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 **R**egularized **A**dversarial **T**raining (**R-AT**) via dropout, which forces the output probability distributions of different sub-models generated by dropout to be consistent under the same adversarial samples. Specifically, we generate adversarial samples by perturbing the word embeddings. For each adversarial sample fed to the model, R-AT minimizes both the adversarial risk and the bidirectional KL-divergence between the adversarial output distributions of two sub-models sampled by dropout. Through extensive experiments on 13 public natural language understanding datasets, we found that R-AT has improvements for many models (e.g., rnn-based, cnn-based, and transformer-based models). For the GLUE benchmark, when R-AT is only applied to the fine-tuning stage, it is able to improve the overall test score of the BERT-base model from 78.3 to 79.6 and the RoBERTa-large model from 88.1 to 88.6. Theoretical analysis reveals that R-AT has potential gradient regularization during the training process. Furthermore, R-AT can reduce the inconsistency between training and testing of models with dropout[1].

1 Introduction

With the development of computing 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

[*]Corresponding author
[1]https://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 pre-training 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.

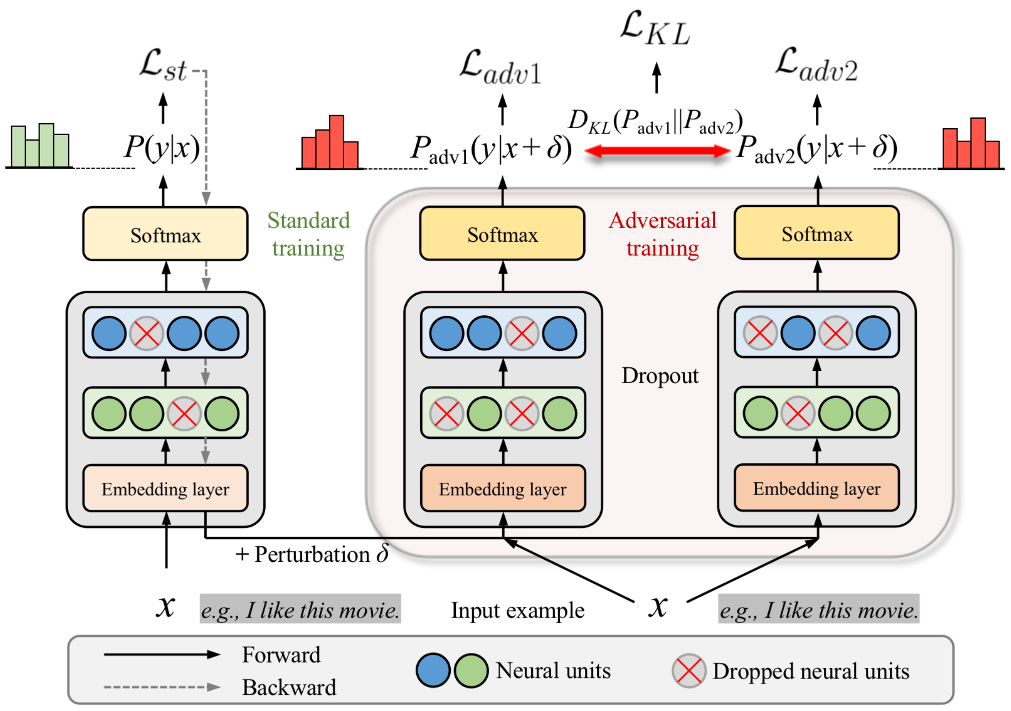


Figure 1: The overall framework of our proposed R-AT. The model first obtains gradient-based perturbation using forward-backward propagation calculation of the standard training and then obtains the adversarial sample by adding perturbation to the clean input sample. An adversarial sample will go through the model twice and generate two distributions P_{adv1} and P_{adv2}, due to two different dropout sub-models.

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 $x_i = [\texttt{CLS}], e_1, e_2, ..., e_E]$, to the model, and the model outputs a sequence of contextualized token representations , the formula is as follows:

$$h^L_{[\texttt{CLS}]}, h^L_1, ..., h^L_E = \text{BERT}([\texttt{CLS}], e_1, ..., e_E) \tag{1}$$

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 $h_{[\texttt{CLS}]}$ of the [CLS] token in BERT:

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

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

$$\mathcal{L}_{st} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y^i_c log(p(y^i_c|x_i)) \tag{3}$$

where N is batch size, $c \in \mathbb{R}^{d_c}$, and C is the number of classes. $\mathcal{L}_{st}$ 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 $\mathcal{X}$, perturbation coefficient ϵ, Loss weight coefficient α, Learning rate τ.

```
1: for $\text{epoch} = 1 \ldots N_{ep}$ do
2:     for $(x, y) \in \mathcal{X}$ do
3:         Calculate the standard training gradient
4:             $g^i_{st} \leftarrow \nabla_x \mathcal{L}(\theta, \mu, x, y)$
5:         Calculate the perturbation value
6:             $\delta_{adv} = \epsilon \cdot g^i_{st} / \parallel g^i_{st} \parallel_2$
7:         Get two output distributions
8:             $P^{\mu}_1(y|x + \delta_{adv}); P^{\mu}_2(y|x + \delta_{adv})$
9:         Calculate adversarial and $KL$ gradient
10:            $g^i_{adv} \leftarrow 1/2 \nabla_x [-log P^{\mu}_1(y|x + \delta_{adv})$
11:                $-log P^{\mu}_2(y|x + \delta_{adv})]$
12:            $g^i_{KL} \leftarrow \nabla_x [D^{bi}_{KL}(P^{\mu}_1(y|x + \delta_{adv})$
13:                $\parallel P^{\mu}_2(y|x + \delta_{adv}))]$
14:        Update parameter:
15:            $\theta \leftarrow \theta - \tau(g_{st} + g_{adv} + \alpha \cdot g_{KL})$
16:    end for
17: end for
18: Output: $\theta$
```

perturbation 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:

$$\min_{\theta} \mathbb{E}_{(x,y)\sim D}[\max_{\delta_{adv} \in S} \mathcal{L}(\theta, \mu, x + \delta_{adv}, y)] \quad (4)$$

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

$$\delta_{adv} = \epsilon \cdot \frac{\nabla_x \mathcal{L}(\theta, \mu, x, y)}{||\nabla_x \mathcal{L}(\theta, \mu, x, y)||_2} \quad (5)$$

where ϵ is the perturbation coefficient, $\nabla_x \mathcal{L}(\theta, \mu, x, y)$ 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 $x_i + \delta_{adv_i}$ 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 $P^{\mu}_1(y_i|x_i + \delta_{adv_i})$ and $P^{\mu}_2(y_i|x_i + \delta_{adv_i})$. 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 sub-models of different structures. When the dropout technique is enabled, the model produces two different output distributions $P^{\mu}_1(y_i|x_i + \delta_{adv_i})$ and $P^{\mu}_2(y_i|x_i + \delta_{adv_i})$ for the same adversarial sample $x_i + \delta_{adv_i}$. For these two output distributions, cross-entropy is used to calculate two corresponding adversarial losses:

$$\begin{aligned} \mathcal{L}^i_{adv_1} &= -log P^{\mu}_1(y_i|x_i + \delta_{adv_i}) \\ \mathcal{L}^i_{adv_2} &= -log P^{\mu}_2(y_i|x_i + \delta_{adv_i}) \end{aligned} \quad (6)$$

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:

$$\begin{aligned} \mathcal{L}^i_{KL} = [&D^{bi}_{KL}(P^{\mu}_1(y_i|x_i + \delta_{adv_i}) \\ &\parallel P^{\mu}_2(y_i|x_i + \delta_{adv_i})]/2 \end{aligned} \quad (7)$$

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

$$\mathcal{L}^i_{st} = -log P^{\mu}(y_i|x_i) \quad (8)$$

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

$$\mathcal{L}_{total} = \mathcal{L}_{st} + \frac{1}{2}(\mathcal{L}_{adv_1} + \mathcal{L}_{adv_2}) + \alpha \mathcal{L}_{KL} \quad (9)$$

where α is the coefficient weight to control $\mathcal{L}^i_{KL}$. The training goal is to minimize the formula loss function $\mathcal{L}^i_{total}$. 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

Model	CoLA (Mcc)	SST-2 (Acc)	MRPC (F1/Acc)	STS-B (Pe/Sp Corr)	QQP (F1/Acc)	MNLI (M/MM Acc)	QNLI (Acc)	RTE (Acc)	WNLI (Acc)	Avg.
BERT-base[a]	60.1	93.1	89.7/85.3	88.6/88.5	87.5/90.7	83.9/84.1	90.7	65.0	56.3	81.0
BERT + R-AT[§]	61.8	93.7	90.5/86.1	88.8/88.6	89.3/91.2	85.1/85.3	91.6	68.3	56.9	82.9
RoBERTa-base[b]	59.1	94.8	92.0/89.0	90.0/89.9	88.8/91.6	87.5/87.3	92.8	77.9	56.3	84.4
RoBERTa + R-AT[§]	60.8	95.1	92.5/89.4	90.5/90.3	89.7/91.6	89.2/89.3	93.3	78.2	56.5	85.1
ALBERT-base[c]	52.6	89.1	88.0/83.6	88.9/88.3	87.0/90.3	81.8/81.5	90.3	71.1	53.5	80.5
ALBERT + R-AT[§]	53.1	90.3	88.7/84.3	89.3/88.9	88.3/91.1	82.5/82.3	91.2	72.3	55.4	81.4

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: [a]: (Devlin et al., 2019); [b]: (Liu et al., 2019b); [c]: (Lan et al., 2020).

Model	Score	CoLA 8.5k	SST-2 67k	MRPC 3.7k	STS-B 7k	QQP 364k	MNLI 393k	QNLI 108k	RTE 2.5k	WNLI 634
BERT-base[a]	78.3	52.1	93.5	88.9/84.8	87.1/85.8	71.2/89.2	84.6/83.4	90.5	66.4	65.1
BERT$_{base}$ + FreeLB	79.4	54.5	93.6	88.1/83.5	87.7/86.7	72.7/89.6	85.7/84.6	91.8	70.1	65.1
BERT$_{base}$ + R-AT[§]	79.6	54.3	93.8	89.7/85.2	87.5/85.9	72.4/89.5	85.8/84.7	91.4	69.5	65.4
MT-DNN[b]	87.6	68.4	96.5	92.7/90.3	91.1/90.7	73.7/89.9	87.9/87.4	**96.0**	86.3	89.0
XLNet-large[c]	–	67.8	**96.8**	93.0/**90.7**	91.6/91.1	74.2/90.3	90.2/89.8	–	86.3	90.4
RoBERTa-large[d]	88.1	67.8	96.7	92.3/89.8	92.2/91.9	74.3/90.2	90.8/90.2	95.4	88.2	89.0
RoBERTa$_{large}$ + FreeLB[‡]	88.4	68.0	96.8	**93.1**/90.8	**92.3/92.1**	74.8/90.3	91.1/90.7	95.6	88.7	89.0
RoBERTa$_{large}$ + R-AT[§]	**88.6**	**69.0**	96.7	92.6/90.1	92.1/91.8	**75.0/90.5**	**91.1/90.9**	95.3	**89.9**	**89.7**
GLUE Human	87.1	66.4	97.8	86.3/80.8	92.7/92.6	59.5/80.4	92.0/92.8	91.2	93.6	95.9

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: [a]: (Devlin et al., 2019); [b]: (Liu et al., 2019a); [c]: (Yang et al., 2019); [d]: (Liu et al., 2019b); [f]: (Zhu et al., 2020).

robust and exhibits better generalization. Notably, our method is independent of the model structure, implying that R-AT can almost easily be applied to deep neural network models with different structures (e.g., rnn-based models, cnn-based models, and transformer-based models).

3.3 Training Process

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 $P_1^{\mu}(y|x + \delta_{adv})$ and $P_2^{\mu}(y|x + \delta_{adv})$. 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),

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

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

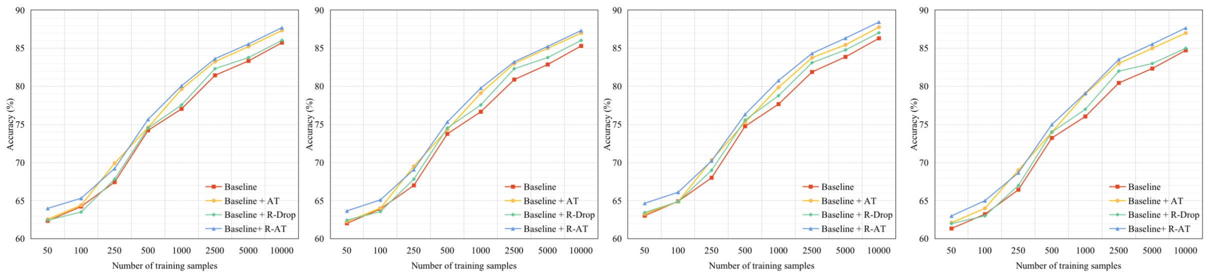
RoBERTa-base (110M) , ALBERT-base (12M) and RoBERTa-large (355M), we use the huggingface' Pytorch[2] implementation. Moreover, we use single-task finetuning for all dev set and test set results. The hyperparameters of our models are provided in the appendix A. For the non-pre-trained models, we utilized 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).

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. This experimental result also presents that the effects of R-AT method vary among different original deep learning models. As shown in Table 3, R-AT improves the average performance of the three baseline models on the four datasets by 2.31%, 1.18%, and 1.37%, respectively. Using R-AT strategy may

[2] https://github.com/huggingface/transformers



(a) Baseline model: BiLSTM (b) Baseline model: TextCNN (c) Baseline model: TextHAN (d) Baseline model: BERT-base

Figure 2: Comparison of performance on training sets of different sizes.

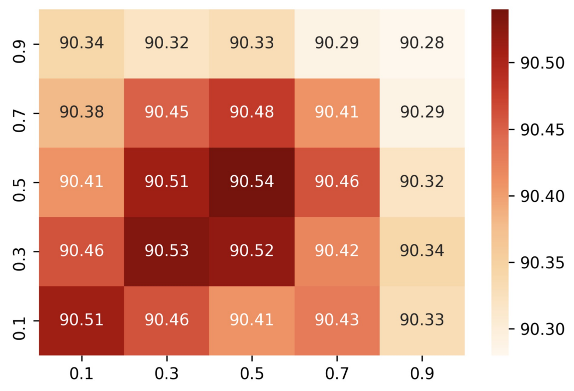


Figure 3: R-AT with two dropout rate combinations.

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-

Baselines	$K=2$	$K=3$	Gap
BiLSTM	90.54 ± 0.29	90.61 ± 0.27	+ 0.07
TextCNN	90.03 ± 0.18	90.05 ± 0.19	+ 0.02
TextHAN	91.71 ± 0.15	91.66 ± 0.23	- 0.05
BERT-base	89.58 ± 0.21	89.61 ± 0.23	+ 0.03

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

creased 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:

$$\mathcal{L}^i_{tot} = \mathcal{L}^i_{st} + \frac{1}{3}(\mathcal{L}^i_{adv_1} + \mathcal{L}^i_{adv_2} + \mathcal{L}^i_{adv_3}) + \beta\mathcal{L}^i_{KL}$$
$$\text{where } \mathcal{L}^i_{KL} = D^{bi}_{KL}(P^\mu_1 \parallel P^\mu_2) + D^{bi}_{KL}(P^\mu_1 \parallel P^\mu_3) + D^{bi}_{KL}(P^\mu_2 \parallel P^\mu_3) \tag{10}$$

The new weight coefficient β of KL loss is 1/3 of weight coefficient α.

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:

$$\begin{aligned}\min_\theta &\mathbb{E}_{(x,y)\sim D}\mathbb{E}_\mu[\mathcal{L}(\theta,\mu_{st},x,y)\\ &+ \max_{\delta_{adv}\in S}\mathcal{L}(\theta,\mu^{(1)}_{adv},x+\delta_{adv},y)\\ &+ \max_{\delta_{adv}\in S}\mathcal{L}(\theta,\mu^{(2)}_{adv},x+\delta_{adv},y)]\end{aligned} \tag{11}$$

$$\begin{aligned}s.t.\mathbb{E}_{(x,y)\sim D}\mathbb{E}_\mu[D^{bi}_{KL}(P_1^{\mu^{(1)}adv}(y|x+\delta_{adv}\\ \parallel P_2^{\mu^{(2)}adv}(y|x+\delta_{adv}))] = 0\end{aligned} \tag{12}$$

where μ is expressed as a random mask matrix of dropout, each dimension of which is independently sampled from a Bernoulli distribution. $\mu_{st}, \mu^{(1)}_{adv}, \mu^{(2)}_{adv}$ 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 $f(x+\delta_{adv}) = \mathcal{L}(\theta,\mu^{(1)}_{adv},x+\delta_{adv},y)$ and $f(x+\delta_{adv}) = \mathcal{L}(\theta,\mu^{(2)}_{adv},x+\delta_{adv},y)$ at points x:

$$\begin{aligned}&\min_\theta \mathbb{E}_{(x,y)\sim D}\mathbb{E}_\mu[\max_{\delta_{adv}\in S}[\mathcal{L}(\theta,\mu^{(1)}_{adv},x,y)+\\ &< \mathcal{L}(\theta,\mu^{(1)}_{adv},x,y),\delta_{adv} >] + \max_{\delta_{adv}\in S}[\mathcal{L}(\theta,\mu^{(2)}_{adv}\\ &,x,y)+ < \mathcal{L}(\theta,\mu^{(2)}_{adv},x,y),\delta_{adv} >]]\end{aligned} \tag{13}$$

Although the probability of $\mu_{st}, \mu^{(1)}_{adv}, \mu^{(2)}_{adv}$ is different in the dropout sampling process, if we weaken the dropout difference, $\mu_{st}, \mu^{(1)}_{adv}, \mu^{(2)}_{adv}$ are all approximately represented as μ. After substituting the values $\delta_{adv} = \epsilon \cdot \nabla_x\mathcal{L}(\theta,\mu,x,y)/ \parallel \nabla_x\mathcal{L}(\theta,\mu,x,y) \parallel_2$ that maximizes antagonistic loss, the following formula (15) is obtained:

$$\begin{aligned}&\min_\theta \mathbb{E}_{(x,y)\sim D}\mathbb{E}_\mu[\mathcal{L}(\theta,\mu,x,y)+ < \mathcal{L}(\theta,\mu,x,y),\\ &\epsilon\cdot\nabla_x\mathcal{L}(\theta,\mu,x,y)/ \parallel \nabla_x\mathcal{L}(\theta,\mu,x,y) \parallel_2> +\mathcal{L}\\ &(\theta,\mu,x,y)+ < \mathcal{L}(\theta,\mu,x,y),\epsilon\cdot\nabla_x\mathcal{L}(\theta,\mu,x,y)\\ &/ \parallel \nabla_x\mathcal{L}(\theta,\mu,x,y) \parallel_2>]\end{aligned} \tag{14}$$

$$\begin{aligned}=> \min_\theta \mathbb{E}_{(x,y)\sim D}\mathbb{E}_\mu[2(\mathcal{L}(\theta,\mu,x,y)\\ +\epsilon\cdot \parallel \nabla_x\mathcal{L}(\theta,\mu,x,y) \parallel_2)]\end{aligned} \tag{15}$$

We can observe the final optimization function formula (16), which actually adds a "gradient penalty" $\epsilon\cdot \parallel \nabla_x\mathcal{L}(\theta,\mu,x,y) \parallel_2$ 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:

$$\hat{y} = \sum_\mu P(\mu)P(y|x,\mu) \tag{16}$$

Here, the probability distribution $P(y|x,\mu)$ of each sub-model is defined by the mask matrix μ, and

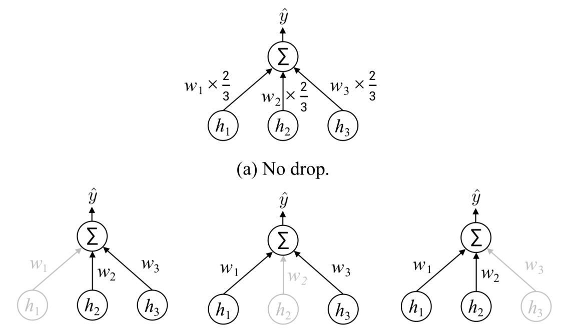

(a) No drop.
(b1) Drop the h_1 unit.
(b2) Drop the h_2 unit.
(b3) Drop the h_3 unit.

Figure 4: Weights average and sub-models' average.

$P(\mu)$ 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 $\hat{y}_m$:

$$\begin{aligned}\hat{y}_m &= \sum_{\mu} P(\mu)P(y|x,\mu) \\ &= \frac{1}{3}\times\sigma(w_2\cdot h_2 + w_3\cdot h_3) \\ &+ \frac{1}{3}\times\sigma(w_1\cdot h_1 + w_3\cdot h_3) \\ &+ \frac{1}{3}\times\sigma(w_1\cdot h_1 + w_2\cdot h_2)\end{aligned} \tag{17}$$

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

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

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).

$$\begin{aligned}(w_2\cdot h_2 + w_3\cdot h_3) &= (w_1\cdot h_1 + w_3\cdot h_3) \\ (w_1\cdot h_1 + w_3\cdot h_3) &= (w_1\cdot h_1 + w_2\cdot h_2) \\ (w_1\cdot h_1 + w_2\cdot h_2) &= (w_2\cdot h_2 + w_3\cdot h_3)\end{aligned} \tag{19}$$

$$=> w_1\cdot h_1 = w_2\cdot h_2 = w_3\cdot h_3. \tag{20}$$

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:

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

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

Eneko Agirre, Llu'is M'arquez, and Richard Wicentowski, editors. 2007. *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*. Association for Computational Linguistics.

Anish Athalye, Nicholas Carlini, and David Wagner. 2018. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International conference on machine learning*, pages 274–283. PMLR.

Jimmy Ba and Brendan Frey. 2013. Adaptive dropout for training deep neural networks. *Advances in neural information processing systems*, 26:3084–3092.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Roy Bar Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. 2006. The second PASCAL recognising textual entailment challenge.

Mikhail Belkin, Daniel J Hsu, and Partha Mitra. 2018. Overfitting or perfect fitting? risk bounds for classification and regression rules that interpolate. *Advances in Neural Information Processing Systems*, 31:2300–2311.

Luisa Bentivogli, Ido Dagan, Hoa Trang Dang, Danilo Giampiccolo, and Bernardo Magnini. 2009. The fifth PASCAL recognizing textual entailment challenge.

Rich Caruana, Steve Lawrence, and Lee Giles. 2000. Overfitting in neural nets: Backpropagation. In *Conjugate Gradient, and Early Stopping, NeuralInformation Processing Systems Conference, NIPS, Denver*, volume 408.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The PASCAL recognising textual entailment challenge. In *Machine learning challenges. evaluating predictive uncertainty, visual object classification, and recognising tectual entailment*. Springer.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

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

Dumitru Erhan, Pierre-Antoine Manzagol, Yoshua Bengio, Samy Bengio, and Pascal Vincent. 2009. The difficulty of training deep architectures and the effect of unsupervised pre-training. In *Artificial Intelligence and Statistics*, pages 153–160. PMLR.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third PASCAL recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. 2016. *Deep learning*, volume 1. MIT Press.

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

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Sepp Hochreiter and Jürgen Schmidhuber. 1995. Simplifying neural nets by discovering flat minima. In *Advances in neural information processing systems*, pages 529–536.

Lei Huang, Xianglong Liu, Bo Lang, Adams Wei Yu, Yongliang Wang, and Bo Li. 2018. Orthogonal weight normalization: Solution to optimization over multiple dependent stiefel manifolds in deep neural networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR.

Shankar Iyer, Nikhil Dandekar, and Kornl Csernai. 2017. First quora dataset release: Question pairs.

Guoliang Kang, Jun Li, and Dacheng Tao. 2016. Shakeout: A new regularized deep neural network training scheme. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105.

Anders Krogh and John A Hertz. 1992. A simple weight decay can improve generalization. In *Advances in neural information processing systems*, pages 950–957.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. *ICLR*.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature*, 521(7553):436–444.

Hector J Levesque, Ernest Davis, and Leora Morgenstern. 2011. The Winograd schema challenge. In *AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning*.

Weizhi Li, Gautam Dasarathy, and Visar Berisha. 2020. Regularization via structural label smoothing. In *International Conference on Artificial Intelligence and Statistics*, pages 1453–1463. PMLR.

Xiaobo Liang, Lijun Wu, Juntao Li, Yue Wang, Qi Meng, Tao Qin, Wei Chen, Min Zhang, and Tie-Yan Liu. 2021. R-drop: regularized dropout for neural networks. *arXiv preprint arXiv:2106.14448*.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019a. Multi-task deep neural networks for natural language understanding. In *ACL*.

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

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.

Takeru Miyato, Andrew M Dai, and Ian Goodfellow. 2017. Adversarial training methods for semi-supervised text classification. In *ICLR*.

Reza Moradi, Reza Berangi, and Behrouz Minaei. 2020. A survey of regularization strategies for deep models. *Artificial Intelligence Review*, 53(6):3947–3986.

Rafael Müller, Simon Kornblith, and Geoffrey Hinton. 2019. When does label smoothing help? *arXiv preprint arXiv:1906.02629*.

Andrew Y Ng et al. 1997. Preventing" overfitting" of cross-validation data. In *ICML*, volume 97, pages 245–253. Citeseer.

Shiwen Ni, Jiawen Li, and Hung-Yu Kao. 2021. Dropattack: A masked weight adversarial training method to improve generalization of neural networks. *arXiv preprint arXiv:2108.12805*.

Shiwen Ni, Jiawen Li, and Hung-Yu Kao. 2022. Hat4rd: Hierarchical adversarial training for rumor detection in social media. *Sensors*, 22(17):6652.

Ben Poole, Jascha Sohl-Dickstein, and Surya Ganguli. 2014. Analyzing noise in autoencoders and deep networks. *arXiv preprint arXiv:1406.1831*.

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

Tim Salimans and Durk P Kingma. 2016. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *Advances in neural information processing systems*, 29:901–909.

Ali Shafahi, Mahyar Najibi, Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. 2019. Adversarial training for free! In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pages 3358–3369.

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*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. 2013. Regularization of neural networks using dropconnect. In *International conference on machine learning*, pages 1058–1066. PMLR.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2019. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *ICLR*.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2018. Neural network acceptability judgments. *arXiv preprint 1805.12471*.

Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. 2016. Learning structured sparsity in deep neural networks. *Advances in neural information processing systems*, 29:2074–2082.

Roman Werpachowski, András György, and Csaba Szepesvari. 2019. Detecting overfitting via adversarial examples. *Advances in Neural Information Processing Systems*, 32:7858–7868.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *NAACL*.

Yuxin Wu and Kaiming He. 2018. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19.

Liang Xu, Hai Hu, Xuanwei Zhang, Lu Li, Chenjie Cao, Yudong Li, Yechen Xu, Kai Sun, Dian Yu, Cong Yu, et al. 2020. Clue: A chinese language understanding evaluation benchmark. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 4762–4772.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *NeurIPS*.

Chang-Bin Zhang, Peng-Tao Jiang, Qibin Hou, Yunchao Wei, Qi Han, Zhen Li, and Ming-Ming Cheng. 2021. Delving deep into label smoothing. *IEEE Transactions on Image Processing*, 30:5984–5996.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28:649–657.

Chen Zhu, Yu Cheng, Zhe Gan, Siqi Sun, Tom Goldstein, and Jingjing Liu. 2020. Freelb: Enhanced adversarial training for natural language understanding. In *ICLR*.

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 $3 \sim 9$. However, different datasets may have different optimal perturbation coefficient values.

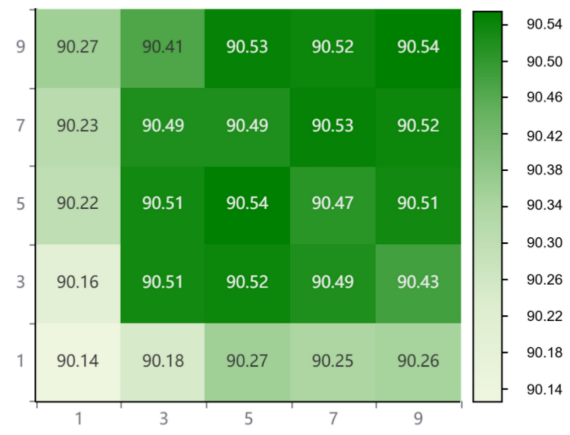


Figure 5: R-AT with two different perturbation coefficient combinations. The abscissa is the first dropout rate, and the ordinate is the second dropout rate.

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.

Corpus	|**Train**|	|**Test**|	**Task**	**Metrics**	**Domain**
			Single-Sentence Tasks		
CoLA	8.5k	**1k**	acceptability	Matthews corr.	misc.
SST-2	67k	1.8k	sentiment	acc.	movie reviews
			Similarity and Paraphrase Tasks		
MRPC	3.7k	1.7k	paraphrase	acc./F1	news
STS-B	7k	1.4k	sentence similarity	Pearson/Spearman corr.	misc.
QQP	364k	**391k**	paraphrase	acc./F1	social QA questions
			Inference Tasks		
MNLI	393k	**20k**	NLI	matched acc./mismatched acc.	misc.
QNLI	105k	5.4k	QA/NLI	acc.	Wikipedia
RTE	2.5k	3k	NLI	acc.	news, Wikipedia
WNLI	634	**146**	coreference/NLI	acc.	fiction books

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.

Dataset	Task description	Language	Classes	Dataset splitting		
				Train	Validation	Test
IMDB	Movie reviews sentiment analysis	English	2	40,000	5,000	5,000
AGNEWS	AG'News topic classification	English	4	110,000	10,000	7,600
TNEWS	Short Text Classificaiton for News	Chinese	15	53,360	10,000	10,000
IFLYTEK	Long Text topic classification	Chinese	119	12,133	2,599	2,600

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

Hyper-parameters	CoLA	SST-2	MRPC	STS-B	QQP	MNLI	QNLI	RTE	WNLI
α	0.5	0.5	0.8	0.5	0.5	0.8	0.8	0.5	0.5
ϵ	1E-1	1E-1	1.5E-1	1E-1	1E-1	1E-1	1.5E-1	1E-1	1.5E-1
τ	2E-5	1E-5	1E-5	1E-5	2E-5	2E-5	1E-5	2E-5	1E-5
Dropout	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
Batch size	24	32	32	32	32	32	32	24	16

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

Loss function	IMDB	AGNEWS	TNEWS	IFLYTEK	AVG.
CE Loss	90.13 ± 0.45	93.01 ± 0.31	52.89 ± 0.16	53.57 ± 0.23	72.40
MAE Loss	90.03 ± 0.17	92.72 ± 0.23	52.87 ± 0.27	53.52 ± 0.31	72.29
MSE Loss	89.97 ± 0.32	92.29 ± 0.19	53.47 ± 0.33	53.48 ± 0.37	72.30
KL Loss	90.24 ± 0.33	92.19 ± 0.13	52.43 ± 0.22	54.12 ± 0.38	72.25
JS Loss	90.44 ± 0.31	$\mathbf{93.08 \pm 0.11}$	53.77 ± 0.17	$\mathbf{54.42 \pm 0.35}$	72.93
Bi-KL Loss	$\mathbf{90.54 \pm 0.29}$	93.05 ± 0.03	$\mathbf{53.80 \pm 0.18}$	54.40 ± 0.37	**72.95**

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