false positive rate in generating rationales. Note that saliency strategy requires the target true labels. There is also a line of research in which practitioners attempt to guide models to perform human-like reasoning by training rationale generation simultaneously with the target task that requires human annotation (Atanasova et al., 2020b; Zhao et al., 2020; Li et al., 2018). However, as a by-product, our trained CPs tries to generate rationales at inference without any supervision.

6 Conclusion

In this paper we introduced AdapLeR, a model that dynamically identifies and drops less contributing token representations through layers. Specifically, AdapLeR accomplishes this by training a set of Contribution Predictors based on saliencies extracted from a finetuned model and applying a gradual masking technique to simulate input-adaptive token removal during training. Empirical results on seven diverse text classification tasks show considerable improvements over previous methods. Furthermore, we demonstrated that contribution predictors generate rationales that are highly in line with those manually specified by humans. As future work, we aim to apply our technique to more tasks and see whether it can be adapted to NLP tasks that rely on all token representations (e.g., question answering). Additionally, combining our width-based strategy with a depth-based one (e.g., early exiting) might potentially yield greater efficiency, something we plan to pursue as future work.

References

Samira Abnar and Willem Zuidema. 2020. Quantifying attention flow in transformers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4190–4197, Online. Association for Computational Linguistics.

Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. 2017. Towards better understanding of gradient-based attribution methods for deep neural networks. *arXiv preprint arXiv:1711.06104*.

Pepa Atanasova, Jakob Grue Simonsen, Christina Lioma, and Isabelle Augenstein. 2020a. A diagnostic study of explainability techniques for text classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*

(EMNLP), pages 3256–3274, Online. Association for Computational Linguistics.

Pepa Atanasova, Jakob Grue Simonsen, Christina Lioma, and Isabelle Augenstein. 2020b. Generating fact checking explanations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7352–7364, Online. Association for Computational Linguistics.

Jasmijn Bastings and Katja Filippova. 2020. The elephant in the interpretability room: Why use attention as explanation when we have saliency methods? In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 149–155, Online. Association for Computational Linguistics.

George Chrysostomou and Nikolaos Aletras. 2021. Enjoy the salience: Towards better transformer-based faithful explanations with word salience. *ArXiv*, abs/2108.13759.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Jay DeYoung, Sarthak Jain, Nazneen Fatema Rajani, Eric Lehman, Caiming Xiong, Richard Socher, and Byron C. Wallace. 2020. ERASER: A benchmark to evaluate rationalized NLP models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4443–4458, Online. Association for Computational Linguistics.

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

Kawin Ethayarajh. 2019. How contextual are contextualized word representations? Comparing the geometry of BERT, ELMo, and GPT-2 embeddings. 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 55–65, Hong Kong, China. Association for Computational Linguistics.

Cristóbal Eyzaguirre, Felipe del Río, Vladimir Araujo, and Álvaro Soto. 2021. Dact-bert: Differentiable adaptive computation time for an efficient bert inference. *arXiv preprint arXiv:2109.11745*.

Yunchao Gong, L. Liu, Ming Yang, and Lubomir D. Bourdev. 2014. Compressing deep convolutional networks using vector quantization. *ArXiv*, abs/1412.6115.

Saurabh Goyal, Anamitra Roy Choudhury, Saurabh Raje, Venkatesan Chakaravarthy, Yogish Sabharwal, and Ashish Verma. 2020. Power-bert: Accelerating bert inference via progressive word-vector elimination. In *International Conference on Machine Learning*, pages 3690–3699. PMLR.

Song Han, Huizi Mao, and William J. Dally. 2016. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. *arXiv: Computer Vision and Pattern Recognition*.

Yihui He, Xiangyu Zhang, and Jian Sun. 2017. Channel pruning for accelerating very deep neural networks. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1398–1406.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *ArXiv*, abs/1503.02531.

Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. 2017. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2):1–35.

Sarthak Jain and Byron C. Wallace. 2019. Attention is not Explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3543–3556, Minneapolis, Minnesota. Association for Computational Linguistics.

Sarthak Jain, Sarah Wiegreffe, Yuval Pinter, and Byron C. Wallace. 2020. Learning to faithfully rationalize by construction. In *ACL*.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. TinyBERT: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174, Online. Association for Computational Linguistics.

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 2018. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 252–262, New Orleans, Louisiana. Association for Computational Linguistics.

Gyuwan Kim and Kyunghyun Cho. 2021. Length-adaptive transformer: Train once with length drop, use anytime with search. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6501–6511, Online. Association for Computational Linguistics.

Josef Klafka and Allyson Ettinger. 2020. Spying on your neighbors: Fine-grained probing of contextual embeddings for information about surrounding words. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4801–4811, Online. Association for Computational Linguistics.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2016. Visualizing and understanding neural models in NLP. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 681–691, San Diego, California. Association for Computational Linguistics.

Sizhen Li, Shuai Zhao, Bo Cheng, and Hao Yang. 2018. An end-to-end multi-task learning model for fact checking. In *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, pages 138–144, Brussels, Belgium. Association for Computational Linguistics.

Kaiyuan Liao, Yi Zhang, Xuancheng Ren, Qi Su, Xu Sun, and Bin He. 2021. A global past-future early exit method for accelerating inference of pre-trained language models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2013–2023, Online. Association for Computational Linguistics.

Weijie Liu, Peng Zhou, Zhiruo Wang, Zhe Zhao, Haotang Deng, and Qi Ju. 2020. FastBERT: a self-distilling BERT with adaptive inference time. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6035–6044, Online. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In *NeurIPS*.

Hosein Mohebbi, Ali Modarressi, and Mohammad Taher Pilehvar. 2021. Exploring the role of BERT token representations to explain sentence probing results. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 792–806, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Damian Pascual, Gino Brunner, and Roger Wattenhofer. 2021. Telling BERT's full story: from local attention to global aggregation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 105–124, Online. Association for Computational Linguistics.

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

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Victor Sanh, Thomas Wolf, and Alexander M. Rush. 2020. Movement pruning: Adaptive sparsity by fine-tuning. *ArXiv*, abs/2005.07683.

Roy Schwartz, Gabriel Stanovsky, Swabha Swayamdipta, Jesse Dodge, and Noah A. Smith. 2020. The right tool for the job: Matching model and instance complexities. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6640–6651, Online. Association for Computational Linguistics.

Sofia Serrano and Noah A. Smith. 2019. Is attention interpretable? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2931–2951, Florence, Italy. Association for Computational Linguistics.

Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. 2020. Q-bert: Hessian based ultra low precision quantization of bert. In *AAAI*.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*.

D Smilkov, N Thorat, B Kim, F Viégas, and M Wattenberg. 2017. Smoothgrad: removing noise by adding noise. arxiv. *arXiv preprint arxiv:1706.03825*.

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.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019. Patient knowledge distillation for BERT model compression. 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 4323–4332, Hong Kong, China. Association for Computational Linguistics.

Tianxiang Sun, Yunhua Zhou, Xiangyang Liu, Xinyu Zhang, Hao Jiang, Zhao Cao, Xuanjing Huang, and Xipeng Qiu. 2021. Early exiting with ensemble internal classifiers. *arXiv preprint arXiv:2105.13792*.

Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. 2020. MobileBERT: a compact task-agnostic BERT for resource-limited devices. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2158–2170, Online. Association for Computational Linguistics.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3319–3328.

Thierry Tambe, Coleman Hooper, Lillian Pentecost, Tianyu Jia, En-Yu Yang, Marco Donato, Victor Sanh, Paul N. Whatmough, Alexander M. Rush, David Brooks, and Gu-Yeon Wei. 2021. Edgebert: Sentence-level energy optimizations for latency-aware multi-task nlp inference. *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Hanrui Wang, Zhekai Zhang, and Song Han. 2021. Spatten: Efficient sparse attention architecture with cascade token and head pruning. *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 97–110.

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

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi 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 Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. 2020. DeeBERT: Dynamic early exiting for accelerating BERT inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2246–2251, Online. Association for Computational Linguistics.

Ji Xin, Raphael Tang, Yaoliang Yu, and Jimmy Lin. 2021. BERxiT: Early exiting for BERT with better fine-tuning and extension to regression. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 91–104, Online. Association for Computational Linguistics.

Canwen Xu, Wangchunshu Zhou, Tao Ge, Furu Wei, and Ming Zhou. 2020. BERT-of-theseus: Compressing BERT by progressive module replacing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7859–7869, Online. Association for Computational Linguistics.

Deming Ye, Yankai Lin, Yufei Huang, and Maosong Sun. 2021. TR-BERT: Dynamic token reduction for accelerating BERT inference. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5798–5809, Online. Association for Computational Linguistics.

Hao Yuan, Yongjun Chen, Xia Hu, and Shuiwang Ji. 2019. Interpreting deep models for text analysis via optimization and regularization methods. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5717–5724.

Omar Zaidan and Jason Eisner. 2008. Modeling annotators: A generative approach to learning from annotator rationales. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 31–40, Honolulu, Hawaii. Association for Computational Linguistics.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *NIPS*.

Chen Zhao, Chenyan Xiong, Corby Rosset, Xia Song, Paul Bennett, and Saurabh Tiwary. 2020. Transformer-xh: Multi-evidence reasoning with extra hop attention. In *International Conference on Learning Representations*.

Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian McAuley, Ke Xu, and Furu Wei. 2020. Bert loses patience: Fast and robust inference with early exit. In *Advances in Neural Information Processing Systems*, volume 33, pages 18330–18341. Curran Associates, Inc.

A Evaluating PoWER-BERT in Single Instance Mode

Due to the static structure of PoWER-BERT, the speedup ratios reported in Goyal et al. (2020) are based on wall time acceleration with batch-wise inference procedure. This means that some inputs might need extra padding to make all inputs with the same token length. However, since our approach and other dynamic approaches are based on single instance inference, in our procedure inputs are fed without being padded. To even out this discrepancy, we apply a single instance flops computation on the PoWER-BERT, which means we compute the computational cost for all input lengths that appear in the test dataset. Some instnaces may have shorter input length than some values in the resulting retention configuration (number of tokens that are retained in each layer). To overcome this issue, we update the retention configuration by selecting the minimum between the input length and each layers' number of tokens retained, to build a new retention configuration for each input length. For instance, if the retention configuration trained model on a given task be (153, 125, 111, 105, 85, 80, 72, 48, 35, 27, 22, 5), for an input with 75 tokens length, the new configuration which is used for speedup computation will be: (75, 75, 75, 75, 75, 75, 72, 48, 35, 27, 22, 5).

B AdapLeR Training Hyperparameters

For the initial step of finetuning BERT, we used the hyperparameters in Table 3. For both finetuning and training with length reduction, we employed an AdamW optimizer (Loshchilov and Hutter, 2019) with a weight decay rate of 0.1, warmup proportion 6% of total training steps and a linear learning rate decay which reaches to zero at the end of training.

For the adaptive length reduction training step, we also used the same hyperparameters in Table 3 with two differences: Since MRPC and CoLA have small training sets, to prolong the gradual soft-removal process, we increased the training duration to 10 epochs. Moreover, we increase the learning rate to 3e-5. Other hyperparameters are stated in Table 4. To set a trend for λ, it needs to start from a small but effective value ($10 < \lambda < 100$) and grow exponentially per each epoch to reach an extremely high amount at the end of the training to mimic a hard removal function ($1e{+}5 < \lambda$). Hence, datasets with the same amount of training epochs have similar λ trends.

Dataset	Epoch	LR	MaxLen.	BSZ
SST-2	5	$2e$-5	64	32
IMDB	5	$2e$-5	512	16
CoLA	5	$2e$-5	64	32
MRPC	5	$2e$-5	128	32
MNLI	3	$2e$-5	128	32
QNLI	5	$2e$-5	128	32
AG's News	5	$2e$-5	128	32

Table 3: Hyperparameters in each dataset; LR: Learning rate; BSZ: Batch size; MaxLen: Maximum Token Length

Dataset	γ	ϕ	λ
SST-2	5e-3	5e-4	10^{Epoch}
IMDB	5e-3	5e-4	10^{Epoch}
CoLA	3e-3	3e-2	10×3^{Epoch}
MRPC	3e-2	5e-2	10×3^{Epoch}
MNLI	5e-3	5e-4	50^{Epoch}
QNLI	5e-3	1e-4	10^{Epoch}
AG's News	1e-1	1e-1	10^{Epoch}

Table 4: ALR hyperparameters in each dataset; Since λ increases expoonentially on each epoch the coorresponding formula is written.