R-AT: Regularized Adversarial Training for Natural Language Understanding

Shiwen Ni, Jiawen Li, and Hung-Yu Kao

Intelligent Knowledge Management Lab
Department of Computer Science and Information Engineering
National Cheng Kung University
Tainan, Taiwan

(P78083033, P78073012)@gs.ncku.edu.tw; hykao@mail.ncku.edu.tw

Abstract

Currently, adversarial training has become a popular and powerful regularization method in the natural language domain. In this paper, we propose a simple regularization technique upon adversarial training and dropout, called R-AT, which regularizes adversarial training by “twice adversarial dropout”. Specifically, the proposed method R-AT minimizes bidirectional KL-divergence between output distributions of the two sub-models sampled via dropout for each adversarial sample generated by adding gradient-based perturbation to clean examples. In this process, the constrained model predicts the output distribution of the adversarial sample, which is different from the original sample. This phenomenon will affect the model’s inference performance and stability.

1 Introduction

With the development of computer hardware, deep learning (LeCun et al., 2015) has achieved surprising performance in many artificial intelligence domains, such as, image recognition, speech recognition, natural language understanding, etc. However, “overfitting” is one of the biggest concerns in a deep learning community (Ng et al., 1997; Caruana et al., 2000; Belkin et al., 2018; Werpachowski et al., 2019). At present, in the actual training of deep neural networks, various regularization techniques (Srivastava et al., 2014; Erhan et al., 2009; Miyato et al., 2017) are indispensable to prevent overfitting and improve the generalization ability of deep neural networks.

Among them, dropout is currently one of the most widely used regularization methods, which randomly drops some neural units and all their input and output connections during the training process. The most significant advantage of dropout is that it can easily generate exponential parameter-sharing sub-models during the training process. Gao et al. (2021) and Liang et al. (2021) put data to pass through the model with dropout twice to generate different representations or output distributions, and then use varying representations for contrastive learning or regularization of distinct output distributions, which enhances the model’s effect to a certain extent. Furthermore, adversarial training is used to minimize the maximal risk for label-preserving input perturbations, which has been demonstrated to be a powerful regularization method for improving model generalization. Some of the latest adversarial training methods (Zhu et al., 2020; Ni et al., 2021) have accomplished impressive performance, particularly in natural language processing tasks. However, when adversarial training is executed in combination with dropout, the model inputting the same adversarial example will generate a relatively different output distribution. This phenomenon will affect the model’s inference performance and stability.

Inspired by SimCSE (Gao et al., 2021) and R-Drop (Liang et al., 2021), we proposed a simple regularization technique upon adversarial training and dropout, called R-AT, which regularizes adversarial training by “twice adversarial dropout”. Specifically, the proposed method R-AT minimizes bidirectional KL-divergence between output distributions of the two sub-models sampled via dropout for each adversarial sample generated by adding gradient-based perturbation to clean examples. In this process, the constrained model predicts the output distribution of the adversarial sample, which is different from the original sample. This phenomenon will affect the model’s inference performance and stability.

*Corresponding author

1https://github.com/IKMLab/R-AT
distribution of adversarial examples, acquires information about the distribution of non-target classes, and enhances generalization in inference. It is also worth noting that compared with standard adversarial training, most previous works on improving adversarial training have added multiple forward and back propagations, whereas our R-AT only increased the forward propagation once, without increasing the number of back propagations. The main contributions of this paper can be summarized as follows:

- We propose a simple but effective regularization strategy based on adversarial training and dropout, namely R-AT, whose output probability distributions of different sub-models generated by dropout are forced to be consistent with each other under the same adversarial sample input.

- We theoretically prove that R-AT has potential gradient regularization during the training process, demonstrating that the model tends to be optimized to a flatter minimum risk to improve generalization. In addition, R-AT can alleviate inconsistency of dropout in training and testing phases.

- Extensive experiments on 13 public natural language understanding datasets and four widely used baseline models demonstrate that our proposed R-AT achieves solid performances and outperforms other powerful regularization methods.

2 Related Work

It is well known that overfitting has always been a significant issue for deep learning models. Empirically, people find that the best-fitting model is typically a large model with appropriate regularization (Goodfellow et al., 2016). To mitigate overfitting and improve the generalization of deep learning models, numerous regularization techniques have been proposed, including weight decay (Kang et al., 2016; Krizhevsky et al., 2012; Krogh and Hertz, 1992; Wen et al., 2016), dropout (Wan et al., 2013; Hinton et al., 2012; Ba and Frey, 2013; Srivastava et al., 2014), normalization (Ba et al., 2016; Ioffe and Szegedy, 2015; Salimans and Kingma, 2016; Huang et al., 2018; Wu and He, 2018), adding noise (Hochreiter and Schmidhuber, 1995; Moradi et al., 2020; Poole et al., 2014), layer-wise pretraining and initialization (Erhan et al., 2009; He et al., 2015), label-smoothing (Müller et al., 2019; Zhang et al., 2021; Li et al., 2020), adversarial training (Goodfellow et al., 2015; Miyato et al., 2017; Zhu et al., 2020; Ni et al., 2021, 2022), and so on. Among them, dropout and its variants are very widely used due to their effectiveness and excellent compatibility with other regularization methods. The most significant advantage of dropout is that it can easily generate exponential parameter-sharing sub-models throughout the training process. Gao et al. (2021) regard dropout as a data enhancement method, which permits data to pass through the model with dropout twice to generate distinct embedding vectors. Liang et al. (2021) further regularize the neural networks on dropout success, which is done at the model output level. In this way, they encourage any two sub-models sampled from the dropout to generate consistent model predictions for the same input sample.

On the other hand, adversarial training (Goodfellow et al., 2015) was first proposed in the field of image recognition to defend against adversarial attacks. To be specific, adversarial training is the process of training a model that minimizes the maximal adversarial risk for label-preserving input perturbations. For the field of natural language processing, adversarial training (Miyato et al., 2017) has recently been proven to be a powerful regularization method to improve model generalization. Numerous studies have recently employed multiple iterations to calculate better perturbations to further enhance adversarial training (Athalye et al., 2018; Shafahi et al., 2019; Zhu et al., 2020; Ni et al., 2021). However, this requires the model to perform forward-backpropagation multiple times in one parameter update, linearly increasing computational cost. As known, the main calculations of neural networks are concentrated in the backpropagation.

The focus of this work is to enhance standard adversarial (Miyato et al., 2017) training without increasing the number of back propagations. We employed the “twice dropout” method to regularize adversarial training. Specifically, for each adversarial sample generated by adding gradient-based perturbation to clean samples, the proposed method R-AT minimizes bidirectional KL-divergence between output distributions of two sub-models sampled by dropout. In this process, the constrained model predicts the distribution of adversarial samples, acquires information about the distributions of non-target classes, and enhances generalization.
3 The Proposed Method

In this section, we first briefly describe the standard fine-tuning procedure for transformer-based model on natural language understanding tasks. We then introduce our regularized adversarial training (R-AT) method that perturb the word embedding matrix of neural networks to generate adversarial examples and further regularize the model’s prediction distribution for adversarial samples. Finally, we introduce in detail the training process of the regularized adversarial training algorithm. Figure 1 depicts the overall framework of R-AT.

3.1 Preliminaries

In this paper, the dataset is defined as $D = \{(x_i, y_i)\}_{i=1,2,...,N}$, where $N$ denotes the number of the training samples and $(x_i, y_i)$ is the labeled sample data pair. Taking the BERT model as an example, the standard text classification task is to feed a token sequence data $x_i = [\text{[CLS]}, e_1, e_2, ..., e_E]$, to the model, and the model outputs a sequence of contextualized token representations, the formula is as follows:

$$h_{\text{[CLS]}}^L, h_1^L, ..., h_E^L = \text{BERT}([\text{CLS}], e_1, ..., e_E) \tag{1}$$

where $L$ denotes the number of model layers.

The standard practice for fine-tuning BERT is to add a softmax classifier on top of the model’s sentence-level representations, such as the final hidden state $h_{\text{[CLS]}}$ of the [CLS] token in BERT:

$$P(y_c|x) = \text{softmax}(W \cdot h_{\text{[CLS]}}) \tag{2}$$

A basic learning goal of a deep learning model is to minimize the cross-entropy loss function, as follows:

$$L_{st} = -\frac{1}{N} \sum_{i=1}^{N} \sum_{c=1}^{C} y_{ci} \log(p(y_{ci}|x_i)) \tag{3}$$

where $N$ is batch size, $c \in \mathbb{R}^{d_e}$, and $C$ is the number of classes. $L_{st}$ is the loss of a mini-Batch under standard training.

3.2 R-AT Regularization

Adversarial training minimizes the maximal adversarial risk associated with label-preserving input perturbations and has been demonstrated to improve neural network generalization. This paper is based on the fast gradient method (FGM) proposed by (Miyato et al., 2017), which is a classic powerful adversarial training method in which the
Algorithm 1 Regularized Adversarial Training

Input: Training samples \( X \), perturbation coefficient \( \epsilon \), Loss weight coefficient \( \alpha \), Learning rate \( \tau \).

1: for epoch = 1 \ldots N_{epochs} do
2:     for \((x, y) \in X\) do
3:         Calculate the standard training gradient
4:             \( g_{st} \leftarrow \nabla_x L(\theta, \mu, x, y) \)
5:         Calculate the perturbation value
6:             \( \delta_{adv} = \epsilon \cdot \frac{\| g_{st} \|_2}{\| \nabla_x L(\theta, \mu, x, y) \|_2} \)
7:     end for
8:     Get two output distributions
9:         \( P^\mu_1(y|x + \delta_{adv}); P^\mu_2(y|x + \delta_{adv}) \)
10:    Calculate adversarial and KL gradient
11:        \( g^s_{adv} \leftarrow 1/2 \nabla_x \left[ -\log P^\mu_1(y|x + \delta_{adv}) \right] \)
12:        \( g^s_{KL} \leftarrow \nabla_x \left[ D^b_{KL}(P^\mu_1(y|x + \delta_{adv}) || P^\mu_2(y|x + \delta_{adv})) \right] \)
13:    Update parameter:
14:        \( \theta \leftarrow \theta - \tau (g_{st} + g_{adv} + \alpha \cdot g_{KL}) \)
15: end for
16: end for
17: Output: \( \theta \)

The perturbation \( \delta_{adv} \) is calculated using one backpropagation. The objective of standard adversarial training with dropout is to explore optimal parameters that minimize the maximum risk of adversarial attacks. The Min-Max formula is as follows:

\[
\min_{\theta} \mathbb{E}_{(x,y) \sim D} \max_{\delta_{adv} \in S} L(\theta, \mu, x + \delta_{adv}, y) \]  

(4)

where \( D \) is the data distribution, \( \theta \) is the model parameter, \( \mu \) is the mask matrix of dropout, \( y \) is the label, and \( L \) is loss function. The mask matrix \( \mu \) specifies the unit to be reserved. \( \delta_{adv} \) is the perturbation under maximizing internal risk. \( S \) is the perturbation constraint space. The perturbation calculation formula is as follows:

\[
\delta_{adv} = \epsilon \cdot \frac{\nabla_x L(\theta, \mu, x, y)}{\| \nabla_x L(\theta, \mu, x, y) \|_2} \]  

(5)

where \( \epsilon \) is the perturbation coefficient, \( \nabla_x L(\theta, \mu, x, y) \) is the model’s gradient to the input \( x \).

Notably, in the natural language processing model, adversarial training perturbs the embedding vector rather than directly perturbing the original text. In this work, we perturb the word embedding matrix of neural networks to generate adversarial examples. Specifically, after each forward-backward pass with clean examples, we calculate the gradient of the loss function in (2) and (5) with respect to the word embedding matrix to calculate the perturbation. The adversarial sample \( x + \delta_{adv} \) is then obtained by adding the perturbation to the clean input sample. At each training step, we feed the adversarial sample \( x_i + \delta_{adv} \). At each training step, we feed it to go through the forward pass of the network twice. Therefore, we can obtain two distributions of the perturbed model, denoted as \( P^\mu_1(y_i|x_i + \delta_{adv}) \) and \( P^\mu_2(y_i|x_i + \delta_{adv}) \). As illustrated in Figure 1, since the dropout operator randomly drops neural units in a model, the two forward passes are indeed based on two submodels of different structures. When the dropout technique is enabled, the model produces two different output distributions \( P^\mu_1(y_i|x_i + \delta_{adv}) \) and \( P^\mu_2(y_i|x_i + \delta_{adv}) \) for the same adversarial sample \( x_i + \delta_{adv} \). For these two output distributions, cross-entropy is used to calculate two corresponding adversarial losses:

\[
\mathcal{L}^i_{adv1} = -\log P^\mu_1(y_i|x_i + \delta_{adv}) \]  

\[
\mathcal{L}^i_{adv2} = -\log P^\mu_2(y_i|x_i + \delta_{adv}) \]  

(6)

Empirically, we hope that a deep learning model can maintain stable and consistent predictions for the same adversarial sample. Therefore, during the training process, the proposed method R-AT tries to regularize on the model predictions by minimizing the bidirectional KL-divergence between these two output distributions of the same adversarial sample as follows:

\[
\mathcal{L}^i_{KL} = |D^b_{KL}(P^\mu_1(y_i|x_i + \delta_{adv}) \parallel P^\mu_2(y_i|x_i + \delta_{adv})|/2 \]  

(7)

According to formula (3), the loss of the clean input sample \( x_i \) under standard training is calculated as follows:

\[
\mathcal{L}^i_{st} = -\log P^\mu(y_i|x_i) \]  

(8)

Under the R-AT technique, the final total loss for the input sample \( x_i \) is calculated as follows:

\[
\mathcal{L}_{\text{total}} = \mathcal{L}_{st} + \frac{1}{2}(\mathcal{L}_{adv1} + \mathcal{L}_{adv2}) + \alpha \mathcal{L}_{KL} \]  

(9)

where \( \alpha \) is the coefficient weight to control \( \mathcal{L}^i_{KL} \).

The training goal is to minimize the formula loss function \( \mathcal{L}_{\text{total}}^i \). Under this method, we not only learn the knowledge in clean and adversarial samples but also further regularize the model’s prediction distribution for adversarial samples. Therefore, the model trained using R-AT technique is more...
The regularized adversarial training algorithm process is shown in Algorithm 1. As illustrated above, for each training stage, lines 3-5 indicate forward and backward propagation to calculate the gradient of standard training, and then lines 5-6 calculate the perturbation according to the standard gradient, yielding adversarial sample of $x + \delta_{adv}$. Lines 7-8 indicate that adversarial sample input model with dropout twice produces two output distributions $P_1^{adv}(y|x + \delta_{adv})$ and $P_2^{adv}(y|x + \delta_{adv})$. Lines 9-11 indicate that the final backpropagation is performed to calculate adversarial and KL-loss gradients.

### 4 Experiments

#### 4.1 Datasets

We provide comprehensive evaluation on R-AT through extensive experiments on 13 natural language understanding datasets: GLUE benchmark (Wang et al., 2019), IMDB (Maas et al., 2011), AGNEWS (Zhang et al., 2015), TNEWS (Xu et al., 2020) and IFLYTEK (Xu et al., 2020). Additional details are provided in the appendix A.

#### 4.2 Experiment Settings

The GLUE benchmark is used to evaluate the pre-trained models (e.g., BERT, RoBERTa, ALBERT) with R-AT. We compare the R-AT with a state-of-the-art adversarial training method (FreeLB (Zhu et al., 2020)). IMDB, AGNEWS, TNEWS and IFLYTEK are used to evaluate the commonly used non-pre-trained models (e.g., BiLSTM, TextCNN, TextHAN) with R-AT.

For the pre-trained models BERT-base (110M),

<table>
<thead>
<tr>
<th>Model</th>
<th>CoLA</th>
<th>SST-2</th>
<th>MRPC</th>
<th>STS-B</th>
<th>QQP</th>
<th>MNLI</th>
<th>QNLI</th>
<th>RTE</th>
<th>WNLI</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT-base</td>
<td>60.1</td>
<td>93.1</td>
<td>89.7</td>
<td>85.3</td>
<td>86.6</td>
<td>88.5</td>
<td>84.7</td>
<td>81.2</td>
<td>89.7</td>
</tr>
<tr>
<td>BERT + R-AT</td>
<td>61.8</td>
<td>93.7</td>
<td>90.5</td>
<td>86.1</td>
<td>88.8</td>
<td>88.6</td>
<td>89.3</td>
<td>91.2</td>
<td>85.1</td>
</tr>
<tr>
<td>RoBERTa-base</td>
<td>59.1</td>
<td>94.8</td>
<td>92.0</td>
<td>89.0</td>
<td>90.0</td>
<td>89.9</td>
<td>88.8</td>
<td>91.6</td>
<td>87.5</td>
</tr>
<tr>
<td>RoBERTa + R-AT</td>
<td>60.8</td>
<td>95.1</td>
<td>92.5</td>
<td>89.4</td>
<td>90.5</td>
<td>90.3</td>
<td>89.7</td>
<td>91.6</td>
<td>89.2</td>
</tr>
<tr>
<td>ALBERT-base</td>
<td>52.6</td>
<td>89.1</td>
<td>88.0</td>
<td>83.6</td>
<td>88.9</td>
<td>88.3</td>
<td>87.0</td>
<td>90.3</td>
<td>81.8</td>
</tr>
<tr>
<td>ALBERT + R-AT</td>
<td>53.1</td>
<td>90.3</td>
<td>88.7</td>
<td>84.3</td>
<td>89.3</td>
<td>88.9</td>
<td>83.9</td>
<td>91.1</td>
<td>82.5</td>
</tr>
</tbody>
</table>

Table 1: Results on the dev sets of GLUE based on the BERT-base, RoBERTa-base and ALBERT-base models, from 5 runs with the same hyperparameter but different random seeds. § The BERT-base, RoBERTa-base and ALBERT-base models are trained with R-AT (Single-task finetuning). References: " (Devlin et al., 2019); b (Liu et al., 2019a); c (Liu et al., 2019b); d (Yang et al., 2019); e (Lan et al., 2020). BERT-base, RoBERTa-base and ALBERT-base models are pre-trained with R-AT (Initialized from MNLI weights for CoLA, MRPC, RTE). All compared results are taken from (Zhu et al., 2020).

<table>
<thead>
<tr>
<th>Model</th>
<th>Score</th>
<th>CoLA</th>
<th>SST-2</th>
<th>MRPC</th>
<th>STS-B</th>
<th>QQP</th>
<th>MNLI</th>
<th>QNLI</th>
<th>RTE</th>
<th>WNLI</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT-base</td>
<td>78.3</td>
<td>52.1</td>
<td>93.5</td>
<td>88.9</td>
<td>84.8</td>
<td>87.1</td>
<td>85.8</td>
<td>71.2</td>
<td>89.2</td>
<td>84.6</td>
</tr>
<tr>
<td>BERT + FreeLB</td>
<td>79.4</td>
<td>54.3</td>
<td>93.6</td>
<td>88.1</td>
<td>83.5</td>
<td>87.7</td>
<td>86.7</td>
<td>72.7</td>
<td>89.6</td>
<td>85.7</td>
</tr>
<tr>
<td>BERT + R-AT</td>
<td>79.6</td>
<td>54.3</td>
<td>93.8</td>
<td>89.7</td>
<td>85.2</td>
<td>87.5</td>
<td>85.9</td>
<td>72.4</td>
<td>89.5</td>
<td>85.8</td>
</tr>
<tr>
<td>MT-DNN</td>
<td>87.6</td>
<td>68.4</td>
<td>96.5</td>
<td>92.7</td>
<td>70.0</td>
<td>91.1</td>
<td>90.7</td>
<td>77.3</td>
<td>89.9</td>
<td>87.9</td>
</tr>
<tr>
<td>XLNet-Large</td>
<td>67.8</td>
<td>96.8</td>
<td>93.0</td>
<td>90.9</td>
<td>97.1</td>
<td>91.6</td>
<td>91.1</td>
<td>74.2</td>
<td>90.3</td>
<td>90.2</td>
</tr>
<tr>
<td>RoBERTa-large</td>
<td>88.1</td>
<td>67.8</td>
<td>96.7</td>
<td>92.3</td>
<td>89.8</td>
<td>92.2</td>
<td>91.9</td>
<td>74.3</td>
<td>90.2</td>
<td>90.8</td>
</tr>
<tr>
<td>RoBERTalarge + FreeLB</td>
<td>88.4</td>
<td>68.0</td>
<td>96.8</td>
<td>93.1</td>
<td>90.8</td>
<td>92.3</td>
<td>92.1</td>
<td>74.8</td>
<td>90.3</td>
<td>91.1</td>
</tr>
<tr>
<td>RoBERTalarge + R-AT</td>
<td>88.6</td>
<td>69.0</td>
<td>96.7</td>
<td>92.6</td>
<td>90.1</td>
<td>92.1</td>
<td>91.8</td>
<td>75.0</td>
<td>90.5</td>
<td>95.3</td>
</tr>
</tbody>
</table>

Table 2: GLUE Test results, scored by the official evaluation server (https://gluebenchmark.com/leaderboard). § The SOTA method, which is ensemble of 7 RoBERTa-Large + FreeLB models for each task (Initialized from MNLI weights for STS-B, MRPC, RTE). § The BERT-base is trained with R-AT (Single-task finetuning); The RoBERTa-Large model is trained with R-AT (Initialized from MNLI weights for CoLA, MRPC, RTE). All compared results are taken from (Zhu et al., 2020). References: : (Devlin et al., 2019); b: (Liu et al., 2019a); c: (Liu et al., 2019b); d: (Yang et al., 2019); e: (Lan et al., 2020).
RoBERTa-base (110M), ALBERT-base (12M) and RoBERTa-large (355M), we use the huggingface’s PyTorch\(^2\) implementation. Moreover, we use single-task finetuning for all dev set and test set results. The hyperparameters of our models are provided in the appendix A. For the non-pre-trained models, we utilized three well-known neural network models as the baseline, namely BiLSTM, TextCNN, TextHAN (BiGRU+Attention). To further explore the role of each part of R-AT we performed an ablation study. We compared baseline + regularized adversarial training (R-AT) with baseline, baseline + dropout, baseline + standard adversarial training and baseline + R-Drop and baseline + AT (no KL-Loss) methods. To ensure the experiment’s fairness, the model structure and hyper-parameters of all baselines are kept consistent.

### 4.3 Experimental Results

#### Pre-trained models on GLUE

We chose BERT, RoBERTa and ALBERT as the baseline models. The experimental results of the dev set on the GLUE benchmark are shown in Table 1. From the experimental results, BERT-base, RoBERTa-base and ALBERT-base use R-AT in the fine-tuning stage to have a significant performance improvement. Moreover, we also submit the experimental results to the GLUE evaluation server to obtain the test set results. As shown in Table 2, the main experimental results have been published on the official leaderboard of GLUE. For the overall score of test sets, R-AT increased the performance of the BERT-base model from 78.3 to 79.6, and the RoBERTa-large (355M) model from 88.1 to 88.6. For adversarial training on natural language understanding tasks, RoBERTa\_large + FreeLB has a state-of-the-art score of 88.4. On GLUEbenchmark, our model RoBERTa\_large + R-AT (without ensemble) achieved a score of 88.6, surpassing the 88.4 score of RoBERTa\_large + FreeLB (ensemble).

<table>
<thead>
<tr>
<th>Models</th>
<th>IMDB</th>
<th>AGNEWS</th>
<th>TNEWS</th>
<th>IFLYTEK</th>
<th>AVG.</th>
</tr>
</thead>
<tbody>
<tr>
<td>BiLSTM (w/o Dropout)</td>
<td>89.21 ± 0.64</td>
<td>90.98 ± 0.11</td>
<td>52.01 ± 0.83</td>
<td>50.36 ± 0.62</td>
<td>70.64</td>
</tr>
<tr>
<td>BiLSTM + Dropout</td>
<td>89.60 ± 0.57</td>
<td>90.94 ± 0.90</td>
<td>51.79 ± 0.31</td>
<td>50.38 ± 0.83</td>
<td>70.68</td>
</tr>
<tr>
<td>BiLSTM + AT</td>
<td>90.49 ± 0.16</td>
<td>92.05 ± 0.32</td>
<td>52.21 ± 0.21</td>
<td>50.62 ± 0.15</td>
<td>71.31</td>
</tr>
<tr>
<td>BiLSTM + R-Drop</td>
<td>89.64 ± 0.35</td>
<td>92.04 ± 0.14</td>
<td>51.47 ± 0.46</td>
<td>51.64 ± 0.34</td>
<td>71.20</td>
</tr>
<tr>
<td>BiLSTM + R-AT (w/o L(_{KL}))</td>
<td>90.01 ± 0.32</td>
<td>92.72 ± 0.15</td>
<td>52.34 ± 0.27</td>
<td>51.31 ± 0.27</td>
<td>71.60</td>
</tr>
<tr>
<td>BiLSTM + R-AT</td>
<td><strong>90.54 ± 0.29</strong></td>
<td><strong>93.05 ± 0.03</strong></td>
<td><strong>53.80 ± 0.18</strong></td>
<td><strong>54.40 ± 0.37</strong></td>
<td><strong>72.95</strong></td>
</tr>
<tr>
<td>TextCNN (w/o Dropout)</td>
<td>88.61 ± 0.17</td>
<td>91.52 ± 0.38</td>
<td>54.65 ± 0.32</td>
<td>53.07 ± 0.77</td>
<td>71.96</td>
</tr>
<tr>
<td>TextCNN + Dropout</td>
<td>88.68 ± 0.23</td>
<td>91.56 ± 0.22</td>
<td>55.36 ± 0.10</td>
<td>53.01 ± 0.68</td>
<td>72.15</td>
</tr>
<tr>
<td>TextCNN + AT</td>
<td>89.31 ± 0.25</td>
<td>91.83 ± 0.69</td>
<td>55.48 ± 0.28</td>
<td>53.12 ± 0.14</td>
<td>72.44</td>
</tr>
<tr>
<td>TextCNN + R-Drop</td>
<td>88.74 ± 0.12</td>
<td>91.70 ± 0.10</td>
<td>55.78 ± 0.21</td>
<td>53.32 ± 0.24</td>
<td>72.39</td>
</tr>
<tr>
<td>TextCNN + R-AT (w/o L(_{KL}))</td>
<td>89.70 ± 0.39</td>
<td>92.24 ± 0.23</td>
<td>55.76 ± 0.29</td>
<td>53.56 ± 0.19</td>
<td>72.82</td>
</tr>
<tr>
<td>TextCNN + R-AT</td>
<td><strong>90.03 ± 0.18</strong></td>
<td><strong>92.41 ± 0.12</strong></td>
<td><strong>55.93 ± 0.13</strong></td>
<td><strong>54.19 ± 0.25</strong></td>
<td><strong>73.14</strong></td>
</tr>
<tr>
<td>TextHAN (w/o Dropout)</td>
<td>90.21 ± 0.31</td>
<td>92.33 ± 0.17</td>
<td>55.58 ± 0.24</td>
<td>54.49 ± 0.13</td>
<td>73.15</td>
</tr>
<tr>
<td>TextHAN + Dropout</td>
<td>90.19 ± 0.11</td>
<td>92.34 ± 0.15</td>
<td>55.52 ± 0.28</td>
<td>55.54 ± 0.14</td>
<td>73.40</td>
</tr>
<tr>
<td>TextHAN + AT</td>
<td>91.25 ± 0.32</td>
<td>92.66 ± 0.18</td>
<td>55.73 ± 0.17</td>
<td>55.45 ± 0.21</td>
<td>73.68</td>
</tr>
<tr>
<td>TextHAN + R-Drop</td>
<td>90.36 ± 0.14</td>
<td>92.33 ± 0.28</td>
<td>56.18 ± 0.22</td>
<td>54.52 ± 0.23</td>
<td>73.35</td>
</tr>
<tr>
<td>TextHAN + R-AT (w/o L(_{KL}))</td>
<td>91.12 ± 0.19</td>
<td>92.68 ± 0.23</td>
<td>56.09 ± 0.29</td>
<td>55.96 ± 0.27</td>
<td>73.96</td>
</tr>
<tr>
<td>TextHAN + R-AT</td>
<td><strong>91.71 ± 0.15</strong></td>
<td><strong>93.06 ± 0.10</strong></td>
<td><strong>56.27 ± 0.21</strong></td>
<td><strong>57.04 ± 0.19</strong></td>
<td><strong>74.52</strong></td>
</tr>
</tbody>
</table>

Table 3: Results of non-pre-trained models. We report their mean ± standard deviation of 5 runs.

\(^2\)https://github.com/huggingface/transformers

Non-pre-trained models. As indicated in Table 2, models with R-AT generally outperform baseline models and other regularization strategies across all datasets. This demonstrates that R-AT method is universal and can be applied to a variety of tasks and languages. Compared to baseline models, our R-AT base models achieve an accuracy improvement of between 0.69% and 4.04%. R-AT-based BiLSTM model achieved the most improvement, increasing accuracy by 4.04% in IFLYTEK dataset. For adversarial training on natural language understanding tasks, RoBERTa\_large + FreeLB has a state-of-the-art score of 88.4. On GLUEbenchmark, our model RoBERTa\_large + R-AT (without ensemble) achieved a score of 88.6, surpassing the 88.4 score of RoBERTa\_large + FreeLB (ensemble).
result in a significant performance improvement.

For ablation studies, we observe that the AT and R-Drop base models outperform the original model. In contrast, R-AT base models perform the best. This is because R-AT strategy combines the benefits of adversarial training and R-Drop, making models easy to learn a set of bilaterally friendly representations relevant to both two regularization strategies to facilitate model’s robustness and generalization via a mutual learning process. We also found that after removing KL-loss, the effect of R-AT generally decreased and the variance became larger, which indicated that KL-loss played an important regularizing role. In addition, we find the standard deviation of R-AT is small, which shows that the regularization of R-AT is stable.

5 Study and Analysis on R-AT

In this section, we conduct detailed research and analysis on R-AT from multiple aspects.

5.1 R-AT with Different Sized Training Sets

It is well known that the model is more likely to overfit on a smaller training dataset. We study the performance of R-AT on training datasets of various sizes and proceed to compare it against AT and R-Drop regular methods. This experiment is based on the IMDB dataset. As illustrated in Figure 2, our method R-AT not only achieves excellent performance on large training datasets but also significantly improves on small datasets. We can observe that other regularization methods such as AT and R-Drop do not perform well on small training datasets. On the BiLSTM, TextCNN, and TextHAN, and Bert-base baseline models, our R-AT has achieved the greatest improvement. This shows that R-AT has excellent regularization effects on datasets of different sizes and models with different structures.

5.2 R-AT with Two Different Dropout Rates

R-AT is propagated forward twice in each adversarial training step. Additionally, two sub-models are generated through dropout. In the previous experiment, the dropout rate was the same twice (e.g., 0.5 for BiLSTM, TextCNN, and TextHAN, and 0.1 for Bert-base). In this work, we also studied the influence of R-AT using two different dropout rates during training. We selected the two dropout rates from {0.1, 0.3, 0.5, 0.7, 0.9} with total 25 combinations. Figure 3 depicts the experimental outcomes. We discovered that the same dropout rates are the best choice (e.g., 0.3 and 0.3; 0.5 and 0.5). The experimental results indicate that when R-AT employs two different dropout rates, it can also maintain a good stable effect, but the dropout rate should not be too large (such as 0.9), implying that randomness is too high and the optimization difficulty is increased.

5.3 K-times R-AT: An Adversarial Sample Passes the Model K Times

Given that R-AT is based on “twice dropout” to regularize adversarial training, it is interesting to examine the effect of “K-times dropout” based on R-AT. In addition to previous research, we in-
<table>
<thead>
<tr>
<th>Baselines</th>
<th>$K=2$</th>
<th>$K=3$</th>
<th>Gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>BiLSTM</td>
<td>90.54 ± 0.29</td>
<td>90.61 ± 0.27 + 0.07</td>
<td></td>
</tr>
<tr>
<td>TextCNN</td>
<td>90.03 ± 0.18</td>
<td>90.05 ± 0.19 + 0.02</td>
<td></td>
</tr>
<tr>
<td>TextHAN</td>
<td>91.71 ± 0.15</td>
<td>91.66 ± 0.23 - 0.05</td>
<td></td>
</tr>
<tr>
<td>BERT-base</td>
<td>89.58 ± 0.21</td>
<td>89.61 ± 0.23 + 0.03</td>
<td></td>
</tr>
</tbody>
</table>

Table 4: Comparison of 2-times and 3-times R-AT. We report their mean ± standard deviation of 5 runs.

increased the forward propagation from two to three times, implying that an adversarial example runs the model three times. As stated in Section 3, when $K = 2$, the total loss of a data $x_i$ is Formula (9). When $K = 3$, the total loss of $x_i$ is:

$$L_i^{tot} = L_i^0 + \frac{1}{3}(L_i^{adv_1} + L_i^{adv_2} + L_i^{adv_3}) + \beta L_i^{KL}$$

where

$$L_i^{KL} = D_{KL}(P_i^\mu \parallel P_2^\mu) + D_{KL}(P_i^\mu \parallel P_3^\mu)$$

(10)

The new weight coefficient $\beta$ of KL loss is 1/3 of weight coefficient $\alpha$.

The experimental results on IMDB are shown in Table 3. We can see that the improvement of 3-times R-AT compared to 2-times R-AT is not obvious. For example, when $K = 2$, the accuracy for IMDB test set is 90.54%; when $K = 3$, the accuracy for IMDB test set is 90.61%. The two results are similar. More times means more consumption. Therefore, we believe that the two times R-AT provides sufficient regularization to the model training.

6 Theoretical Analysis of R-AT

We provide a theoretical perspective to explain why R-AT can be used as a regularization method to improve model generalization. According to Section 3, the following constrained optimization formula can be used to express regularized adversarial training in detail:

$$\min_{\theta} E_{(x,y)\sim D} E_{\mu}[L(\theta, \mu_{st}, x, y) + \max_{\delta_{adv} \in S} L(\theta, \mu_{adv}^{(1)}, x + \delta_{adv}, y)]$$

$$+ \max_{\delta_{adv} \in S} L(\theta, \mu_{adv}^{(2)}, x + \delta_{adv}, y)]$$

s.t. $E_{(x,y)\sim D} E_{\mu}[D_{KL}(P^{(1)\mu}_{adv}(y|x + \delta_{adv})) = 0$ 

where $\mu_{st}, \mu_{adv}^{(1)}, \mu_{adv}^{(2)}$ are different random mask matrices generated by three forward propagations through dropout.

According to our optimization goal, we can analyze R-AT from two aspects. The first is the effect of generated two adversarial Loss. For formula (11), we Taylor expand functions $f(x + \delta_{adv}) = L(\theta, \mu_{adv}^{(1)}(x, y), \delta_{adv} > 0)$ and $f(x + \delta_{adv}) = L(\theta, \mu_{adv}^{(2)}(x, y), \delta_{adv} > 0)$ at points $x$:

$$\min_{\theta} E_{(x,y)\sim D} E_{\mu}[\max_{\delta_{adv} \in S} L(\theta, \mu_{adv}^{(1)}(x, y), \delta_{adv} > 0) + L(\theta, \mu_{adv}^{(2)}(x, y), \delta_{adv} > 0)]$$

$$+ \max_{\delta_{adv} \in S} L(\theta, \mu_{adv}^{(1)}(x, y), \delta_{adv} > 0) + L(\theta, \mu_{adv}^{(2)}(x, y), \delta_{adv} > 0)]$$

(13)

Although the probability of $\mu_{st}, \mu_{adv}^{(1)}, \mu_{adv}^{(2)}$ is different in the dropout sampling process, if we weaken the dropout difference, $\mu_{st}, \mu_{adv}^{(1)}, \mu_{adv}^{(2)}$ are all approximately represented as $\mu$. After substituting the values $\delta_{adv} = \epsilon \cdot \nabla_x L(\theta, \mu, x, y)$, we can approximate the antagonistic loss, the following formula (15) is obtained:

$$\min_{\theta} E_{(x,y)\sim D} E_{\mu}[\max_{\delta_{adv} \in S} L(\theta, \mu_{adv}^{(1)}(x, y), \delta_{adv} > 0) + L(\theta, \mu_{adv}^{(2)}(x, y), \delta_{adv} > 0)]$$

$$+ \max_{\delta_{adv} \in S} L(\theta, \mu_{adv}^{(1)}(x, y), \delta_{adv} > 0) + L(\theta, \mu_{adv}^{(2)}(x, y), \delta_{adv} > 0)]$$

(14)

$$\Rightarrow \min_{\theta} E_{(x,y)\sim D} E_{\mu}[2L(\theta, \mu, x, y) + \epsilon \cdot \nabla_x L(\theta, \mu, x, y)]$$

(15)

We can observe the final optimization function formula (16), which actually adds a "gradient penalty" $\epsilon \cdot \| \nabla_x L(\theta, \mu, x, y) \|_2$ to input x (embedding vector). Gradient penalty pushes the gradient of some parameters and inputs to approach zero, indicating that the model is likely to be optimized to the extreme point.

Following that, we analyzed the effect of constraint formula (12). As known, while utilizing dropout training and keeping the training and testing consistent, the prediction result $\hat{y}$ of model should be obtained by the following formula:

$$\hat{y} = \sum_{\mu} P(\mu) P(y|x, \mu)$$

(16)

Here, the probability distribution $P(y|x, \mu)$ of each sub-model is defined by the mask matrix $\mu$, and
$P(\mu)$ is the probability distribution of sampling during training. Because this summation formula (16) contains exponential terms, it is not calculable in the actual process. Empirically, in actual testing, we employ a single neural network without dropout.

During testing, if a unit is reserved with probability $p$ during training, the unit’s output weight is multiplied by $p$. Obviously, the result calculated by the above method is an approximate value (this value is accurate in a linear model), and the sub-models average of (16) is replaced by the weights average.

As depicted in Figure 4, we consider a neural network with four neural units (including a nonlinear activation function). Except for the output unit, the other three neural units have a 1/3 probability of being dropped. Regardless of small probability events, the model generates three sub-models, which drop $h_1$ unit, $h_2$ unit, and $h_3$ unit, respectively. As illustrated in Figure 4 (b1), (b2), and (b3), the average of sub-models is used to calculate model output $\hat{y}_m$:

$$\hat{y}_m = \sum_{\mu} P(\mu)P(y|x, \mu)$$

$$= \frac{1}{3} \times \sigma(w_2 \cdot h_2 + w_3 \cdot h_3)$$

$$+ \frac{1}{3} \times \sigma(w_1 \cdot h_1 + w_3 \cdot h_3)$$

$$+ \frac{1}{3} \times \sigma(w_1 \cdot h_1 + w_2 \cdot h_2)$$  (17)

As indicated in Figure 4 (a), weights average method is used to calculate model output $\hat{y}_w$:

$$\hat{y}_w = \sigma(\frac{2}{3} \cdot w_1 \cdot h_1 + \frac{2}{3} \cdot w_2 \cdot h_2 + \frac{2}{3} \cdot w_3 \cdot h_3)$$  (18)

We assume that under constraint formula 14, the output of each sub-model sampled by dropout is the same. Therefore, we obtain the constraint formula (19) as follows, and then formula (19) derives the following formula (20).

$$(w_2 \cdot h_2 + w_3 \cdot h_3) = (w_1 \cdot h_1 + w_3 \cdot h_3)$$

$$(w_1 \cdot h_1 + w_3 \cdot h_3) = (w_1 \cdot h_1 + w_2 \cdot h_2)$$  (19)

$$(w_1 \cdot h_1 + w_2 \cdot h_2) = (w_2 \cdot h_2 + w_3 \cdot h_3)$$

$$\Rightarrow w_1 \cdot h_1 = w_2 \cdot h_2 = w_3 \cdot h_3.$$  (20)

From formula (20), we use $w_1 \cdot h_1$ to replace $w_2 \cdot h_2$ and $w_3 \cdot h_3$ in formulas (17) and (18). The formula is as follows:

$$\hat{y}_m = \hat{y}_w = \sigma(2w_1 \cdot h_1).$$  (21)

According to equation (21), under the constraint of equation (12), the weights average can be approximately equivalent to the sub-model’s average. In summary, the above theoretical analysis reveals that R-AT has implicit gradient regularization during the training process and reduces inconsistency between training and testing of dropout models.

7 Conclusion

In this paper, we propose a simple yet powerful regularization method based on adversarial training and dropout, namely R-AT. Particularly, R-AT dynamically perturbs the input embedding vector to generate adversarial examples, and then an adversarial example is passed through two dropout sub-models to generate two probability distributions. Ultimately, R-AT minimizes bidirectional KL-divergence of two output distributions to regularize model predictions. We evaluate the efficiency of R-AT on 13 popular public datasets. The experimental results indicate that applying R-AT to neural network models with various structures (e.g., rnn-based, cnn-based, and transformer-based models) all significantly improves performance. Moreover, theoretical analysis reveals that R-AT has potential gradient regularization during the training process, implying that the model tends to be optimized to a flatter minimum risk to improve generalization. Furthermore, R-AT can reduce inconsistency between the training and testing of dropout models.

Limitations

The main limitation is that although the number of backpropagations for R-AT remains the same compared to standard adversarial training, the forward propagation requires two times, which makes the training time longer.
Acknowledgements

This work was funded in part by Qualcomm through a Taiwan University Research Collaboration Project NAT-487842 and in part by the Ministry of Science and Technology, Taiwan, under grant MOST 111-2221-E-006-001.

References


Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. ICLR.


Andrew 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.


Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In EMNLP.


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 EMNLP.


### A Datasets and Hyperparameters

In order to comprehensively evaluate R-AT, we used the well-known 13 natural language understanding datasets to conduct experiments. **GLUE Benchmark**\(^3\) (Wang et al., 2019) is centered on 9 English natural language understanding tasks, which cover a broad range of domains, data quantities, and difficulties. These tasks are evaluated on accuracy and are balanced across classes. The description of these datasets is shown in Table 5.

1) **Corpus of Linguistic Acceptability (CoLA)** (Warstadt et al., 2018). This task is to check whether the grammar of the sentence is correct.

2) **Stanford Sentiment Treebank (SST-2)** (Socher et al., 2013). The task is to predict the sentiment (positive/negative) of a given sentence.

3) **Microsoft Research Paraphrase Corpus (MRPC)** (Dolan and Brockett, 2005)). This task is to determine whether the sentences in the sentence pair are semantically equivalent.

4) **Semantic Textual Similarity Benchmark (STS-B)** (Agirre et al., 2007)). Each pair of sentences is manually annotated, with similarity scores ranging from 1 to 5; This is a regression task to predict these scores.

5) **Quora Question Pairs (QQP)** (Iyer et al., 2017)). The task is to determine whether a pair of questions are semantically equivalent.

6) **Multi-Genre NLI (MNLI)** (Williams et al., 2018)). Given a premise sentence and a hypothesis sentence, the task is to predict whether the premise entails the hypothesis, contradicts the hypothesis, or neither (two test set).

7) **Stanford Question Answering Dataset (QNLI)** (Rajpurkar et al., 2016)). The task is to determine whether the context sentence contains the answer to the question.

8) **Recognizing Textual Entailment (RTE)** (Dagan et al., 2006); (Bar Haim et al., 2006); (Giampiccolo et al., 2007); (Bentivogli et al., 2009)). A NLI task that integrates a series of annual textual content challenge datasets.

9) **Winograd NLI (WNLI)** (Levesque et al., 2011)). A reading comprehension task in which the system must read a sentence with pronouns and find the referent of the pronoun from the list.

10-13) The **IMDB** (Maas et al., 2011) dataset is a standard benchmark for sentiment analysis. The **AGnews** topic classification dataset is constructed by Zhang et al. (2015) from the original AG news.

\(^3\)https://gluebenchmark.com
sources. The **TNEWS** is a short news classification dataset in ChineseGLUE (Xu et al., 2020) benchmark. The **IFLYTEK** is a long text topic classification in ChineseGLUE (Xu et al., 2020) benchmark. The statistical introduction of IMDB, AGNEWS, TNEWS (chinese) and IFLYTEK (chinese) is shown in Table 6. For RoBERTa_{large} + R-AT in Table 2, its hyperparameters are provided in Table 7. We use the dev score to select the best model to predict the test set. The experimental hardware is a RTX 3090.

### B Different Perturbation Coefficient

In the previous experiment, the two perturbation coefficient are the same. The perturbation coefficient determines the magnitude of the perturbation value, and we study the impact of different perturbations. We choose the two dropout rates from \{1, 3, 5, 7, 9\} with 25 combinations. The experimental results are shown in Figure 5. We found that the same perturbation coefficient is the best choice (e.g., 5 and 5). The results indicate that R-AT can also maintain a good effect stably when it uses two different perturbation coefficients. The perturbation coefficient is better in the range of 3 ∼ 9. However, different datasets may have different optimal perturbation coefficient values.

![Figure 5: R-AT with two different perturbation coefficient combinations. The abscissa is the first dropout rate, and the ordinate is the second dropout rate.](image)

### C Comparison of different loss functions

R-AT forces the output probability distributions of different sub-models generated by dropout to be consistent with each other under the same adversarial sample input. Therefore, the loss function between the two probability distributions needs to be minimized during model optimization. For the choice of loss function, we compared the following common loss functions:

- CE: Cross entropy loss
- MAE: Mean absolute error loss
- MSE: Mean squared error loss
- KL: Kullback-Leibler divergence loss
- JS: Jensen-Shannon divergence loss
- Bi-KL: Bidirectional KL-divergence loss

We take BiLSTM as the baseline model, and use R-AT with the above 6 different loss functions to optimize the model. The experimental results are shown in Table 8. From the experimental results, Jensen-Shannon divergence loss and Bidirectional KL-divergence loss have the best results. Both of these Loss have symmetry and are suitable for reflecting the difference between the two distributions. In this paper, we use bidirectional KL-divergence loss for experiments. For the STS-B regression task, we use mean squared error loss instead of bidirectional KL-divergence loss.
<table>
<thead>
<tr>
<th>Corpus</th>
<th>Train</th>
<th>Test</th>
<th>Task Description</th>
<th>Language</th>
<th>Classes</th>
<th>Dataset Splitting</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoLA</td>
<td>8.5k</td>
<td>1k</td>
<td>acceptability</td>
<td>English</td>
<td>2</td>
<td>40,000 5,000 5,000</td>
</tr>
<tr>
<td>SST-2</td>
<td>67k</td>
<td>1.8k</td>
<td>sentiment</td>
<td>English</td>
<td>4</td>
<td>110,000 10,000 7,600</td>
</tr>
<tr>
<td>MRPC</td>
<td>3.7k</td>
<td>1.7k</td>
<td>paraphrase</td>
<td>English</td>
<td>2</td>
<td>153,360 10,000 10,000</td>
</tr>
<tr>
<td>STS-B</td>
<td>7k</td>
<td>1.4k</td>
<td>sentence similarity</td>
<td>English</td>
<td>15</td>
<td>12,133 2,599 2,600</td>
</tr>
<tr>
<td>QQP</td>
<td>364k</td>
<td>391k</td>
<td>paraphrase</td>
<td>English</td>
<td>2</td>
<td>24,926 2,599 2,600</td>
</tr>
</tbody>
</table>

Table 5: Task descriptions and statistics (Wang et al., 2019). All tasks are single sentence or sentence pair classification, except STS-B, which is a regression task. MNLI has three classes; all other classification tasks have two. Test sets shown in bold use labels that have never been made public in any form.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Task Description</th>
<th>Language</th>
<th>Classes</th>
<th>Dataset Splitting</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMDB</td>
<td>Movie reviews sentiment analysis</td>
<td>English</td>
<td>2</td>
<td>40,000 5,000 5,000</td>
</tr>
<tr>
<td>AGNEWS</td>
<td>AG’News topic classification</td>
<td>English</td>
<td>4</td>
<td>110,000 10,000 7,600</td>
</tr>
<tr>
<td>TNEWS</td>
<td>Short Text Classification for News</td>
<td>Chinese</td>
<td>15</td>
<td>153,360 10,000 10,000</td>
</tr>
<tr>
<td>IFLYTEK</td>
<td>Long Text topic classification</td>
<td>Chinese</td>
<td>119</td>
<td>12,133 2,599 2,600</td>
</tr>
</tbody>
</table>

Table 6: The statistical introduction of IMDB, AGNEWS, TNEWS, and IFLYTEK datasets.

<table>
<thead>
<tr>
<th>Hyper-parameters</th>
<th>CoLA</th>
<th>SSTR-2</th>
<th>MRPC</th>
<th>STS-B</th>
<th>QQP</th>
<th>MNLI</th>
<th>QNLI</th>
<th>RTE</th>
<th>WNLI</th>
</tr>
</thead>
<tbody>
<tr>
<td>α</td>
<td>0.5</td>
<td>0.5</td>
<td>0.8</td>
<td>0.5</td>
<td>0.8</td>
<td>0.8</td>
<td>0.8</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>ε</td>
<td>1E-1</td>
<td>1E-1</td>
<td>1.5E-1</td>
<td>1E-1</td>
<td>1E-1</td>
<td>1E-1</td>
<td>1E-1</td>
<td>1E-1</td>
<td>1E-1</td>
</tr>
<tr>
<td>τ</td>
<td>2E-5</td>
<td>1E-5</td>
<td>1E-5</td>
<td>1E-5</td>
<td>2E-5</td>
<td>2E-5</td>
<td>2E-5</td>
<td>2E-5</td>
<td>1E-5</td>
</tr>
<tr>
<td>Dropout</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>Batch size</td>
<td>24</td>
<td>32</td>
<td>32</td>
<td>32</td>
<td>32</td>
<td>32</td>
<td>32</td>
<td>24</td>
<td>16</td>
</tr>
</tbody>
</table>

Table 7: Hyper-parameters when fine-tuning our models on GLUE benchmark.

<table>
<thead>
<tr>
<th>Loss function</th>
<th>IMDB</th>
<th>AGNEWS</th>
<th>TNEWS</th>
<th>IFLYTEK</th>
<th>AVG.</th>
</tr>
</thead>
<tbody>
<tr>
<td>CE Loss</td>
<td>90.13 ± 0.45</td>
<td>93.01 ± 0.31</td>
<td>52.89 ± 0.16</td>
<td>53.57 ± 0.23</td>
<td>72.40</td>
</tr>
<tr>
<td>MAE Loss</td>
<td>90.03 ± 0.17</td>
<td>92.72 ± 0.23</td>
<td>52.87 ± 0.27</td>
<td>53.52 ± 0.31</td>
<td>72.29</td>
</tr>
<tr>
<td>MSE Loss</td>
<td>89.97 ± 0.32</td>
<td>92.29 ± 0.19</td>
<td>53.47 ± 0.33</td>
<td>53.48 ± 0.37</td>
<td>72.30</td>
</tr>
<tr>
<td>KL Loss</td>
<td>90.24 ± 0.33</td>
<td>92.19 ± 0.13</td>
<td>52.43 ± 0.22</td>
<td>54.12 ± 0.38</td>
<td>72.25</td>
</tr>
<tr>
<td>JS Loss</td>
<td>90.44 ± 0.31</td>
<td><strong>93.08 ± 0.11</strong></td>
<td>53.77 ± 0.17</td>
<td><strong>54.42 ± 0.35</strong></td>
<td>72.93</td>
</tr>
<tr>
<td>Bi-KL Loss</td>
<td><strong>90.54 ± 0.29</strong></td>
<td>93.05 ± 0.03</td>
<td><strong>53.80 ± 0.18</strong></td>
<td>54.40 ± 0.37</td>
<td><strong>72.95</strong></td>
</tr>
</tbody>
</table>

Table 8: Comparison of different loss functions. We report their mean ± standard deviation of 5 runs.