AD-DROP: Attribution Driven Dropout for Robust Language Model Finetuning

Anonymous Author(s)
Affiliation
Address
email

Abstract

Finetuning large pretrained language models on downstream tasks is apt to suffer from overfitting when limited training data is available. While dropout proves to be an effective antidote by randomly dropping a proportion of units, existing research has not examined its effect on the self-attention mechanism. In this paper, we investigate this problem through self-attention attribution and find that dropping attention positions with low attribution scores can accelerate training and increase the risk of overfitting. Motivated by this observation, we propose Attribution Driven Dropout (AD-DROP), which randomly discards high attribution positions to encourage the model to make predictions by relying more on low attribution positions to reduce overfitting. We also develop a cross-tuning strategy to alternate finetuning and AD-DROP to avoid dropping high attribution positions excessively. Extensive experiments on the GLUE benchmark show that AD-DROP not only effectively mitigates overfitting on small datasets but also leads to performance improvements on large datasets. These results confirm the success of AD-DROP as a strategic regularizer to prevent overfitting during finetuning.

1 Introduction

Pretraining large language models (PrLMs) on massive unlabeled corpora and finetuning them on downstream tasks has become a new paradigm [1–3]. Their success can be largely attributed to the self-attention mechanism [4], yet these self-attention networks are often redundant [5, 6] and tend to cause overfitting when finetuned on downstream tasks due to the mismatch between their overparameterization and limited annotated data [7–13]. To address this issue, various regularization techniques such as data augmentation [14, 15], adversarial training [16, 17]), and dropout-based methods [11, 13, 18] have been developed. Among them, dropout-based methods are widely adopted for their simplicity and effectiveness. Dropout [19], which randomly discards a proportion of units, is at the core of dropout-based methods. Recently, several variants of dropout have been proposed, such as Concrete Dropout [20], DropBlock [21], and AutoDropout [22]. However, these variants generally follow the vanilla dropout to randomly drop units during training and pay little attention to the effect of dropout on self-attention. In this paper, we seek to fill this gap from the perspective of self-attention attribution [23] and aim to reduce overfitting when finetuning PrLMs.

Attribution [24] is an interpretability method that attributes model predictions to input features via saliency measures such as gradient [25, 26]. It is also used to explain the influence patterns of self-attention in recent literature [23, 27, 28]. Our prior experiment of self-attention attribution (Section 2.2) reveals that attention positions are not equally important in preventing overfitting, and

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

dropping low attribution positions is more likely to cause overfitting than discarding high attribution positions. This observation suggests that attention positions should not be treated the same in dropout.

Motivated by the above argument, we propose **A**ttribution **D**riven **Drop**out (AD-DROP) to better finetune PrLMs based on self-attention attribution. The general idea of AD-DROP is to drop a set of self-attention positions with high attribution scores. We illustrate the difference between vanilla dropout and AD-DROP by their attention maps in Figure 1. When finetuning a PrLM on a batch of training samples, AD-DROP involves four steps. First, predictions are made through a forward computation without dropping any attention position. Second, we compute the attribution score of each attention position by gradient [25] or integrated gradient [26] attribution methods. Third, we randomly sample a set of positions with high attribution scores and generate a masking matrix for each attention map. Finally, the masking matrices are applied to the next forward computation to make predictions for backpropagation. AD-DROP can be regarded as a strategic dropout regularizer that forces the model to make predictions by relying more on low attribution positions to reduce overfitting. Nevertheless, excessive neglect of high attribution positions would leave insufficient information for training. For this reason, we further propose a cross-tuning strategy that performs finetuning and AD-DROP alternately to improve the training stability.

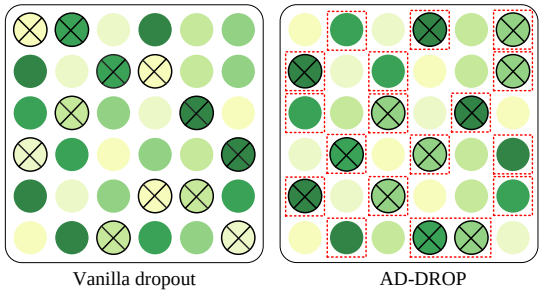


Figure 1: Attention maps of vanilla dropout and our AD-DROP. Darker attention positions indicate higher attribution scores, and crossed circles mean dropped attention positions. Red-dotted boxes refer to candidate discard regions with high attribution scores. Unlike vanilla dropout which randomly masks attention positions, AD-DROP focuses on dropping high attribution positions in candidate discard regions.

To verify the effectiveness of AD-DROP, we conduct extensive experiments with BERT [1] and RoBERTa [2] on eight downstream tasks of the GLUE benchmark [29]. Experimental results show that the two models tuned with AD-DROP obtain remarkable improvements over the original finetuning approach. Specifically, BERT achieves an average improvement of 1.98/0.87 points on the dev/test sets while RoBERTa achieves an average improvement of 1.29/0.62 points. Moreover, ablation studies and analysis demonstrate that gradient-based attribution [25, 26] is a more suitable saliency measure for implementing AD-DROP than directly using attention weights or random sampling. The cross-tuning strategy also proves to play a crucial role in improving training stability.

To sum up, this work reveals that self-attention positions are not equally important for dropout when finetuning PrLMs. Arguably, low attribution positions are more difficult to optimize than high attribution positions, and dropping these positions tends not to relieve but accelerate overfitting. This leads to a novel dropout regularizer, AD-DROP, driven by self-attention attribution. Although proposed for self-attention units, AD-DROP can be potentially extended to other units as dropout.

2 Methodology

2.1 Preliminaries

Since Transformers [4] are the backbone of PrLMs, we first review the details of self-attention in Transformers and self-attention attribution [23]. Let $\mathbf{X} \in \mathbb{R}^{n \times d}$ be the input of a Transformer block, where n is the sequence length and d is the embedding size. Self-attention in this block first maps $\mathbf{X}$ into three matrices $\mathbf{Q}_h$, $\mathbf{K}_h$ and $\mathbf{V}_h$ via linear projections as query, key, and value respectively for the h-th head. Then, the attention output of this head is calculated as:

$$\text{Attention}\left(\mathbf{Q}_h, \mathbf{K}_h, \mathbf{V}_h\right) = \mathbf{A}_h \mathbf{V}_h = \text{softmax}\left(\frac{\mathbf{Q}_h {\mathbf{K}_h}^{\mathrm{T}}}{\sqrt{d_k}} + \mathbf{M}_h\right) \mathbf{V}_h, \tag{1}$$

where $\sqrt{d_k}$ is a scaling factor. $\mathbf{M}_h$ is the masking matrix to apply dropout in self-attention, and elements in $\mathbf{M}_h$ will be set to $-\infty$ if the corresponding connections are masked and to 0 otherwise.

Based on the attention maps $\mathbf{A} = [\mathbf{A}_1, \mathbf{A}_2, \cdots, \mathbf{A}_H]$ for H attention heads, gradient attribution [25, 30] directly produces an attribution matrix $\mathbf{B_h}$ by computing the following partial derivative:

$$\mathbf{B}_h = \frac{\partial F_c(\mathbf{A})}{\partial \mathbf{A}_h}, \tag{2}$$

where $F_c(\cdot)$ denotes the logit output of the Transformer for class c.

To provide a theoretically more sound attribution method, Sundararajan et al. [26] propose integrated gradient, which is employed by Hao et al. [23] as a saliency measure for self-attention attribution. Specifically, Hao et al. [23] compute the attribution matrix $\mathbf{B}_h$ as:

$$\mathbf{B}_h = \frac{\mathbf{A}_h}{m} \odot \sum_{k=1}^{m} \frac{\partial F_c\left(\frac{k}{m}\mathbf{A}\right)}{\partial \mathbf{A}_h}, \tag{3}$$

where m is the number of steps for approximating the integration in integrated gradient, and $\odot$ is the element-wise multiplication operator. Despite its theoretical advantage over gradient attribution, integrated gradient requires m times more computational effort, which is especially expensive when it is applied to all the attention heads in Transformers. Moreover, our experiments in Section 3.4 show that gradient attribution achieves comparable performance with integrated gradient but requires much less computational cost, suggesting that gradient attribution is more desirable for AD-DROP.

2.2 Prior Attribution Experiments

To better motivate our work, we first conduct a prior experiment on MRPC [31] to investigate how different self-attention positions affect finetuning performance based on attribution results. `RoBERTa_base` [2] is used as the base model. To begin with, we first perform a forward computation of the model on each batch of training samples to obtain the logit output of each sample corresponding to the gold label. Then, we compute an attribution matrix $\mathbf{B}_h$ for the self-attention positions in the first layer[1] by gradient attribution with Eq. (2) and sort each row of the matrix. Finally, we randomly sample a set of attention positions with high or low attribution scores in each row to generate a masking matrix $\mathbf{M}_h$, which is fed into Eq. (1) to make the final predictions. After each epoch of training, we evaluate the model on the development set. Two baseline dropping strategies (i.e., dropping by random sampling and without dropping any position) are employed for comparison. We plot the loss curves of the model with these dropping strategies on both training and development sets in Figure 2. The observations are threefold. First, dropping low attribution positions makes the model fit the training data rapidly, whereas it performs poorly on the development set, indicating that the model is not properly optimized. Second, compared with the other dropping strategies, dropping high attribution positions reduces the fitting speed significantly. Third, random dropping only slightly reduces overfitting, compared to training without dropping. These observations suggest that attention positions are of different importance in preventing overfitting. We conjecture that low attribution positions are more difficult to optimize than high attribution positions. While dropping low attribution positions tends to accelerate overfitting, discarding high attribution positions helps reduce overfitting.

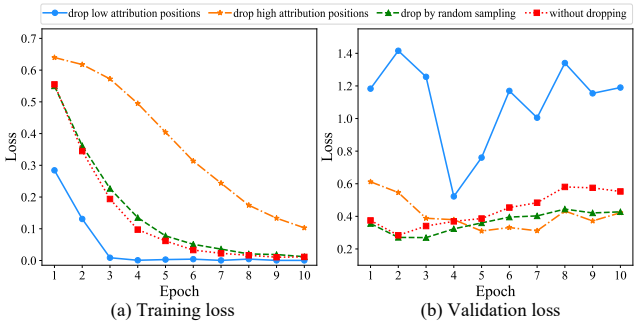


Figure 2: Results of training and validation losses when finetuning RoBERTa with different dropping strategies. The dropping rate is set to 0.3 if it applies.

[1] We provide more results and discussions in Appendix A.

2.3 Attribution Driven Dropout

Inspired by the observations in Section 2.2, we propose a novel regularizer, AD-DROP, to better prevent overfitting when adapting PrLMs to downstream tasks. The motivation of AD-DROP is to minimize the over-reliance of these models on particular features which affect their generalization. Formally, given training set $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$ of N samples, where x_i is the i-th sample and y_i is its label, the goal of AD-DROP is to finetune a PrLM $F(\cdot)$ of L layers on $\mathcal{D}$. Same as the vanilla dropout [19], AD-DROP is only applied during the training phase.

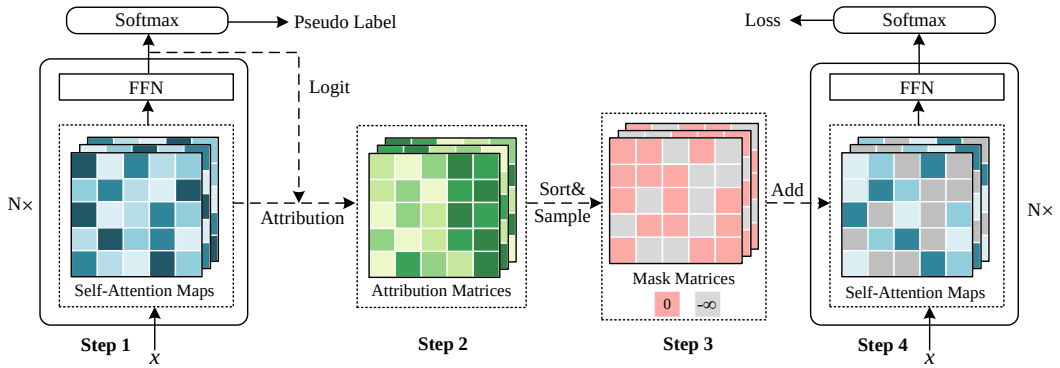


Figure 3: Illustration of AD-DROP in four steps.

As shown in Figure 3, the idea of AD-DROP can be described in four steps. First, we conduct a forward computation of the model to obtain the label with the highest probability as pseudo label. The reason we adopt pseudo label rather than gold label for attribution will be explained shortly. Specifically, for the input x_i with n tokens, we apply $F(\cdot)$ to encode it and obtain its pseudo label $\tilde{c}$:

$$\tilde{c} = \arg\max_{c} \left(p\left(c|x_i\right) \right), \tag{4}$$

where $p\left(c|x_i\right)$ is the probability of class c for x_i. After the forward computation, we also obtain a set of attention maps $\mathbf{A} = [\mathbf{A}_1, \mathbf{A}_2, \cdots, \mathbf{A}_H]$ for each layer according to Eq. (1).

Second, we compute the attribution matrices $\mathbf{B} = [\mathbf{B}_1, \mathbf{B}_2, \cdots, \mathbf{B}_H]$ for H heads according to Eq. (2). Specifically, the attribution matrix $\mathbf{B}_h$ for the h-th head is computed as:

$$\mathbf{B}_h = \frac{\partial F_{\tilde{c}}\left(\mathbf{A}\right)}{\partial \mathbf{A}_h}, \tag{5}$$

where $F_{\tilde{c}}\left(\mathbf{A}\right)$ is the logit output of the pseudo label $\tilde{c}$ before softmax [2].

Third, we generate a mask matrix $\mathbf{M}_h$ based on $\mathbf{B}_h$. To this end, we first sort each row of $\mathbf{B}_h$ in ascending order and obtain a sorted attribution matrix $\widehat{\mathbf{B}}_h$. Then, we define a candidate discard region $\mathbf{S}_h$, in which each element $s_{i,j}$ is defined as:

$$s_{i,j} = \begin{cases} 1, & b_{i,j} < \widehat{b}_{i,\mathrm{int}(n(1-p))} \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

where $b_{i,j}$ and $\widehat{b}_{i,j}$ are elements of $\mathbf{B}_h$ and $\widehat{\mathbf{B}}_h$, respectively, $\mathrm{int}\left(\cdot\right)$ is an integer function, and $p \in (0, 1)$ is used to control the size of the candidate discard region. Next, we apply dropout in the region to produce the mask matrix $\mathbf{M}_h$ as:

$$m_{i,j} = \begin{cases} -\infty, & (s_{i,j} + u_{i,j}) = 0 \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

where $u_{i,j} \sim \mathrm{Bernoulli}\left(1 - q\right)$ is an element of matrix $\mathbf{U}_h \in \mathbb{R}^{n \times n}$.

Finally, $\mathbf{M}_h$ is fed into self-attention of Eq. (1) for the second forward computation, and the final output is used to calculate the loss for backpropagation.

[2] The negative loss will be used for regression tasks such as STS-B.

Discussion The reasons that AD-DROP uses pseudo labels for attribution are twofold. First, adopting gold labels will divulge label information and lead to inconsistency between training and inference. Second, for misclassified samples in the first forward computation, AD-DROP with gold labels tends to continue to make incorrect predictions because high attribution attention positions derived from gold labels may be located in low attribution regions derived from pseudo labels. Therefore, dropping these positions does not help the model to correct wrong predictions, while AD-DROP with pseudo labels urges the model to rely on important features in the current pass and may correct the wrong predictions. The results with gold labels will be reported in Section 3.4.

2.4 Cross-Tuning Algorithm

We further design a cross-tuning algorithm to avoid dropping high attribution positions excessively when applying AD-DROP. The idea of cross-tuning is to execute the original finetuning and AD-DROP alternatively. Specifically, it performs the original finetuning at odd epochs and AD-DROP at even epochs. The overall process of cross-tuning is described in Algorithm 1, where Lines 3-5 are the original finetuning operations and Lines 7-9 describe the process of AD-DROP.

Algorithm 1 Cross-tuning

Input: shuffled training samples $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$, PrLM F with parameters $\mathbf{W}$
Output: updated parameters $\widetilde{\mathbf{W}}$
1: Initialize F with $\mathbf{W}$, $epoch = 1$
2: **while** not converged **do**
3: Calculate the prediction $p\left(y_i|x_i\right)$ and loss via forward computation.
4: **if** $epoch\%2 == 1$ **then**
5: Backpropagate the loss to update model parameters $\mathbf{W}$.
6: **else**
7: Perform AD-DROP by Eq. (4)-(7) to obtain mask matrices $\mathbf{M} = [\mathbf{M}_1, \mathbf{M}_2, \cdots, \mathbf{M}_H]$.
8: Calculate the new prediction $p\left(y_i|x_i\right)$ and new loss by feeding $\mathbf{M}$ into Eq. (1).
9: Backpropagate the new loss to update model parameters $\mathbf{W}$.
10: $epoch = epoch + 1$
11: **return** $\widetilde{\mathbf{W}} = \mathbf{W}$

3 Experiments

3.1 Datasets

We conduct experiments on eight tasks of the GLUE benchmark [29], including SST-2 [32], MNLI [33], QNLI [34], QQP [35], CoLA [36], STS-B [37], MRPC [31], and RTE [38]). The details of these datasets are introduced in Appendix B.1. The evaluation metrics are Matthew's Corrcoef (Mcc) [39] for CoLA, Pearson Corrcoef (Pcc) [40] for STS-B, and Accuracy (Acc) for the others.

3.2 Implementation Details

We implement our AD-DROP in Pytorch with the Transformers package [41]. We choose `bert-base-uncased` and `roberta-base` as the base models with default settings, and train them on GeForce RTX 3090 GPUs. We tune the learning rate in {1e-5, 2e-5, 3e-5} and the batch size in {16, 32, 64}. Following Miao et al. [17], we perform early stopping to select the number of training epochs. The two critical hyperparameters p and q are searched within $[0.1, 0.9]$ with step size 0.1. For integrated gradient in Eq. (3), we follow Hao et al. [23] and set m to 20. We apply AD-DROP only in the first layer for the datasets of SST-2, MNLI, QNLI, QQP, and STS-B since finetuning on these datasets is stable and less likely to cause overfitting. For the rest datasets (CoLA, MRPC, and RTE), we apply AD-DROP in all layers. We provide the detailed hyperparameter settings on each dataset in Appendix B.2. Our code is submitted in the supplementary material and will be released later.

Table 1: Overall results of finetuned models on the GLUE benchmark. The symbol † denotes results directly taken from the original papers. Best average results are shown in bold.

Methods	SST-2	MNLI	QNLI	QQP	CoLA	STS-B	MRPC	RTE	Average
				Development					
$\text{BERT}_{\text{base}}$	92.3	84.6	91.5	91.3	60.3	89.9	85.1	70.8	83.23
+*SCAL*† [17]	92.8	84.1	90.9	91.4	61.7	-	-	69.7	-
+*SuperT*† [42]	93.4	84.5	91.3	91.3	58.8	89.8	87.5	72.5	83.64
+*R-Drop*† [18]	93.0	85.5	92.0	91.4	62.6	89.6	87.3	71.1	84.06
+AD-DROP	93.9	85.1	92.3	91.8	64.6	90.4	88.5	75.1	**85.21**
$\text{RoBERTa}_{\text{base}}$	95.3	87.6	92.9	91.9	64.8	90.9	90.7	79.4	86.69
+*R-Drop* [18]	95.2	87.8	93.2	91.7	64.7	91.2	90.5	80.5	86.85
+*HiddenCut*† [15]	95.8	88.2	93.7	92.0	66.2	91.3	92.0	83.4	87.83
+AD-DROP	95.8	88.0	93.5	92.0	66.8	91.4	92.2	84.1	**87.98**
				Test					
$\text{BERT}_{\text{base}}$	93.6	84.7	90.4	89.3	52.8	85.6	81.4	68.4	80.78
+AD-DROP	94.3	85.2	91.6	89.4	53.3	86.6	84.1	68.7	**81.65**
$\text{RoBERTa}_{\text{base}}$	94.8	87.5	92.8	89.6	58.3	88.7	86.3	75.1	84.14
+AD-DROP	95.9	87.6	93.4	89.5	58.5	89.3	87.9	76.0	**84.76**

3.3 Overall Results

We report the overall results of the finetuned models in Table 1. We first compare AD-DROP with existing regularization methods on the development sets, including the original finetuning (FT), SCAL [17], SuperT [42], R-Drop [18], and HiddenCut [15]). We observe that AD-DROP surpasses the baselines on most of the datasets. Specifically, AD-DROP yields an average improvement of 1.98 and 1.29 points over FT on $\text{BERT}_{\text{base}}$ and $\text{RoBERTa}_{\text{base}}$, respectively, and another average improvement of 1.15 and 1.13 points over R-Drop. We then discuss the performance of AD-DROP on the test sets. Results in Table 1 show that AD-DROP achieves consistent improvement over FT, boosting the average scores of $\text{BERT}_{\text{base}}$ and $\text{RoBERTa}_{\text{base}}$ by 0.87 and 0.62, respectively. Besides, compared with large datasets, AD-DROP achieves more gains on small datasets, which are more likely to cause overfitting, illustrating that AD-DROP is more suitable for small data scenarios.

3.4 Ablation Study

We conduct ablation experiments on four small datasets to investigate the impact of different components. Due to the limited number of submissions imposed by the GLUE server for evaluation, the results here are reported on the development sets.

Table 2: Results of ablation studies, where *r/w* means "replace with" and *w/o* means "without".

Methods	CoLA	STS-B	MRPC	RTE
$\text{BERT}_{\text{base}}$	60.3	89.9	85.1	70.8
+AD-DROP (GA)	**64.6**	90.4	**88.5**	**75.1**
r/w IGA	63.8	**90.7**	**88.5**	74.4
r/w AA	63.6	90.0	88.0	74.7
r/w RD	62.1	90.2	87.8	74.7
r/w gold labels	63.2	-	88.0	74.4
w/o cross-tuning	62.1	90.4	87.3	71.5
$\text{RoBERTa}_{\text{base}}$	64.8	90.9	90.7	79.4
+AD-DROP (GA)	66.8	91.4	**92.2**	**84.1**
r/w IGA	**68.1**	**91.6**	91.4	82.7
r/w AA	66.3	91.5	91.2	82.3
r/w RD	66.5	91.5	**92.2**	82.0
r/w gold labels	66.4	-	91.2	82.0
w/o cross-tuning	67.3	91.3	90.4	80.5

Attribution methods AD-DROP can be implemented with different attribution methods to generate the mask matrix in Eq. (1), such as integrated gradient attribution (IGA) introduced Eq. (3), attention weights for attribution (AA), and randomly generating the discard region (RD) in Eq. (6). We replace the gradient attribution (GA) in Eq. (5)-(6) with these methods. From Table 2, we can make three observations. First, AD-DROP with gradient-based attribution methods (GA and IGA) surpasses that with the other methods (AA or RD) on most datasets, illustrating that gradient-based methods are better at finding features that are likely to cause

overfitting. Second, IGA outperforms GA in some cases. Although IGA provides better theoretical justification than GA for attribution, it requires prohibitively more computational cost than GA (see Section 4.4 for efficiency analysis), making GA a more desirable choice for AD-DROP. Third, AD-DROP improves the original $\text{BERT}_{\text{base}}$ and $\text{RoBERTa}_{\text{base}}$ with any of the masking strategies, demonstrating the robustness of AD-DROP to overfitting when finetuning these models.

Pseudo labels *vs* gold labels In Section 2.3, we discuss the motivation of using pseudo labels for attribution in AD-DROP. To verify the reasonability, we conduct an experiment with gold labels for attribution. As the results show in Table 2, using gold labels for attribution deteriorates the performance, illustrating that AD-DROP with pseudo labels for attribution is preferable.

Cross-tuning To verify the effectiveness of the cross-tuning strategy, we ablate it and apply only AD-DROP in all training epochs. As shown in Table 2, removing cross-tuning causes noticeable performance degradation on most of the datasets. This can be explained by the intuition that AD-DROP without cross-tuning tends to discard high attribution positions excessively and make the model difficult to converge normally. To vividly demonstrate the effect of AD-DROP with or without cross-tuning, we visualize the distributions of the performance on the RTE[3] development set when enumerating the parameters p and q in the range of [0.1, 0.9]. The results are plotted in Figure 4, where each blue/orange point denotes the accuracy with a pair of p and q values. We observe from the figure that AD-DROP without cross-tuning cannot be trained properly under some parameter settings. However, it works well for most parameter settings when cross-tuning is applied, demonstrating that cross-tuning is vital for improving training stability.

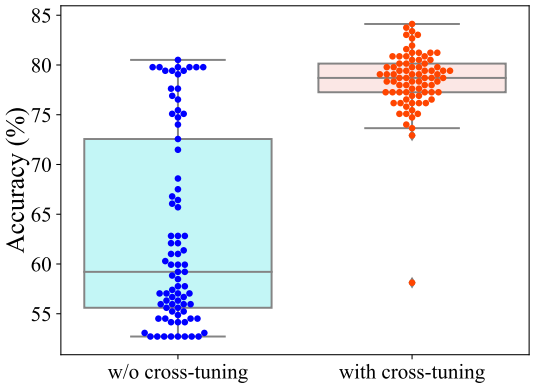


Figure 4: Results of AD-DROP with and without cross-tuning when enumerating p and q in [0.1, 0.9]. RoBERTa is chosen as the base model.

4 Analysis

In this section, we further conduct several experiments for more thorough analysis.

4.1 Repeated Experiments

To reduce the influence of randomness, we conduct repeated experiments on the four small datasets (i.e., CoLA, STS-B, MRPC and RTE). We repeat the training of each model with five random seeds and report the average score and standard deviation. From Table 3, we observe that AD-DROP outperforms the original FT on all the datasets. In addition, AD-DROP results in lower standard deviations on most of the datasets, showing that AD-DROP is more robust in finetuning PrLMs than the original approach.

Table 3: Results of repeated experiments on development sets. Each result is the average of five runs with standard deviation.

Methods	CoLA	STS-B	MRPC	RTE
$\text{BERT}_{\text{base}}$	$61.8_{\pm 1.9}$	$89.4_{\pm 0.5}$	$85.2_{\pm 1.3}$	$71.2_{\pm 1.2}$
+AD-DROP	$\mathbf{63.4}_{\pm 0.4}$	$\mathbf{90.1}_{\pm 0.5}$	$\mathbf{87.4}_{\pm 0.9}$	$\mathbf{73.9}_{\pm 1.1}$
$\text{RoBERTa}_{\text{base}}$	$64.3_{\pm 0.9}$	$91.0_{\pm 0.2}$	$89.8_{\pm 0.8}$	$79.1_{\pm 1.7}$
+AD-DROP	$\mathbf{66.4}_{\pm 0.9}$	$\mathbf{91.2}_{\pm 0.1}$	$\mathbf{91.3}_{\pm 0.7}$	$\mathbf{82.5}_{\pm 0.9}$

4.2 Effect of Data Size

To study the impact of data size, we compare AD-DROP with the original FT on QNLI and QQP,[4] two relatively large datasets, and report their performance when the number of training samples changes.

[3]Results on the other datasets are provided in Appendix C.1.

[4]Results on QQP are shown in Appendix C.2.

RoBERTa is chosen as the base model. Figure 5 shows that AD-DROP outperforms FT consistently on QNLI. Moreover, AD-DROP improves the efficiency of data use as training AD-DROP with 60% training samples produces comparable performance to FT with full data.

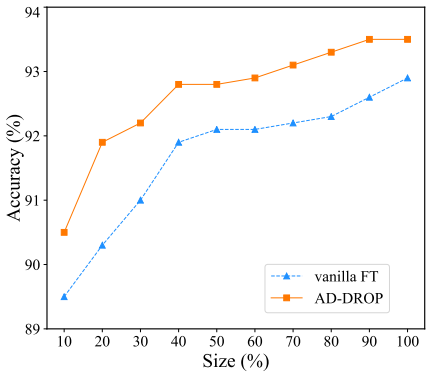
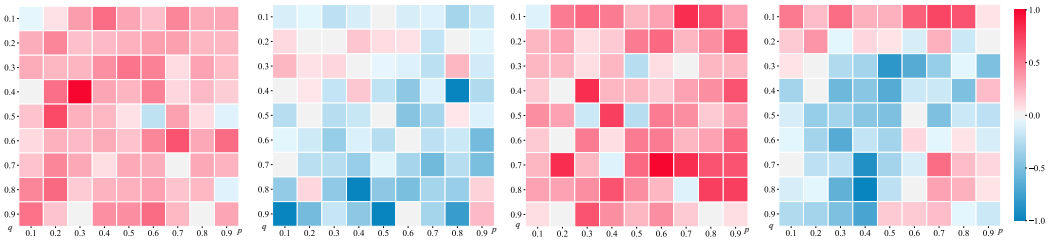


Figure 5: Results of AD-DROP and FT as the number of training samples changes.

4.3 Hyperparameter Sensitivity

AD-DROP involves two hyperparameters p and q to control the number of discarded attention positions. To investigate the sensitivity of AD-DROP to them, we show the results of different p and q combinations on CoLA and RTE in Figure 6, in which we apply `MaxAbsScaler`[5] to project the difference between the results of AD-DROP and FT into the interval of $[-1.0, 1.0]$. We observe that BERT with AD-DROP is not hyperparameter-sensitive as it outperforms the baseline under most settings. In contrast, RoBERTa with AD-DROP is more sensitive and requires a careful search for optimal hyperparameter settings. The possible reason is that RoBERTa is pretrained with more data and more effective tasks than BERT, making it less prone to overfitting than BERT.



(a) BERT on CoLA (b) RoBERTa on CoLA (c) BERT on RTE (d) RoBERTa on RTE

Figure 6: Results of sensitivity study on CoLA and RTE. Rows refer to p and columns refer to q. Blue blocks indicate the results of AD-DROP below the baseline (FT), and red blocks mean results of AD-DROP above the baseline. Darker colors mean greater gaps with the baseline.

4.4 Computational Efficiency

For computational efficiency analysis, we quantitatively study the computational cost of AD-DROP with different dropping strategies (GA, IGA, AA, and RD) relative to the original FT on CoLA, STS-B, MRPC, and RTE. BERT is chosen as the base model for this experiment. As shown in Table 4, although IGA achieves more favorable performance on one of the datasets, it requires higher computational cost than its counterparts, especially when applied in all the layers. In contrast, AD-DROP with GA is more competitive in both performance and computational cost.

Table 4: Results of performance and computational cost of AD-DROP with different masking strategies (GA, IGA, AA, and RD) relative to the original FT. The symbol ‡ means AD-DROP is only applied in the first layer. BERT is chosen as the base model.

Methods	CoLA		STS-B‡		MRPC		RTE	
	Mcc	Time	Pcc	Time	Acc	Time	Acc	Time
RD	+1.8	×1.4	+0.3	×1.0	+2.7	×1.3	+3.9	×1.4
AA	+3.3	×1.4	+0.1	×1.0	+2.9	×1.9	+3.9	×1.6
GA	+4.3	×3.6	+0.5	×1.3	+3.4	×4.1	+4.3	×4.5
IGA	+3.5	×99.6	+0.8	×10.2	+3.4	×110.1	+3.6	×125.7

[5]https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MaxAbsScaler.html

5 Related Work

Dropout Dropout is a widely used regularizer to mitigate overfitting when training deep neural networks. Vanilla dropout [19] randomly selects neurons with a predefined probability and sets their values to zeros during training. By doing so, the neurons cannot co-adapt and the trained networks can lead to better generalization. In recent years, many variants of dropout have emerged. One body of research aims to adopt different strategies to drop units in neural networks. For example, DropConnect [43] randomly selects connections between neurons to discard. DropBlock [21] defines a structured dropout that randomly drops the units in a specific contiguous region of a feature map. AutoDropout [22] aims to improve the dropout pattern of DropBlock by introducing an automatic method to design dropout structures. HiddenCut [15] drops contiguous spans within the hidden space, in which the attention weights are utilized to select the dropped spans strategically. Another body of research devotes to addressing the inconsistency between training and inference when dropout is applied. For instance, mixout [11] randomly replaces selected parameters with original pretrained weights rather than setting them to zeros. CHILD-TUNING [13] selects a child network and masks out the gradients of the non-child network during the backward step, only updating weights in the child network. R-Drop [18] performs dropout twice in the forward steps to produce two sub-models and then applies KL-divergence for their output distributions, forcing the two sub-models to be consistent with each other. However, most of these methods follow the random sampling strategy of dropout and pay little attention to the different importance of self-attention positions in PrLMs.

Attribution Numerous studies have been undertaken to interpret the behaviors of deep neural networks (DNNs). As a theory for interpretability, attribution aims to evaluate the impact of input features on predictions [24]. Generally, attribution methods can be divided into perturbation-based [44–46], gradient-related [25, 47, 26], and attention-based [48–50] methods. We focus on reviewing the gradient-related methods as they are more relevant to our work. Specifically, earlier works [25, 51, 52] try to explain model decisions via gradients since gradients indicate the direction and rate that changes the loss the fastest. However, Sundararajan et al. [26] point out that gradient attribution violates the sensitivity axiom in some cases that the gradients will be zero for the function in saturated areas, and propose integrated gradient as a theoretically more sound attribution method.

Other efforts have been devoted to revealing the behavior patterns of PrLMs. Kovaleva et al. [8] and Clark et al. [9] use attention weights for attribution to investigate what specific knowledge BERT [1] learns. Jain and Wallace [27] and Brunner et al. [53] investigate the identifiability of attention weights and conclude that attention weights are not a faithful explanation for model predictions. Hao et al. [23] apply integrated gradient [26] as a saliency measure for self-attention attribution in BERT, and use the attribution result to interpret information interactions inside Transformers. Similarly, Lu et al. [28] develop influence patterns based on integrated gradient to explain information flow in BERT. Unlike these works, we aim to examine the effect of dropout on self-attention through self-attention attribution and to reduce overfitting when finetuning PrLMs.

6 Conclusion

We proposed a novel dropout regularizer, AD-DROP, to mitigate overfitting when finetuning PrLMs on downstream tasks. Unlike previous dropout-based methods that generally adopt the random sampling strategy to discard units, AD-DROP draws inspiration from self-attention attribution which reveals that attention positions are not equally important in reducing overfitting and that dropping inappropriate positions may exacerbate the problem. Therefore, AD-DROP focuses on discarding high attribution attention positions to prevent the model from relying heavily on these positions to make predictions. Besides, we proposed a cross-tuning strategy that performs the original finetuning and our AD-DROP alternately to stabilize the finetuning process. Extensive experiments and analysis on the GLUE benchmark demonstrate the effectiveness of AD-DROP. Although originally proposed and evaluated based on self-attention attribution, AD-DROP can be readily extended to other neural network units as vanilla dropout, which deserves further research efforts.

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[2] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020.

[4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[5] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontañón, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: Transformers for longer sequences. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[6] Sebastian Jaszczur, Aakanksha Chowdhery, Afroz Mohiuddin, Lukasz Kaiser, Wojciech Gajewski, Henryk Michalewski, and Jonni Kanerva. Sparse is enough in scaling transformers. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 9895–9907, 2021.

[7] Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one? *Advances in Neural Information Processing Systems*, 32, 2019.

[8] Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. Revealing the dark secrets of bert. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4365–4374, 2019.

[9] Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D Manning. What does bert look at? an analysis of bert's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, 2019.

[10] Yihe Dong, Jean-Baptiste Cordonnier, and Andreas Loukas. Attention is not all you need: Pure attention loses rank doubly exponentially with depth. In *International Conference on Machine Learning*, pages 2793–2803. PMLR, 2021.

[11] Cheolhyoung Lee, Kyunghyun Cho, and Wanmo Kang. Mixout: Effective regularization to finetune large-scale pretrained language models. In *International Conference on Learning Representations*, 2019.

[12] Marius Mosbach, Maksym Andriushchenko, and Dietrich Klakow. On the stability of fine-tuning bert: Misconceptions, explanations, and strong baselines. In *International Conference on Learning Representations*, 2020.

[13] Runxin Xu, Fuli Luo, Zhiyuan Zhang, Chuanqi Tan, Baobao Chang, Songfang Huang, and Fei Huang. Raise a child in large language model: Towards effective and generalizable fine-tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9514–9528, 2021.

[14] Jacob Andreas. Good-enough compositional data augmentation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7556–7566, 2020.

[15] Jiaao Chen, Dinghan Shen, Weizhu Chen, and Diyi Yang. Hiddencut: Simple data augmentation for natural language understanding with better generalizability. 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 4380–4390, 2021.

[16] Xiaodong Liu, Hao Cheng, Pengcheng He, Weizhu Chen, Yu Wang, Hoifung Poon, and Jianfeng Gao. Adversarial training for large neural language models. *arXiv preprint arXiv:2004.08994*, 2020.

[17] Deshui Miao, Jiaqi Zhang, Wenbo Xie, Jian Song, Xin Li, Lijuan Jia, and Ning Guo. Simple contrastive representation adversarial learning for nlp tasks. *arXiv preprint arXiv:2111.13301*, 2021.

[18] Lijun Wu, Juntao Li, Yue Wang, Qi Meng, Tao Qin, Wei Chen, Min Zhang, Tie-Yan Liu, et al. R-drop: regularized dropout for neural networks. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

[19] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[20] Yarin Gal, Jiri Hron, and Alex Kendall. Concrete dropout. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 3584–3593, 2017.

[21] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Dropblock: A regularization method for convolutional networks. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

[22] Hieu Pham and Quoc Le. Autodropout: Learning dropout patterns to regularize deep networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 9351–9359, 2021.

[23] Yaru Hao, Li Dong, Furu Wei, and Ke Xu. Self-attention attribution: Interpreting information interactions inside transformer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 12963–12971, 2021.

[24] Yu Zhang, Peter Tiňo, Aleš Leonardis, and Ke Tang. A survey on neural network interpretability. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2021.

[25] David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert Müller. How to explain individual classification decisions. *The Journal of Machine Learning Research*, 11:1803–1831, 2010.

[26] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, pages 3319–3328. PMLR, 2017.

[27] Sarthak Jain and Byron C Wallace. 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, 2019.

[28] Kaiji Lu, Zifan Wang, Piotr Mardziel, and Anupam Datta. Influence patterns for explaining information flow in bert. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

[29] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

[30] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *In Workshop at International Conference on Learning Representations*. Citeseer, 2014.

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

[32] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 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, 2013.

[33] Adina Williams, Nikita Nangia, and Samuel R Bowman. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*, 2017.

[34] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

[35] Zihan Chen, Hongbo Zhang, Xiaoji Zhang, and Leqi Zhao. Quora question pairs. *URL https://www. kaggle. com/c/quora-question-pairs*, 2018.

[36] Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. Cola: The corpus of linguistic acceptability (with added annotations). 2019.

[37] Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*, 2017.

[38] Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. The fifth pascal recognizing textual entailment challenge. In *TAC*, 2009.

[39] Brian W Matthews. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure*, 405(2):442–451, 1975.

[40] Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. Pearson correlation coefficient. In *Noise reduction in speech processing*, pages 1–4. Springer, 2009.

[41] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/2020.emnlp-demos.6`.

[42] Chen Liang, Simiao Zuo, Minshuo Chen, Haoming Jiang, Xiaodong Liu, Pengcheng He, Tuo Zhao, and Weizhu Chen. Super tickets in pre-trained language models: From model compression to improving generalization. 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 6524–6538, 2021.

[43] Li Wan, Matthew Zeiler, Sixin Zhang, Yann LeCun, and Rob Fergus. Regularization of neural networks using dropconnect. In *Proceedings of the 30th International Conference on International Conference on Machine Learning-Volume 28*, pages III–1058, 2013.

[44] Jiwei Li, Will Monroe, and Dan Jurafsky. Understanding neural networks through representation erasure. *arXiv preprint arXiv:1612.08220*, 2016.

[45] Shi Feng, Eric Wallace, Alvin Grissom II, Mohit Iyyer, Pedro Rodriguez, and Jordan Boyd-Graber. Pathologies of neural models make interpretations difficult. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3719–3728, 2018.

[46] Karl Schulz, Leon Sixt, Federico Tombari, and Tim Landgraf. Restricting the flow: Information bottlenecks for attribution. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=S1xWh1rYwB`.

[47] Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, Klaus-Robert Müller, and Wojciech Samek. Layer-wise relevance propagation for neural networks with local renormalization layers. In *International Conference on Artificial Neural Networks*, pages 63–71. Springer, 2016.

[48] Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*, 2015.

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

[50] Sarah Wiegreffe and Yuval Pinter. Attention is not not explanation. 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 11–20, 2019.

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

[52] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

[53] Gino Brunner, Yang Liu, Damian Pascual, Oliver Richter, Massimiliano Ciaramita, and Roger Wattenhofer. On identifiability in transformers. In *8th International Conference on Learning Representations (ICLR 2020)(virtual)*. International Conference on Learning Representations, 2020.

[54] Han Shi, Jiahui Gao, Hang Xu, Xiaodan Liang, Zhenguo Li, Lingpeng Kong, Stephen MS Lee, and James Kwok. Revisiting over-smoothing in bert from the perspective of graph. In *International Conference on Learning Representations*, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] Our contributions and scope are accurately reflected in the abstract and introduction.
   (b) Did you describe the limitations of your work? [Yes] We discuss the potential limitations of our work in Appendix D.
   (c) Did you discuss any potential negative societal impacts of your work? [No] Our work is a general training method and will not bring any negative societal impact.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] We have read the guidelines and ensured our paper conforms to them.
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We have submitted our code and data in the supplemental material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] We describe our training details in Section 3.2 and Appendix B.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We conduct repeated experiments in Section 4.1.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] We describe the type of GPUs used in this work in Section 3.2.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We provide appropriate citations for the used data and tools.
   (b) Did you mention the license of the assets? [No] All used datasets are publicly available.
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Appendix: More Prior Experiments

Our observations in Figure 2(a) show that dropping low attribution positions makes the model fit the training data rapidly, while dropping high attribution positions reduces the fitting speed. To further probe the effect of dropping low or high attention positions, we finetune a RoBERTa on the training set and evaluate its performance on the development set by applying the two dropping strategies. The results on MRPC, SST-2, and QNLI are plotted in Figure 7. Similar phenomena can be observed that the model rapidly fits the data while dropping only a small proportion of low attribution positions. As the dropping rate increases, the accuracy remains stable until discarding too many low attribution positions. When dropping high attribution positions, we observe an opposite trend that the performance deteriorates sharply. These results further confirm the observations in Section 2.2 that attention positions should not be treated equally important in dropout.

Note that we only drop positions in the first layer of RoBERTa for the above experiments to exclude the impact of different layers. We also conduct experiments in the other layers on SST-2, and the overall results are shown in Figure 8. We note that similar results are obtained in the first few layers, while the trend becomes less obvious as the number of layers increases. It could be caused by the over-smoothing [54] issue that the representations of all tokens are similar in the last few layers.

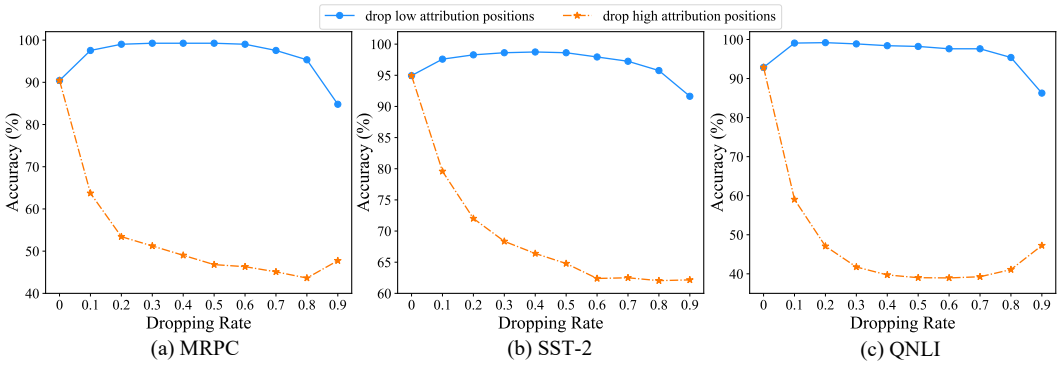


Figure 7: Performance of finetuned RoBERTa on development sets, where two dropping strategies (i.e., drop low/high attribution positions) are applied. Gold labels are used for the attribution.

B Appendix: Experimental Details

B.1 Details of Datasets

Table 5: Statistics of the GLUE datasets.

Dataset	Train	Dev	Test
SST-2	67349	872	1821
MNLI	392702	9815	9796
QNLI	104743	5463	5463
QQP	363846	40430	390965
CoLA	8551	1043	1063
STS-B	5749	1500	1378
MRPC	3668	408	1725
RTE	2490	277	3000

The details of the GLUE datasets are introduced as follows. (1) Stanford Sentiment Treebank (**SST-2**) [32] is a sentence sentiment prediction task. (2) Multi-Genre Natural Language Inference (**MNLI**) [33] is a pairwise sentence classification task that aims to predict whether the relationship between two sentences is entailment, contradiction, or neutral. (3) Question Natural Language Inference (**QNLI**) [34] is a binary pairwise sentence classification task that aims to predict whether a question-answer pair contains the correct answer. (4) Quora Question Pairs (**QQP**) [35] is a binary pairwise sentence classification task that aims to predict whether two questions are equivalent semantically. (5) The Corpus of Linguistic Acceptability (**CoLA**) [36] aims to indicate whether a single English sentence conforms to linguistics. (6) The goal of Semantic Textual Similarity Benchmark (**STS-B**) [37] is to predict how two given sentences are semantically similar. (7) Microsoft Research Paraphrase Corpus (**MRPC**) [31] aims to predict if two sentences are semantically equivalent. (8) Recognizing Textual Entailment (**RTE**) [38] is similar to MNLI but has binary labels. The statistics of these datasets are shown in Table 5.

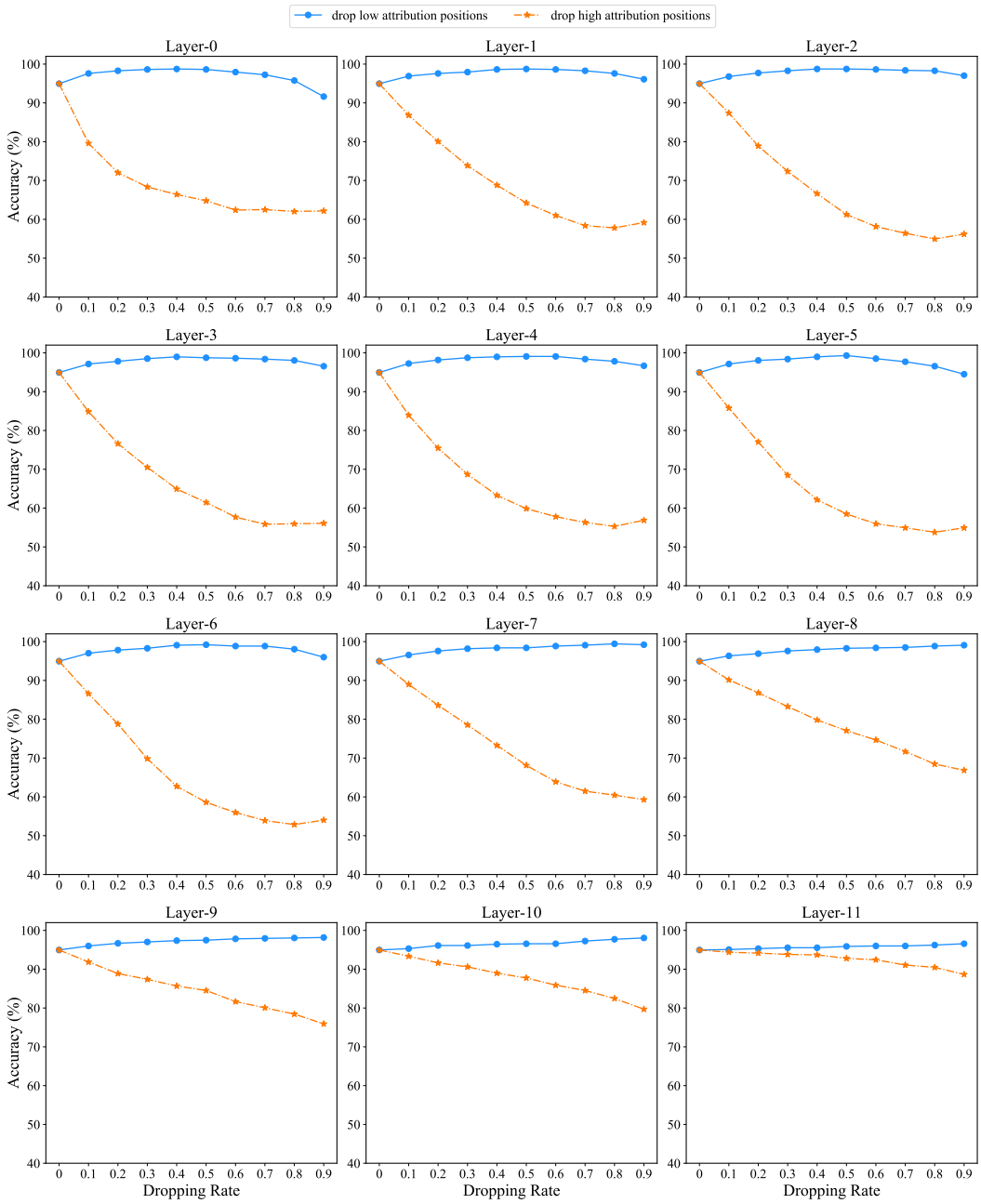


Figure 8: Results of dropping self-attention positions in different layers of RoBERTa on SST-2.

B.2 Hyperparameter Settings

Table 6 presents the final hyperparameter settings of AD-DROP for BERT/RoBERTa. The setting with only one value means the shared parameter for BERT and RoBERTa.

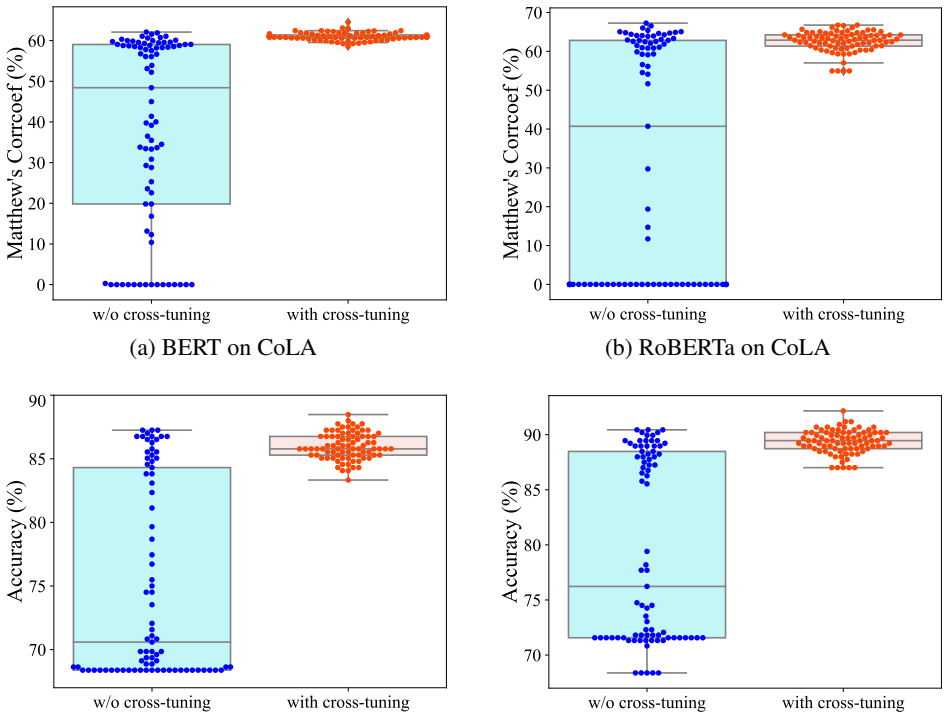
Table 6: Hyperparameter settings of AD-DROP for BERT and RoBERTa.

Dataset	Learning rate	Batch size	Length	p	q
SST-2	1e-5	16/64	120	0.6/0.3	0.8/0.7
MNLI	1e-5	16/32	128	0.5/0.4	0.9/0.2
QNLI	1e-5	16	128	0.8	0.8/0.4
QQP	1e-5	16	120	0.2/0.7	0.7/0.9
CoLA	1e-5/2e-5	16	47	0.3/0.8	0.4/0.3
STS-B	1e-5/2e-5	16	100	0.9/0.1	0.7/0.5
MRPC	1e-5/2e-5	16	100	0.5/0.8	0.8/0.3
RTE	1e-5	16	128	0.6/0.7	0.7/0.1

C Appendix: More Experimental Results

C.1 Ablation of Cross-Tuning

We further report the results of removing cross-tuning in AD-DROP when enumerating p and q in the range of [0.1, 0.9] on the CoLA and MRPC datasets. We observe consistent performance degradation in Figure 9 after removing the cross-tuning strategy from AD-DROP.



(a) BERT on CoLA (b) RoBERTa on CoLA

(c) BERT on MRPC (d) RoBERTa on MRPC

Figure 9: Results of AD-DROP with and without cross-tuning when enumerating p and q in the range of [0.1, 0.9] on the CoLA and MRPC datasets.

C.2 Effect of Data Size on QQP

Figure 10 shows a comparison between AD-DROP and vanilla FT as the size of training examples changes. We observe that AD-DROP performs consistently better than vanilla FT with different sizes of training data.

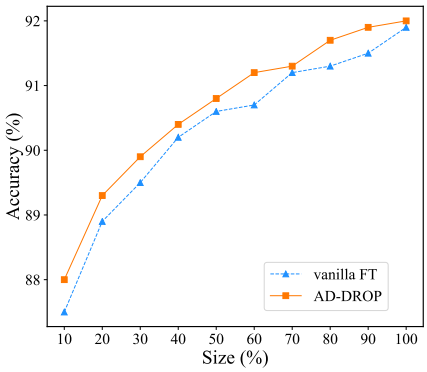


Figure 10: Results of comparison between AD-DROP and vanilla FT as the size of training data changes on QQP.

D Appendix: Limitations

We discuss potential limitations of AD-DROP as follows. First, as reported in Section 4.4, training with AD-DROP requires more computational cost than the original fine-tuning approach especially when integral gradients is applied for attribution in all the attention heads. Therefore, we propose to use gradient for attribution in AD-Drop as it achieves competitive performance with acceptable computational cost. Second, AD-DROP introduces additional hyperparameters (p and q) and requires more search efforts for the best hyperparameters.