



# AdapLeR: Speeding up Inference by Adaptive Length Reduction

Ali Modarressi\* Hosein Mohebbi\* Mohammad Taher Pilehvar

 Iran University of Science and Technology, Iran

 Cognitive Science and AI, Tilburg University, Netherlands

 Tehran Institute for Advanced Studies, Khatam University, Iran

m\_modarressi@comp.iust.ac.ir

h.mohebbi@uvt.nl

mp792@cam.ac.uk

## 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 method 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 22x during inference time without much sacrifice in performance. 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 a significantly lower false positive rate in generating rationales. Our code is freely available at <https://github.com/amodaresi/AdapLeR>.

## 1 Introduction

While large-scale pre-trained language models exhibit remarkable performances on various NLP benchmarks, their excessive computational costs and high inference latency have limited their usage in resource-limited settings. In this regard, there have been various attempts at improving the efficiency of BERT-based models (Devlin et al., 2019), including knowledge distillation (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., 2020b; Liao et al., 2021; Xin et al., 2020; Liu et al., 2020; Xin et al., 2021; Sun et al., 2021; Eyzaquirre 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., 2020b), trained confidence predictors (Xin et al., 2021), or the number of agreements between predictions of intermediate classifiers (Zhou et al., 2020).

Most of these input-adaptive 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 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

\* Equal Contribution.

† Work done as a Master’s student at IUST.

come at the cost of reducing 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 weights. 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 suboptimal solution (Ye et al., 2021).

In this work, we introduce **Adaptive Length Reduction (AdapLeR)**. Instead of relying on attention weights, our method trains 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 in contrast to our approach 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 repre-

sentations 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  $(x_1, \dots, x_n)$  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:

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

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

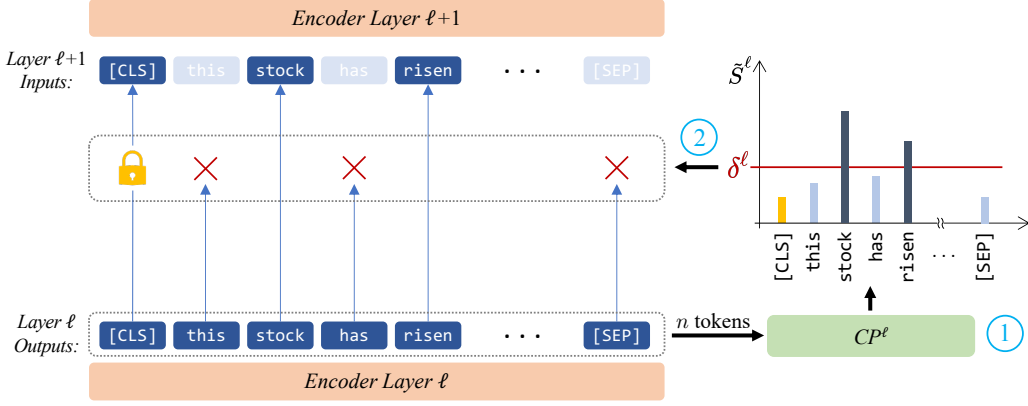


Figure 1: To reduce the inference computation, in each layer (1) the attribution score of the token representation is estimated and (2) based on a reduced uniform-level threshold ( $\delta^\ell = \eta^\ell/n$ ) token representations with low importance score are removed. Since the final layer’s classifier 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.

specific input feature. Despite having been widely utilized in other fields earlier (Ancona et al., 2018; 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). These methods are based on computing the first-order derivative of the output logit  $y_c$  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., 2018), 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 *gradient* $\times$ *input* (Bastings and Filippova, 2020), also known as *saliency*, which also considers the direction of input vectors to determine the most important tokens. 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):

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

### 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 have been recently shown

as a reliable criterion in measuring token’s contributions (Bastings and Filippova, 2020; Pascual et al., 2021). In Section 5.1 we will show results for 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 elaborate on how we leverage these predictors during inference (Section 3.2) and training (Section 3.3).

#### 3.1 Contribution Predictor

Computing gradients during inference is problematic as backpropagation computation prolongs inference time, which is contrary to our main goal. To circumvent this, we simply add a CP after each layer  $\ell$  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  $\ell$ . As the sum of these scores is equal to one, a uniform level indicates that all tokens contribute equally to the prediction and should be retained. On the other hand, the lower-scoring tokens could be viewed as unnecessary tokens if the contribution scores are concentrated only on a subset of tokens. Given that the final classification head uses the last hidden state of the [CLS] token, we preserve this token’s representation in all layers. Despite preserving this, other tokens might be removed from a layer when [CLS] has a significantly high estimated contribution score than others. Based on this intuition, we define a cutoff threshold based on the uniform level as:  $\delta^\ell = \eta^\ell \cdot 1/n$  with  $0 < \eta^\ell \leq 1$  to distinguish important tokens. Tokens are considered important if their contribution score exceeds  $\delta$  (which is a value equal or smaller than the uniform score). Intuitively, a larger  $\eta$  provides a higher  $\delta$  cutoff level, thereby dropping a larger number of tokens, hence, yielding more speedup. The value of  $\eta$  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  $\eta$ . We train each layer’s  $\eta^\ell$  using an auxiliary training objective, which allows the model to adjust the cutoff value to control the speedup-performance tradeoff. 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 fine-tuning, saliency extraction, and adaptive length re-training. In the first phase, we simply fine-tune the backbone model (BERT) on a given target task. We then extract the saliencies of three top-performing checkpoints from the fine-tuning process and compute the average of them to mitigate potential inconsistencies in saliency scores (cf. Section 2.2).

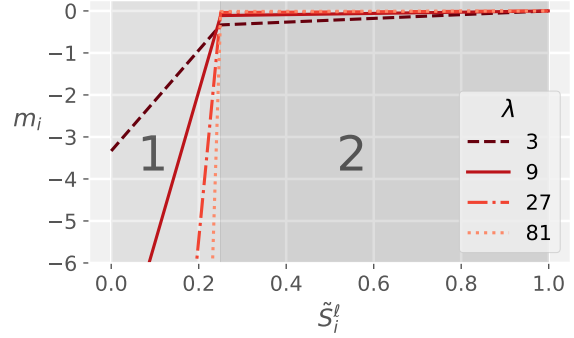


Figure 2: The soft-removal function plotted with  $\lambda \in \{3, 9, 27, 81\}$  and  $\delta^\ell = 0.25$ . As  $\lambda$  increases, the removal region (1) gets steeper while the other zone (2), which is almost horizontal, approaches the zero level.

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

**Soft-removal function.** During training, if tokens are immediately dropped similarly to the inference mode, the effect of dropping tokens cannot be captured using a gradient backpropagation procedure. Using batch-wise training in this scenario will also be problematic as the structure will vary with each example. Hence, inspired by the padding mechanism of self-attention models (Vaswani et al., 2017) we introduce a new procedure that gradually masks out less contributing token representations. In each layer, after predicting contribution scores, instead of instantly removing the token representations, we accumulate a negative mask to the attention mask vector  $M$  using a soft-removal function:

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

This function consists of two main zones (Figure 2). In the first term, the less important tokens with scores lower than the threshold ( $\delta^\ell$ ) are assigned higher negative masking as they get more distant



from  $\delta$ . The slope is determined by  $\lambda_{adj} = \lambda/\delta$ , where  $\lambda$  is a hyperparameter that is increased exponentially after each epoch (e.g.,  $\lambda \leftarrow 10 \times \lambda$  after finishing each epoch). Increasing  $\lambda$  makes the soft-removal function stronger and more decisive in masking the representations. To avoid under-going 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  $\delta^\ell$  threshold. Consider a scenario in which  $\eta^\ell$  sharply drops, causing most of  $\hat{S}_i^\ell$  get over the  $\delta^\ell$  threshold. In this case, the non-zero value in the second term of Equation 4, which facilitates optimizing  $\eta^\ell$ .

**Training the Contribution Predictors.** The CPs are trained by an additional term which is based on the KL-divergence<sup>1</sup> of each layer’s CP output with the extracted saliencies. The main training objective is a minimization of the following loss:

$$\mathcal{L} = \mathcal{L}_{CE} + \gamma \mathcal{L}_{CP} \quad (5)$$

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

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

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. As we leverage the representation of the [CLS] token in the last layer for classification, this token acts as a pooler and gathers information about the context of the input. In other words, 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  $\theta^\ell$ :

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

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  $\theta$  push the

<sup>1</sup>Inclusive KL loss. Check Appendix A.

CPs to shift the contribution towards the [CLS] token to gather most of the task-specific information and avoids carrying redundant tokens through the model.

**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  $[-\infty, 0]$  to a  $[0$  (fully masked/removed), 1 (fully retained)] bound.

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

Equation 8 demonstrates how the length loss is computed inside the model and how it is added to the main classification loss. During training, we assign a separate optimization process which tunes  $\eta$  and  $\theta$  to adjust the thresholds and the amount of [CLS] pooling<sup>2</sup> alongside with the 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 AdapLeR on inference speedup, we selected eight various 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, MRPC (Dolan and Brockett, 2005) for paraphrase, AG’s News (Zhang et al., 2015) for topic classification, DBpedia (Lehmann et al., 2015) for knowledge extraction, MNLI (Williams et al., 2018) for NLI,

<sup>2</sup>Since  $\theta$  is not in the computational DAG, we employed a dummy variable inside the model. See Appendix B.

Model	SST-2		IMDB		HateXplain		MRPC		MNLI		QNLI		AG’s news		DBpedia	
	Acc.	Speedup	Acc.	Speedup	Acc.	Speedup	F1.	Speedup	Acc.	Speedup	Acc.	Speedup	Acc.	Speedup	Acc.	Speedup
BERT	92.7	1.00x	93.8	1.00x	68.3	1.00x	87.5	1.00x	84.2	1.00x	90.3	1.00x	94.4	1.00x	99.3	1.00x
DistilBERT	92.2	2.00x	92.9	2.00x	68.2	2.00x	88.0	2.00x	81.8	2.00x	88.1	2.00x	94.2	2.00x	99.3	2.00x
PoWER-BERT	92.1	1.18x	92.2	1.70x	66.9	2.69x	88.0	1.07x	82.9	1.10x	89.7	1.23x	92.1	12.50x	98.1	14.80x
TR-BERT	92.1	1.46x	93.2	2.90x	67.9	2.23x	81.9	1.16x	84.8	1.00x	89.0	1.09x	93.2	10.20x	98.9	10.01x
<b>AdapLeR</b>	92.3	1.49x	91.7	3.21x	68.6	4.73x	87.6	1.27x	82.9	1.42x	89.3	1.47x	92.5	17.10x	98.9	22.23x

Table 1: Comparison of our proposed method (AdapLeR) with other baselines in eight classification tasks in terms of performance and speedup. For each dataset the corresponding metric has been reported (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.

QNLI (Rajpurkar et al., 2016) for question answering, and HateXplain (Mathew et al., 2021) for hate speech.<sup>3</sup> 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 predictions to the evaluation server.

## 4.2 Experimental Setup

As our baseline, we report results for the pre-trained BERT model (base-uncased) (Devlin et al., 2019) which is also the backbone of AdapLeR. We also compare against three other approaches: DistilBERT (uncased) (Sanh et al., 2019) as a static compression method, PoWER-BERT and TR-BERT as two strong length reduction methods (cf. Sec. 1). We used the provided implementations and suggested hyperparameters<sup>4</sup> to train these baselines. To fine-tune the backbone model, we used same hyperparameters over all tasks (see Section D for details). 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 four hyperparameters ( $\gamma, \phi, \lambda, \beta$ )<sup>5</sup> which are involved in the training process. Among these,  $\phi$  and  $\gamma$  are the primary terms that have considerable effects on AdapLeR’s downstream performance and speedup. 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. We used grid search to

<sup>3</sup>See the statistics of datasets in Table 5 in Appendix.

<sup>4</sup>Since some of the datasets were not used originally, we had to search the hyperparameters based on the given ranges.

<sup>5</sup>Note that  $\theta$  and  $\eta$  are trainable terms that are tuned by the model during training.

find the optimal values for these two terms, while keeping the other hyperparameters constant over all datasets. Hyperparameter selection is further discussed in Section D.

**FLOPs Computation.** We followed Ye et al. (2021) and Liu et al. (2020) and measured computational complexity in terms of FLOPs, i.e., the number of floating-point operations (FLOPs) in a single inference procedure. This allows us to assess models’ speedups independently of their operating environment (e.g., CPU/GPU). The total FLOPs of a given model is a summation of the measured FLOPs over all test examples. Then, a model’s speedup can be defined as the total FLOPs measured on BERT (our baseline) divided by the corresponding model’s total FLOPs. To have a fair comparison, we also computed FLOPs for PoWER-BERT in a single instance mode, described in Section C.

## 4.3 Results

Table 1 shows performance and speedup for AdapLeR and other comparison models across eight different datasets. While preserving the same level of performance, AdapLeR outperforms other techniques in terms of speedup across all tasks (ranging from +0.2x to +7.4x compared to the best model in each dataset).

It is noteworthy that the results also reveal some form of dependency on the type of the tasks. Some tasks may need less amount of contextualism during inference and could be classified by using only a fraction of input tokens. For instance, in AG’s News, the topic of a sentence might be identifiable with a single token (e.g., *soccer*  $\rightarrow$  Topic: Sports, see Figure 6 in the Appendix for an example). PoWER-BERT adopts attention weights in its token selection which requires at least one layer of computation to be determined, and TR-BERT ap-

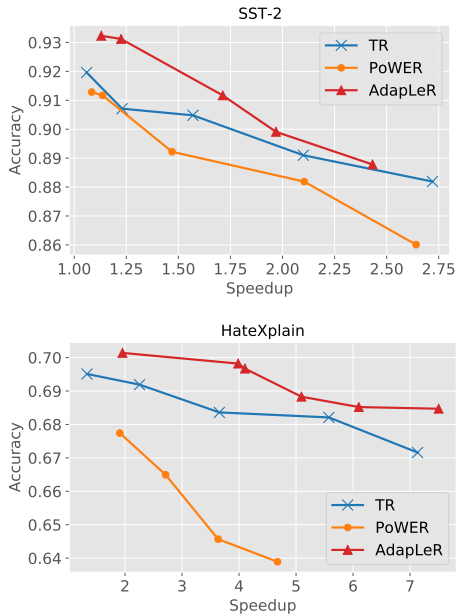


Figure 3: Accuracy-Speedup trade-off curve for AdapLeR and two other state-of-the-art reduction methods; TR: TR-BERT, PoWER: PoWER-BERT on two different tasks.

plies token elimination only in two layers to reduce the training search space. In contrast, our procedure performs token elimination for all layers of the model, enabling a more effective removal of redundant tokens. On the other hand, we observe that TR-BERT and PoWER-BERT lack any speedup gains for tasks such as QNLI, MNLI, and MRPC which need a higher degree of contextualism during inference. However, AdapLeR can offer some speedups even for these tasks.

**Speedup-Performance Tradeoff.** To provide a closer look at the efficiency of AdapLeR in comparison with the other state-of-the-art length reduction methods, we illustrate speedup-accuracy curves in Figure 3. We provide these curves for two tasks in which other length reduction methods show comparable speedups to AdapLeR. For each curve, the points were obtained by tuning the most influential hyperparameters of the corresponding model. As we can see, AdapLeR significantly outperforms the other two approaches in all two tasks. An interesting observation here is that the curves for TR-BERT and AdapLeR are much higher than that of PoWER-BERT. This can be attributed to the input-adaptive procedure employed by the former two methods for determining the number of reduced tokens (whereas PoWER-BERT adopts a fixed retention configuration in token elimination).

Strategy	Movie Reviews		MultiRC	
	Acc.	Speedup	Acc.	Speedup
Full input	93.3	1.0x	67.7	1.0x
Human rationale	96.7	3.7x	76.6	4.6x
Saliency	<b>92.3</b>	3.7x	<b>66.4</b>	4.4x
Attention ALL	78.5	3.7x	62.9	4.4x
Attention [CLS]	70.3	3.7x	63.7	4.4x

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 best corresponding strategy for each task (the closest performance to the upper bound).

## 5 Analysis

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

### 5.1 Rationale as an Upper Bound

A natural question that arises when dealing with token pruning is that of *importance measure*: what is the most appropriate criterion for assessing the relative importance of tokens within a sentence? We resort to human rationale as a reliable upper bound for measuring token importance. 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 tasks in the benchmark, 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 of the passage. Whereas the latter contains a passage, a question, and multiple candidate answers, which is cast as a binary classification task of passage/question/answer triplets in the ERASER benchmark.

In order to verify the reliability of human rationales, we fine-tuned BERT based on the rationales only, i.e., by excluding those tokens that are not highlighted as being important in the input. In Table 2, the first two rows show the performance of BERT on the two tasks with full input and with human rationales only. We see that fine-tuning merely

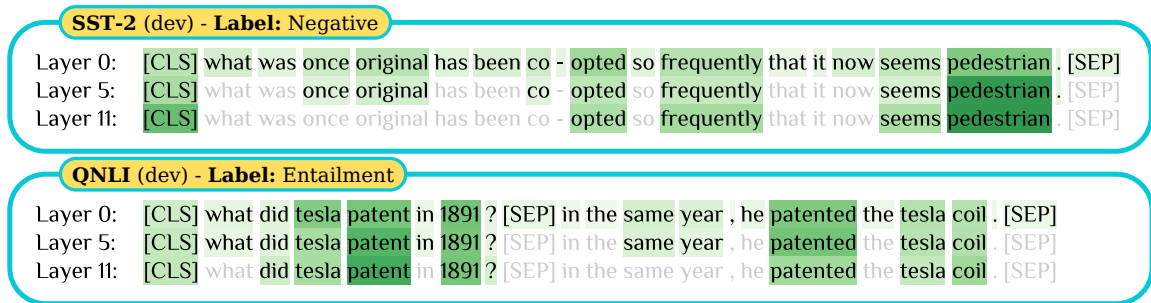


Figure 4: The illustration of contribution scores obtained by CPs in three different layers of the model for two input examples from SST-2 (sentiment) and QNLI (Question-answering NLI) tasks. The contribution scores are shown by color intensity. Only the highlighted token representations are processed in each layer. See more full-layer plots in the appendix 6.

on rationales not only yields less computation cost, but also results in a better performance when compared with the full input setting. Obviously, human annotations are not available for a whole range of downstream NLP tasks; therefore, this criterion is infeasible in practice and can only be viewed as an upper bound for evaluating different strategies in measuring token importance.

## 5.2 Saliency vs. Attention

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. 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 brings about two advantages. Firstly, representations in the embedding layer are non-contextualized, allowing us to measure the importance of each token independently from the others. Secondly, the backpropagation of gradients through layers to the beginning of the model provides us with aggregated values for the relative importance of each token based on the entire model. Similarly, we aggregated the 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 the attention weight matrix in each layer is multiplied with the preceding ones 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 sentence, 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 comparison, 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. Computing these weights is convenient as they are already computed during the forward pass, whereas computing saliency requires an additional backpropagation step. Note that in our approach, saliency scores are easily estimated within inference time by the pre-trained CPs.

## 5.3 Contribution Predictor Evaluation

In this section we validate our Contribution Predictors in selecting the most contributed tokens. Figure 4 illustrates two examples from the SST-2 and QNLI datasets in which CPs identify and gradually drop the irrelevant tokens through layers, finally focusing mostly on the most important token representations; *pedestrian* (adjective) in SST-2 and *tesla coil* in the passage part of QNLI (both of which are highly aligned with human rationale).

In order to quantify the extent to which AdapLeR’s CPs can preserve rationales without requiring direct human annotations in an unsuper-



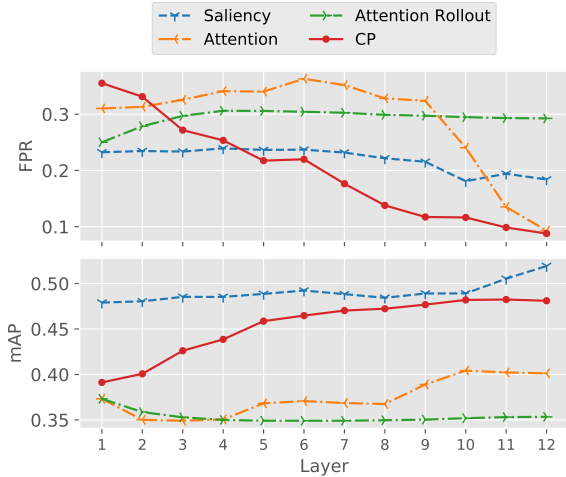


Figure 5: Agreement with human rationales in terms of mean Average Precision and False Positive Rate for Contribution Predictor (CP) and three alternative techniques.

vised manner we carried out the following experiment. To investigate the effectiveness of trained CPs in predicting human rationales we computed the output scores of CPs in AdapLeR for each token representation in each layer. We also fine-tuned a BERT model on the Movie Review dataset and computed layer-wise raw attention, attention rollout, and saliency scores for each token representation. 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 uniform-level threshold (i.e.,  $1/n$ ) to reach a binary score indicating tokens selected in each layer.

As for evaluation, we used the Average Precision (AP) and False Positive Rate (FPR) metrics 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. We used soft scores for computing AP and binary scores for computing FPR.

Figure 5 shows the agreement between human rationales and the selected tokens based on the two metrics. In comparison with the other widely used strategies for selecting important tokens, such as saliency and attention, our CPs have significantly less false positive rate in preserving ratio-

nales through layers. Despite having similar FPRs at the final layer, CP is preferable to attention in that it can better identify rationales at the earlier layers, allowing the model to combine the most relevant token representations when building the final one. This in turn results in better performance, as was also shown in the previous experiment in Section 5.2. Also, we see that the curve of mAP for saliency is consistently higher than other strategies in terms of alignment with human rationales which supports our choice of saliency as a measure for token importance.

Finally, we note that there is a line of research that attempts at guiding 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). As a by-product of the contribution estimation process, our trained CPs are able to generate these rationales at inference without the need for human-generated annotations.

## 6 Conclusion

In this paper, we introduced AdapLeR, a novel method that accelerates inference by dynamically identifying and dropping less contributing token representations through layers of BERT-based models. Specifically, AdapLeR accomplishes this by training a set of Contribution Predictors based on saliencies extracted from a fine-tuned model and applying a gradual masking technique to simulate input-adaptive token removal during training. Empirical results on eight diverse text classification tasks show considerable improvements over existing 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 those tasks that require all token representations to be present in the final layer of the model (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.

## Broader Impact

Using our proposed method, pre-trained language models can use fewer FLOPs, reducing energy use and carbon emissions (Schwartz et al., 2020a).

## 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. 2018. [Towards better understanding of gradient-based attribution methods for deep neural networks](#). In *International Conference on Learning Representations*.
- 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](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8189–8200, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- 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](#). In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.
- 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 Wiegrefe, 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.
- Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. 2015. Dbpedia—a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic web*, 6(2):167–195.
- 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*.
- Binny Mathew, Punyajoy Saha, Seid Muhie Yimam, Chris Biemann, Pawan Goyal, and Animesh Mukherjee. 2021. Hatexplain: A benchmark dataset for explainable hate speech detection. In *AAAI*.
- 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 Rush. 2020. Movement pruning: Adaptive sparsity by fine-tuning. *Advances in Neural Information Processing Systems*, 33:20378–20389.



- Roy Schwartz, Jesse Dodge, Noah Smith, and Oren Etzioni. 2020a. Green ai. *Communications of the ACM*, 63:54 – 63.
- Roy Schwartz, Gabriel Stanovsky, Swabha Swayamdipta, Jesse Dodge, and Noah A. Smith. 2020b. [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, Łukasz 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. [Spaten: Efficient sparse attention architecture with cascade token and head pruning](#). *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 97–110.
- 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 Inclusive KL Loss Consideration

We opted for an inclusive KL loss since CPs should be trained to cover all tokens considered important by saliency and not to be mode seeking (i.e., covering a subset of high contributing tokens considered by the saliency scores.). Suppose an exclusive KL is selected. Due to the limited learning capacity of the CP and miscalculation possibility from the saliency, the CP may be trained to maximize its contribution on noninformative tokens. While in an inclusive setting, it trains to extend its coverage over all high-saliency tokens.

Additionally, our initial research indicated that using a symmetric loss (e.g. Jensen-Shannon divergence) would produce similar results but with a significantly longer convergence time.

## B Optimization of $\theta$

In Section 3.3, we introduced  $\theta^\ell$  as a trainable parameter that increases the saliency score of [CLS]. We can deduce from Equations 6 and 7 that this parameter does not exist in the model’s computational DAG and we need to compute the derivative of  $\tilde{S}^\ell$  w.r.t.  $\theta^\ell$  to train this parameter. Hence, first we assume that  $\tilde{S}^\ell$  is a close estimate of  $\hat{S}^\ell$  (due to the CPs’ training objective). Second, using a dummy variable  $\theta_d^\ell$ —that is involved in the computational graph and is always equal to 1—we reformulate  $\tilde{S}^\ell$ :

$$\hat{S}_i^\ell \approx \tilde{S}_i^\ell = \frac{\theta_d^\ell \tilde{S}_1^\ell \mathbf{1}[i = 1] + \tilde{S}_i^\ell \mathbf{1}[i > 1]}{\theta_d^\ell \tilde{S}_1^\ell + \sum_{i=2}^n \tilde{S}_i^\ell} \quad (9)$$

This reformulation is valid due to  $\theta_d^\ell = 1$  and  $\sum_{i=1}^n \tilde{S}_i^\ell = 1$ . Now we compute the partial derivative w.r.t.  $\theta_d^\ell$  which is the gradient that is computed in the backpropagation:

$$\frac{\partial \tilde{S}_i^\ell}{\partial \theta_d^\ell} = \frac{\tilde{S}_1^\ell (\sum_{i=2}^n \tilde{S}_i^\ell \mathbf{1}[i = 1] - \tilde{S}_i^\ell \mathbf{1}[i > 1])}{(\theta_d^\ell \tilde{S}_1^\ell + \sum_{i=2}^n \tilde{S}_i^\ell)^2} \quad (10)$$

By knowing that  $\theta_d^\ell = 1$ :

$$\frac{\partial \tilde{S}_i^\ell}{\partial \theta_d^\ell} = \tilde{S}_1^\ell ((1 - \tilde{S}_1^\ell) \mathbf{1}[i = 1] - \tilde{S}_i^\ell \mathbf{1}[i > 1]) \quad (11)$$

Now using our initial assumption ( $\hat{S}_i^\ell \approx \tilde{S}_i^\ell$ ), we can substitute  $\tilde{S}_i^\ell$  with  $\hat{S}_i^\ell$  based on Equation 7:

$$\begin{aligned} \frac{\partial \tilde{S}_i^\ell}{\partial \theta_d^\ell} &= \hat{S}_1^\ell ((1 - \hat{S}_1^\ell) \mathbf{1}[i = 1] - \hat{S}_i^\ell \mathbf{1}[i > 1]) \\ &= \frac{\theta^\ell S_1^\ell (\sum_{i=2}^n S_i^\ell \mathbf{1}[i = 1] - S_i^\ell \mathbf{1}[i > 1])}{(\theta^\ell S_1^\ell + \sum_{i=2}^n S_i^\ell)^2} \end{aligned} \quad (12)$$

In addition, the gradient of  $\hat{S}_i^\ell$  w.r.t.  $\theta^\ell$  is as follows (cf. Equation 7):

$$\frac{\partial \hat{S}_i^\ell}{\partial \theta^\ell} = \frac{S_1^\ell (\sum_{i=2}^n S_i^\ell \mathbf{1}[i = 1] - S_i^\ell \mathbf{1}[i > 1])}{(\theta^\ell S_1^\ell + \sum_{i=2}^n S_i^\ell)^2} \quad (13)$$

By comparing Equations 12 and 13, these derivatives are related with a term of  $\theta^\ell$ :

$$\frac{\partial \hat{S}_i^\ell}{\partial \theta^\ell} \approx \frac{\partial \tilde{S}_i^\ell}{\partial \theta^\ell} = \frac{1}{\theta^\ell} \frac{\partial \tilde{S}_i^\ell}{\partial \theta_d^\ell} \quad (14)$$

Therefore, during training, we can compute the gradient w.r.t. the dummy variable  $\theta_d^\ell$  and then divide it by  $\theta^\ell$ .

### C 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 instances 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).

### D AdapLeR Training Hyperparameters

For the initial step of fine-tuning BERT, we used the hyperparameters in Table 3. For both fine-tuning 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  $\lambda$ , it needs to start from a small but effective value ( $10 < \lambda < 100$ ) and grow exponentially per each epoch to reach an ex-

Dataset	Epoch	LR	MaxLen.	BSZ
SST-2	5	$2e-5$	64	32
IMDB	5	$2e-5$	512	16
HateXplain	5	$3e-5$	72	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
DBpedia	3	$2e-5$	128	32

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

tremely 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  $\lambda$  trends.

Dataset	$\gamma$	$\phi$	$\lambda$
SST-2	$5e-3$	$5e-4$	$10^{Epoch}$
IMDB	$5e-3$	$5e-4$	$10^{Epoch}$
HateXplain	$5e-2$	$2e-2$	$50^{Epoch}$
MRPC	$3e-2$	$5e-2$	$10 \times 3^{Epoch}$
MNLI	$5e-3$	$5e-4$	$50^{Epoch}$
QNLI	$5e-3$	$1e-4$	$10^{Epoch}$
AG’s News	$1e-1$	$1e-1$	$10^{Epoch}$
DBPedia	$1e-1$	$1e-1$	$50^{Epoch}$

Table 4: AdapLeR hyperparameters in each dataset; Since  $\lambda$  increases exponentially on each epoch the corresponding formula is written.

### E Statistics of Datasets

### F Additional Qualitative Examples



Figure 6: The illustration of contribution scores obtained by CPs in each layers of the model for different input examples from QNLI (Question-answering NLI), SST-2 (sentiment), and AG’s news (topic classification) tasks. The color intensity indicates the degree of contribution scores. Only the highlighted token representations are processed in each layer

Task	Number of Examples		Number of Tokens
	Train	Test	Mean / Median
SST-2	67349	1821	14 / 11
IMDB	25000	25000	275 / 233
HateXplain	15383	1924	30 / 27
MRPC	3668	1725	53 / 53
MNLI	392702	9796 <sup>†</sup> / 9847 <sup>‡</sup>	40 / 37
QNLI	104743	5463	50 / 47
AG’s News	120000	7600	53 / 51
DBPedia	560000	70000	64 / 64

Table 5: The statistics of datasets: number of training and test examples and average and median of sequence length (number of tokens) of test examples based on BERT’s tokenizer. <sup>†</sup> and <sup>‡</sup> indicate *matched* and *mis-matched* versions of MNLI test split, respectively.