AD-DROP: Attribution-Driven Dropout for Robust Language Model Fine-Tuning

Tao Yang, Jinghao Deng, Xiaojun Quan, Qifan Wang, Shaoliang Nie
1School of Computer Science and Engineering, Sun Yat-sen University 2Meta AI
1{yangt225,dengjh27}@mail2.sysu.edu.cn, quanxj3@mail.sysu.edu.cn
2{wqfcru, snie}@fb.com

Abstract

Fine-tuning large pre-trained 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 some 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 fine-tuning and AD-DROP to avoid dropping high-attribution positions excessively. Extensive experiments on various benchmarks show that AD-DROP yields consistent improvements over baselines. Analysis further confirms that AD-DROP serves as a strategic regularizer to prevent overfitting during fine-tuning.

1 Introduction

Pre-training large language models (PrLMs) on massive unlabeled corpora and fine-tuning them on downstream tasks has become a new paradigm [1–3]. Their success can be partly attributed to the self-attention mechanism [4], yet these self-attention networks are often redundant [5, 6] and tend to cause overfitting when fine-tuned on downstream tasks due to the mismatch between their overparameterization and the 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 [18–20] 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 fine-tuning PrLMs.

Attribution [23] 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 [23–28]. Our prior experiment of self-attention attribution (Section 2.2) reveals that attention positions are not equally important in preventing overfitting, and 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.

*Corresponding author.
Motivated by the above, we propose Attribution-Driven Dropout (AD-DROP) to better fine-tune 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. 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 discards attention positions, AD-DROP focuses on dropping high-attribution positions in candidate discard regions.

Unlike vanilla dropout which randomly discards attention positions, AD-DROP focuses on dropping high-attribution positions in candidate discard regions. Unlike vanilla dropout which randomly discards attention positions, AD-DROP focuses on dropping high-attribution positions in candidate discard regions.

Motivated by the above, we propose Attribution-Driven Dropout (AD-DROP) to better fine-tune 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 fine-tuning 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 sample a set of positions with high attribution scores and generate a mask matrix for each attention map. Finally, the mask 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. Hence, we further propose a cross-tuning strategy that performs fine-tuning and AD-DROP alternately to improve the training stability.

To verify the effectiveness of AD-DROP, we conduct extensive experiments with different PrLMs (i.e., BERT [1], RoBERTa [2], ELECTRA [29], and OPUS-MT [30]) on various datasets (i.e., GLUE [31], CoNLL-2003 [32], WMT 2016 EN-RO and TR-EN [33], HANS [34], and PAWS-X [35]). Experimental results show that the models tuned with AD-DROP obtain remarkable improvements over that tuned with the original fine-tuning approach. For example, on the GLUE benchmark, 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 simple random sampling. Moreover, they also demonstrate that the cross-tuning strategy plays a crucial role in improving training stability.

To sum up, this work reveals that self-attention positions are not equally important for dropout when fine-tuning 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 $X \in \mathbb{R}^{n \times d}$ 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 $X$ into three matrices $Q_h, K_h$ and $V_h$ via linear projections as query, key, and value respectively for the $h$-th head. Then, the attention output of this head is calculated as:

$$\text{Attention}(Q_h, K_h, V_h) = A_h V_h = \text{softmax} \left( \frac{Q_h K_h^T}{\sqrt{d_k}} + M_h \right) V_h,$$

where $\sqrt{d_k}$ is a scaling factor. $M_h$ is the mask matrix to apply dropout in self-attention, and elements in $M_h$ will be $-\infty$ if the corresponding positions in attention maps are masked and 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 \cite{35,36} directly produces an attribution matrix \( \mathbf{B}_h \) by computing the following partial derivative:

\[
\mathbf{B}_h = \frac{\partial F_c(\mathbf{A})}{\partial \mathbf{A}_h},
\]

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. \cite{26} propose integrated gradient, which is employed by Hao et al. \cite{23} as a saliency measure for self-attention attribution. Specifically, Hao et al. \cite{23} compute the attribution matrix \( \mathbf{B}_h \) as:

\[
\mathbf{B}_h = \frac{\mathbf{A}_h \odot \sum_{k=1}^{m} \frac{\partial F_c(k/n \mathbf{A})}{\partial \mathbf{A}_h}}{m},
\]

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 A Prior Attribution Experiment

To better motivate our work, we first conduct a prior experiment on MRPC \cite{37} to investigate how different positions in self-attention maps affect fine-tuning performance based on attribution results. RoBERTa-base \cite{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 obtain an attribution matrix \( \mathbf{B}_h \) for the self-attention positions in the first layer by gradient attribution with Eq. (2) and sort each row of the matrix. Finally, we sample a set of self-attention positions with high or low attribution scores in each row to generate a mask 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 trained. 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 the 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.

### 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 may affect their generalization.

---

\(^2\)We provide more results and discussions in Appendix B.
Formally, given a training set $D = \{(x_i, y_i)\}_{i=1}^N$ of $N$ samples, where $x_i$ is the $i$-th sample and $y_i$ is its label, the goal of AD-DROP is to fine-tune a PrLM $F(\cdot)$ of $L$ layers on $D$. Same as the vanilla dropout [19], AD-DROP is only applied in the training phase.

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 the pseudo label. The reason we adopt pseudo labels rather than gold labels 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 $\tilde{c}$:

$$\tilde{c} = \arg \max_c (P_F(c|x_i)),$$

where $P_F(c|x_i)$ is the probability of class $c$ for $x_i$. After the forward computation, we also obtain a set of attention maps $A = [A_1, A_2, \cdots, A_H]$ for each layer according to Eq. (1).

Second, we compute the attribution matrices $B = [B_1, B_2, \cdots, B_H]$ for $H$ heads according to Eq. (2). Specifically, the attribution matrix $B_h$ for the $h$-th head is computed as:

$$B_h = \frac{\partial F(\tilde{c})}{\partial A_h},$$

where $F(\tilde{c})$ is the logit output of pseudo label $\tilde{c}$ before softmax.

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

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

where $b_{i,j}$ and $\tilde{b}_{i,j}$ are elements of $B_h$ and $\tilde{B}_h$, respectively, $\text{int}(\cdot)$ 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 $M_h$ as:

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

where $u_{i,j} \sim \text{Bernoulli}(1-q)$ is an element of matrix $U_h \in \mathbb{R}^{n \times n}$, and $q$ is the dropout rate.

Finally, $M_h$ 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.

---

3The negative loss will be used for both regression and token-level tasks, as introduced in Appendix A.
**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 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 attribution with gold labels will be investigated 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 fine-tuning and AD-DROP alternatively. Specifically, it performs the original fine-tuning 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 fine-tuning 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 $W$  
**Output:** updated parameters $\tilde{W}$

1: Initialize $F$ with $W$, $epoch = 1$
2: while not converged do
3:   Calculate the prediction $P_F(y_i|x_i)$ and loss via forward computation.
4:   if $epoch \% 2 == 1$ then
5:      Backpropagate the loss to update model parameters $W$.
6:   else
7:      Perform AD-DROP by Eq. (4)-(7) to obtain mask matrices $M = [M_1, M_2, \cdots, M_H]$.
8:      Calculate the new prediction $P_{\tilde{F}}(y_i|x_i)$ and new loss by feeding $M$ into Eq. (1).
9:      Backpropagate the new loss to update model parameters $\tilde{W}$.
10:  $epoch = epoch + 1$
11: return $\tilde{W} = W$

### 3 Experiments

#### 3.1 Datasets

We conduct our main experiments on eight tasks of the GLUE benchmark [31], including SST-2 [38], MNLI [39], QNLI [40], QQP [41], CoLA [42], STS-B [43], MRPC [37], and RTE [44]. The evaluation metrics are Matthew’s Corrcoef (Mcc) [45] for CoLA, Pearson Corrcoef (Pcc) [46] for STS-B, and Accuracy (Acc) for the others. To demonstrate that AD-DROP applies to token-level tasks as well, we conduct experiments on Named Entity Recognition (CoNLL-2003 [32]) and Machine Translation (WMT 2016 [33]) datasets, the results of which are shown in Appendix A.2. Besides, we also evaluate AD-DROP on two out-of-distribution (OOD) datasets, including HANS [34] and PAWS-X [35]. The details of these datasets are introduced in Appendix C.1.

#### 3.2 Implementation Details

We implement our AD-DROP in Pytorch with the Transformers package [47]. We train the selected PrLMs 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 choose the number of training epochs on GLUE. 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 the fine-tuning on these datasets is stable and less likely to cause overfitting. For the rest datasets, we apply AD-DROP in all layers. We provide the detailed hyperparameter settings on each dataset in Appendix C.2. Our code is available at [https://github.com/TaoYang225/AD-DROP](https://github.com/TaoYang225/AD-DROP).
### 3.3 Overall Results

We report the overall results of the fine-tuned models in Table 1. We first compare AD-DROP with existing regularization methods on the development sets, including the original fine-tuning, SCAL [17], SuperT [48], 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 on BERT base and RoBERTa base, respectively. We then discuss the performance of AD-DROP on the test sets. Results in Table 1 show that AD-DROP achieves consistent improvement, boosting the average scores of BERT base and RoBERTa 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.

**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 of the 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 4.7 for efficiency analysis), making GA a more desirable choice for AD-DROP. Third, AD-DROP improves the original
BERT\textsubscript{base} and RoBERT\textsubscript{a} base with any of the masking strategies, demonstrating the robustness of AD-DROP to overfitting when fine-tuning 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 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.

### 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 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 on the development sets. From Table 3 we observe that AD-DROP outperforms the original fine-tuning 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 fine-tuning PrLMs than the original approach.

#### 4.2 Effect of Data Size

To study the impact of data size, we compare AD-DROP with the original fine-tuning (FT) approach on QNLI and QQP two relatively large datasets, and report their performance when the number of training samples changes. RoBERT\textsubscript{a} is chosen as the base model. Figure 5 shows that AD-DROP

---

4 Results on the other datasets are shown in Appendix D.1.
5 Results on QQP are shown in Appendix D.2.
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.

### 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 4 in which we apply MaxAbsScaler\(^6\) 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 pre-trained with more data and more effective tasks than BERT, making it less prone to overfitting than BERT.

![Figure 5: Results of AD-DROP and FT as the number of training samples changes.](image)

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

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

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

### 4.4 Larger Model Size

To verify the scalability of AD-DROP for a larger model size, we evaluate AD-DROP with RoBERTa\(_{\text{large}}\) on the RTE and MRPC datasets. Table 4 shows the average scores and standard deviations of five random seeds. There are two main observations. First, AD-DROP achieves consistent improvements over the larger RoBERTa model, illustrating that AD-DROP is scalable to large models. Second, compared with RoBERTa\(_{\text{base}}\) on RTE in Table 3, the larger model significantly reduces the deviation (from 1.7 to 0.86), suggesting that a larger model size indeed helps to improve the stability. AD-DROP further improves the performance and reduces the deviation.

![Table 4: Testing AD-DROP on a larger model.](image)

**Table 4:** Testing AD-DROP on a larger model.

<table>
<thead>
<tr>
<th>Methods</th>
<th>MRPC</th>
<th>RTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>RoBERTa(_{\text{large}})</td>
<td>90.83 ± 0.75</td>
<td>85.99 ± 0.86</td>
</tr>
<tr>
<td>+AD-DROP</td>
<td><strong>91.62 ± 0.53</strong></td>
<td><strong>88.01 ± 0.48</strong></td>
</tr>
</tbody>
</table>

### 4.5 Few-shot Scenario

In this subsection, we test the performance of AD-DROP under few-shot scenarios. Specifically, we carry out 16-, 64-, and 256-shot experiments on SST-2 and CoLA with RoBERTa\(_{\text{base}}\) as the base model and the baseline. We report the average scores and standard deviations of five random seeds in Table 5. We observe that RoBERTa with AD-DROP consistently outperforms the original fine-tuning approach. Besides, AD-DROP tends to bring more benefits when fewer samples are available.

4.6 Out-of-Distribution Generalization

To further demonstrate AD-DROP is beneficial to reducing overfitting, we test AD-DROP with RoBERTa\textsubscript{base} on two out-of-distribution (OOD) datasets, i.e., HANS and PAWS-X. For HANS, we use the checkpoints trained on MNLI and test their performance on the validation set (the test set is not supplied). For PAWS-X, we use the checkpoints trained on QQP and examine its performance on the test set. The evaluation metric is accuracy. From Table 6, we can see that RoBERTa with AD-DROP achieves better generalization, where AD-DROP boosts the performance by 0.66 on HANS and 3.35 on PAWS-X, illustrating that the model trained with AD-DROP generalizes better to OOD data.

4.7 Computational Efficiency

To analyze the computational efficiency, we quantitatively study the computational cost of AD-DROP with different dropping strategies (GA, IGA, AA, and RD) relative to the original fine-tuning on CoLA, STS-B, MRPC, and RTE. BERT is chosen as the base model for this experiment. As shown in Table 7, although IGA achieves more favorable performance on one of the datasets, it requires higher computational costs 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.

5 Related Work

**Dropout** Dropout is a widely used regularizer to mitigate overfitting when training deep neural networks. Vanilla dropout \cite{Srivastava2014} 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 \cite{Wen2015} randomly selects connections between neurons to discard. DropBlock \cite{Zagoruyko2017} defines a structured dropout that randomly drops the units in a specific contiguous region of a feature map. AutoDropout \cite{Zagoruyko2017} aims to improve the dropout pattern of DropBlock by introducing an automatic method to design dropout structures. HiddenCut \cite{Li2019} 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 pre-trained 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 [50,52], gradient-related [25,53,26], and attention-based [54,56] methods. We focus on reviewing the gradient-related methods as they are more relevant to our work. Specifically, earlier works [25,57,58] 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. [59] 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 fine-tuning PrLMs.

6 Conclusion

We propose a novel dropout regularizer, AD-DROP, to mitigate overfitting when fine-tuning 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 propose a cross-tuning strategy that performs the original fine-tuning and our AD-DROP alternately to stabilize the fine-tuning 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 potentially extended to other neural network units as vanilla dropout, which deserves further research efforts.

Acknowledgments

We appreciate the anonymous reviewers for their valuable comments. We thank Feifan Yang, Zihong Liang, and Hong Ding for their early discussions. This work was supported by the National Natural Science Foundation of China (No. 62176270), the Program for Guangdong Introducing Innovative and Entrepreneurial Teams (No. 2017ZT07X355), and the Foundation of Key Laboratory of Machine Intelligence and Advanced Computing of the Ministry of Education.

References


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 E.
   (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 C.
   (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]