Towards Adversarially Robust Text Classifiers by Learning to Reweight Clean Examples

Jianhan Xu[1,2]*, **Cenyuan Zhang**[1,2]*, **Xiaoqing Zheng**[1,2], **Linyang Li**[1,2], **Cho-Jui Hsieh**[3], **Kai-Wei Chang**[3], **Xuanjing Huang**[1,2]
[1]School of Computer Science, Fudan University, Shanghai, China
[2]Shanghai Key Laboratory of Intelligent Information Processing
[3]Department of Computer Science, University of California, Los Angeles, USA
{jianhanxu20,cenyuanzhang17,zhengxq}@fudan.edu.cn
{chohsieh,kwchang}@cs.ucla.edu

Abstract

Most of the existing defense methods improve the adversarial robustness by making the models adapt to the training set augmented with some adversarial examples. However, the augmented adversarial examples may not be natural, which might distort the training distribution, resulting in inferior performance both in clean accuracy and adversarial robustness. In this study, we explore the feasibility of introducing a reweighting mechanism to calibrate the training distribution to obtain robust models. We propose to train text classifiers by a sample reweighting method in which the example weights are learned to minimize the loss of a validation set mixed with the clean examples and their adversarial ones in an online learning manner. Through extensive experiments, we show that there exists a reweighting mechanism to make the models more robust against adversarial attacks without the need to craft the adversarial examples for the entire training set.

1 Introduction

Even though deep neural networks have achieved impressive performance on many natural language processing (NLP) tasks, they are vulnerable to adversarial examples intentionally crafted under certain semantic and syntactic constraints (Jia and Liang, 2017; Ebrahimi et al., 2017; Gao et al., 2018a; Zhao et al., 2018; Cheng et al., 2019; Zheng et al., 2020). The existence and pervasiveness of adversarial examples raise serious concerns, especially when deploying such NLP models to security-sensitive applications.

Recently, many methods have been proposed to defend against adversarial attacks for neural NLP models (Miyato et al., 2017a; Sato et al., 2018a; Jiang et al., 2020; Li and Qiu, 2020; Zhu et al., 2020; Zhou et al., 2021; Dong et al., 2021; Si et al., 2021). Existing defense methods usually augment the clean training examples with the adversarial ones in one way or another in the training stage and fit the models on the augmented training set. However, the introduced adversarial examples may not be natural, which may even hurt the distribution of original training examples, resulting in lower performance on both clean and adversarial test sets. Besides, these methods usually need to generate the adversarial examples for an entire training set, which is computationally intensive. We hypothesize that one of the reasons that NLP models are not robust is because they overfit to training data biases (Bras et al., 2020) — the training data is biased towards a certain distribution, so the resulting model can be broken under some perturbations. Despite augmenting training set by adversarial examples can partially mitigate this problem, are there better and more direct ways to calibrate the training distribution without introducing additional training samples? This motivates us to investigate whether there exists a reweighting mechanism to calibrate the training distribution and lead to robust models.

We propose to train adversarially robust text classifiers by a sample reweighting method, named WETAR (Weighting Examples Towards Adversarial Robustness), in which the example weights are learned to minimize the loss of a validation set mixed with the clean examples and their corresponding adversarial ones. We explore two ways to add adversarial samples to a validation set. A static way is to generate the adversarial examples from the clean data in the validation set before the training begins, and the generated examples remain unchanged throughout the entire training process. The other way is to dynamically craft adversarial examples at every iteration to test the robustness of models against test-time attacks.

Compared with exiting defense methods, our approach can achieve competitive performance without the need to perturb the training set. We show that there indeed exists a reweighting mechanism to

*Equal contribution

Findings of the Association for Computational Linguistics: ACL 2022, pages 1694 - 1707

make the models robust without enlarging the clean training set with any adversarial examples. We determine the example weights of the current batch at every training iteration by an online reweighting method that performs validation at an additional small size. This study is among the first ones to improve the adversarial robustness of NLP neural models by reweighing training examples under the guidance of a relatively small validation set. Through extensive experiments on three different text classification benchmark datasets, we show that our method can significantly increase the robustness to adversarial examples crafted by three representative adversarial attack algorithms.

2 Related Work

2.1 Text Adversarial Defense

The goal of adversarial defenses is to learn a model capable of achieving high test accuracy on both clean and adversarial examples. Recently, many defense methods have been proposed to defend against text adversarial attacks which can roughly be divided into two categories: *empirical* (Miyato et al., 2017b; Sato et al., 2018b; Zhou et al., 2021; Dong et al., 2021) and *certified* (Jia et al., 2019; Huang et al., 2019; Ye et al., 2020) methods.

Adversarial data augmentation is one of the most effective empirical defenses (Ren et al., 2019a; Jin et al., 2020; Li et al., 2020) for NLP models. During the training time, they replace a word with one of its synonyms to create adversarial examples. By augmenting these adversarial examples with the original training data, the model is robust to such perturbations. Zhou et al. (2021) and Dong et al. (2021) relax a set of discrete points (a word and its synonyms) to a convex hull spanned by the word embeddings of all these points, and use a convex hull formed by a word and its synonyms to capture word substitutions. Adversarial training (Miyato et al., 2017b; Zhu et al., 2020) is another one of the most successful empirical defense methods by adding norm-bounded adversarial perturbations to word embeddings and minimizes the resultant adversarial loss. The downside of existing empirical methods is that failure to discover an adversarial example does not mean that another more sophisticated attack could not find one. To address this problem, some certified defenses (Jia et al., 2019; Huang et al., 2019; Ye et al., 2020) have been introduced to guarantee the robustness to certain specific types of attacks. However, the existing certified defense methods make an unrealistic assumption that the defenders can access the synonyms used by the adversaries. They would be broken by more sophisticated attacks by using synonym sets with large sizes (Jin et al., 2020) or generating synonyms dynamically with BERT (Li et al., 2020).

Most of the existing defense methods improve the robustness by making the models adapt to the training set augmented with adversarial examples crafted by adding adversarial perturbations to discrete tokens or distributed embeddings. In contrast, our method does not need to generate adversarial examples for the entire training set and only requires a relatively small validation set to be augmented with the adversarial instances. Besides, we improve the adversarial robustness by learning to assign weights to training examples based on the loss estimated on a validation set instead of exposing the models to certain perturbations during the training process.

2.2 Weighting Examples towards Robustness

Various methods of weighting examples have been proposed to train robust models against training set bias including class imbalance (Lin et al., 2017; Cui et al., 2019) or noisy data (Shin et al., 2020; Wang et al., 2021) or both (Ren et al., 2018). In response to these problems, different weights are assigned to examples in order to match one distribution to another, and the models are trained to optimize the weighted training loss encouraging learning the examples with more weights.

Recently, incorporating the weighting method to improve the robustness against adversaries also have been investigated in the image domain. However, they all use the weighting method to assign weights to the adversarial examples instead of the clean examples. Wang et al. (2020) weight training examples in order to reduce the KL-divergence between the predicted logits of each clean example and that of the adversarial one. Zhang et al. (2021) take into account the geometric distance from data points to the decision boundary and reweight training data based on the difficulty of attacking these data points. To better defend against targeted adversarial attacks, Kim et al. (2021) proposed to reweight training examples based on the entropies of their class softmax probabilities and suggested giving more weights to the examples with higher entropies whose labels could be easily flipped.

Different from existing reweighting methods, we

argue that in order to train a model that performs well in both clean accuracy and adversarial robustness, it only needs to construct a small validation set augmented with adversarial examples to guide training. In addition, we show that the adversarial examples can be added to the original validation set in a static or dynamic way. The constructed validation also can be used for the model selection. This study is among the first ones to improve the adversarial robustness of neural models by reweighing training examples in the language domain.

3 Method

We design a weighting method to improve the adversarial robustness of text classifiers by learning to reweight examples, partly inspired by the meta-learning algorithm proposed by Ren et al. (2018) from the image domain. In particular, we consider both clean accuracy and adversarial robustness by reweighting the training examples according to their similarity to the gradient descent of the validation loss, where the validation set is augmented with the adversarial examples. During the training, we ensure that a clean example and its corresponding adversary are present in the same mini-batch, which teaches the models how to balance the two training objectives. We also show how to make model selection based on the learned weight distribution over the training examples.

For text classification, a neural network-based classifier $f(x)$ with a set of learnable parameters $\boldsymbol{\theta}$ maps an input text $x \in \mathcal{X}$ to a label $y \in \mathcal{Y}$. Given a training set $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$, we assume there is a validation set $\mathcal{D}^v = \{(x_i^v, y_i^v)\}_{i=1}^{M}$ that consists of two parts: a set of clean examples $\mathcal{D}^c$, and a set of adversarial examples $\mathcal{D}^a$ generated by a certain attack algorithm from $\mathcal{D}^c$. The adversarial validation set $\mathcal{D}^a$ can be generated for $\mathcal{D}^c$ statically or dynamically (see Subsection 3.2). We consider a loss function $L_{\boldsymbol{\theta}}(x, y)$, and the goal of regular training is to find a solution of $\boldsymbol{\theta}$ that minimizes the expected loss $\frac{1}{N}\sum_{i=1}^{N} L_{\boldsymbol{\theta}}(x_i, y_i)$ for the training set, where each instance is equally weighted.

3.1 Learning to Reweight Examples

In this study, we guide the training by a relatively small validation set $\mathcal{D}^v$ mixed with clean and adversarial examples through a weighted loss. Thus, each training example x_i would be assigned with a weight w_i, and we learn to reweight the examples by minimizing the following weighted loss:

$$\boldsymbol{\theta}^*(\boldsymbol{w}) = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{N} w_i L_{\boldsymbol{\theta}}(x_i, y_i), \tag{1}$$

where $\boldsymbol{w} = \{w_i\}_{i=1}^{N}$ can be viewed as training hyperparameters whose values are unknown from the beginning and can be optimized based on the validation set $\mathcal{D}^v$:

$$\boldsymbol{w}^* = \arg\min_{w_i \geq 0} \frac{1}{M} \sum_{i=1}^{M} L_{\boldsymbol{\theta}^*(\boldsymbol{w})}(x_i^v, y_i^v), \tag{2}$$

where we use superscript v to denote validation set and subscript i to denote the i-th example.

Determining the optimal $\boldsymbol{w}^*$ is a special case of the bilevel optimization problem where one problem is nested within another, and every single optimization can be very expensive. It could be worse when the adversarial validation set is created in a dynamic way, which is necessary to enhance the models in their ability to defend against test-time attacks. We use an online meta-learning algorithm (Ren et al., 2018) for reweighting training examples. At every training step, a mini-batch $\{x_i, y_i\}_{i=1}^{n}$ is sampled from the training set $\mathcal{D}$, and n is the mini-batch size ($n \ll N$). At the same time, another mini-batch $\{x_i^v, y_i^v\}_{i=1}^{m}$ is also sampled from the validation set $\mathcal{D}^v$. We examine the gradient descent of n sampled training examples on the loss surface and reweight them according to their similarity to the descent direction of m validation data.

We need to determine the importance of each training sample (x_i, y_i) at every training step for a mini-batch sampled from $\mathcal{D}^v$. Following (Koh and Liang, 2017), we assume the weight of each training sample (x_i, y_i) is perturbed by ϵ_i, and correspondingly the parameters are updated to $\boldsymbol{\theta}'_t$ according to the descent direction of the loss on the mini-batch at step t as follows:

$$\boldsymbol{\theta}'_t = \boldsymbol{\theta}_t - \tau \nabla_{\boldsymbol{\theta}_t} \sum_{i=1}^{n} \epsilon_i L_{\boldsymbol{\theta}_t}(x_i, y_i), \tag{3}$$

where τ is the learning rate. To get a cheap estimate of w_i at step t, we calculate the gradient g_{ϵ_i} of ϵ_i by taking a single gradient descent step on a mini-batch of validation samples [1] :

$$g_{\epsilon_i} = \frac{\partial}{\partial \epsilon_i} \frac{1}{m} \sum_{i=j}^{m} L_{\boldsymbol{\theta}'_t}(x_j^v, y_j^v)|_{\epsilon_i = 0}. \tag{4}$$

[1] We use a meta-learning paradigm in this step to calculate the gradient g_{ϵ_i} for each ϵ_i. Specifically, we use the *higher* library (Grefenstette et al., 2019) released by Facebook.

We then estimate that the importance value $\tilde{w}_i$ of sample (x_i, y_i) at step t by comparing the opposite direction of gradient g_{ϵ_i} accumulated at ϵ_i when take adversarial validation mini-batch $\{x_i^v, y_i^v\}_{i=1}^m$ into account:

$$\begin{aligned} \tilde{w}_i &= max(-g_{\epsilon_i}, 0), \\ w_i &= \frac{\tilde{w}_i}{\sum_i \tilde{w}_i + \delta}. \end{aligned} \tag{5}$$

where $\delta = 1$ if $\sum_i \tilde{w}_i = 0$, and $\delta = 0$ otherwise. We consider normalizing the weights of all training examples in a mini-batch so that they sum up to one unless all of them are 0. Once each training example is assigned with a weight under the guidance of the gradients calculated on the validation samples, we can update the parameters with the gradient accumulated through the reweighted loss at step t as follows:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \tau \nabla_{\boldsymbol{\theta_t}} \sum_{i=1}^{n} w_i L_{\boldsymbol{\theta}_t}(x_i, y_i). \tag{6}$$

We refer to Algorithm 1 in Appendix A for details.

3.2 Adversarial Validation Set Construction

We here describe two ways to construct a validation set whose subset of adversarial examples can be generated in a static or dynamic manner. If a clean example (x_i^c, y_i^c) is sampled to be included in a validation mini-batch, we would add into the same mini-batch its adversarial example (x_i^a, y_i^a), where $y_i^c = y_i^a$, crafted by some attack algorithms .

In the static construction method, for every clean example in the validation set, its corresponding adversarial one is generated before the training begins, and the generated adversarial examples remain unchanged throughout the training process. If a clean example is randomly selected to appear in a mini-batch, its adversary generated in advance will be retrieved and included in the same mini-batch.

Although generating the adversarial examples in a static way can speed up the training process, it is questionable whether the resulting models can still perform well under test-time attacks since they should be evaluated on the adversarial examples crafted on the fly against the robustly trained models rather than the original ones. Therefore, we propose to use another dynamic strategy to generate adversarial examples, in which we apply an attack algorithm to craft the adversarial examples for randomly selected clean ones against the current model at every iteration. In practice, we generate all required adversarial examples every one or two epochs to reduce the computational cost.

Some previous studies show that the models tend to overfit the adversarial examples, and their performance on the clean data will drop if too many adversarial examples are used. Therefore, we use a similar training strategy. In a mini-batch, we randomly select ρ percent (say 50%) of the clean data in the validation set and generate their adversarial examples from them using a certain attack algorithm. We then merge these adversarial examples with the clean ones to form a final validation set.

If the static method is used to construct the validation set, the weight distribution of training examples will stabilize to some equilibrium distribution as the number of training epochs increases. Such a weight distribution is calculated for each epoch by accumulating the weights assigned to the sampled examples in every mini-batch and then normalizing the weights of all training examples to sum up to one. To compute the difference between two weight distributions before and after an epoch, we use the Wasserstein distance instead of the popular KL-divergence since the weights of many training examples will be assigned to zeros and the former is more suitable for this situation than the latter. As the training progresses, we can obtain a series of weight distributions and their differences. If such a difference does not reduce significantly for multiple epochs, we say that the distribution has stabilized. The first model obtained with its weight distribution stabilized is chosen as the final model. When the dynamic construction method is used, there will be a "spear-and-shield" battle between defender and attacker. Although the difference in weight distribution fluctuates more with the dynamic construction method than the static one, the trend of the overall decline in the distribution difference still can be used for model selection.

4 Experiments

In the following, we first evaluate the proposed method of WETAR by comparing it to four baseline methods both in clean accuracy and adversarial robustness on three text classification benchmarks. Then, we would like to study how the choice of an attack algorithm to construct the validation sets at the training stage impact the adversarial robustness of resulting models under different attacks. Finally, we investigate whether our method combined with adversarial data augmentation can further improve

Datasets	Methods	*Clean*%	TextFooler			BERT-Attack			TextBugger		
			Aua%	*Suc*%	*#Query*	*Aua*%	*Suc*%	*#Query*	*Aua*%	*Suc*%	*#Query*
SST-2	Base	93.2	5.6	94.0	89.2	6.2	93.3	111.7	27.9	69.8	47.5
	FreeLB	**93.9**	8.5	91.4	95.4	9.2	90.2	118.6	32.0	65.9	49.5
	FreeLB++	92.9	14.2	84.8	117.6	11.7	87.4	139.6	36.7	60.5	51.9
	ADA	89.5	17.8	80.0	80.0	13.8	84.5	141.4	30.7	65.5	53.6
	WETAR-S	92.9	26.2	71.7	145.3	24.3	73.5	183.9	51.7	44.1	56.6
	WETAR-D	93.4	**29.6**	**68.2**	**153.2**	**31.4**	**66.2**	**207.2**	**55.3**	**40.6**	**57.1**
AGNEWS	Base	94.3	19.6	79.3	329.3	22.9	75.9	432.3	41.4	56.3	186.8
	FreeLB	94.9	28.0	70.4	380.3	29.0	69.4	479.8	47.9	49.4	196.4
	FreeLB++	**95.1**	32.0	66.6	412.9	29.9	68.8	487.9	53.1	44.4	193.2
	ADA	94.6	41.1	56.5	424.4	32.6	65.5	508.7	52.0	45.1	220.8
	WETAR-S	94.1	47.7	49.4	464.0	56.4	40.1	**602.3**	68.5	**27.2**	**242.9**
	WETAR-D	94.2	**54.4**	**42.5**	**472.0**	**57.5**	**39.2**	595.0	**68.8**	**27.2**	235.7
MR	Base	87.2	8.3	90.5	107.9	9.9	88.7	141.7	29.9	65.7	53.6
	FreeLB	88.0	8.4	90.5	111.0	9.2	89.6	139.6	31.8	63.9	54.4
	FreeLB++	**88.3**	11.9	86.6	122.4	11.0	87.6	153.4	34.2	61.3	56.2
	ADA	85.1	14.3	83.1	128.2	11.3	86.7	148.3	34.0	60.0	57.8
	WETAR-S	86.6	**30.4**	**64.9**	**156.6**	**31.7**	**63.4**	**206.9**	48.1	44.6	63.7
	WETAR-D	86.2	24.9	71.2	151.9	28.4	67.1	203.4	**48.7**	**43.6**	**62.9**

Table 1: The experimental results of our WETAR and baselines on SST-2, AGNEWS, and MR datasets. We use WETAR-S to denote the setting where the adversarial examples are constructed by the static method and WETAR-D to that by the dynamic method. The best results are highlighted in bold font.

the robustness of text classifiers.

We conducted experiments on two different tasks on three widely-used datasets: Stanford Sentiment Treebank (SST-2) (Socher et al., 2013), AG-News corpus (AGNEWS) (Zhang et al., 2015) and Movie Reviews (MR) (Pang and Lee, 2005). SST-2 consists about $67,000$ training sentences for binary classification and MR contains about $9,000$ movie reviews for training. AGNEWS has four categories pertaining about $30,000$ new articles. For each dataset, we randomly select one-tenth examples from the training set to form a validation set from which the adversarial examples will be generated to guide the training and select the model. In Section 4, all experimental results are obtained over three runs with different initialization. We refer to Appendix B for more implementation details.

4.1 Attack Algorithms

The following three adversarial attack methods are used to evaluate the robustness of models, reimplemented by TextAttack toolkit (Morris et al., 2020).

TextFooler (Jin et al., 2020) uses a greedy searching method, ranking the words in an input sequence based on the predicted changes before and after deleting them. Counter-fitted embeddings are used to find synonyms to replace the selected words.

BERT-Attack (Li et al., 2020) uses a BERT-based model to estimate an importance score of each subword for the prediction, and generate the top-K candidate sub-words by the masked language model to replace the word with the highest score.

TextBugger (Li et al., 2019) locates the vulnerable words by calculating the changes in predictions before and after removing them from a text. Different from TextFooler and BERT-Attack, both character-level perturbation and word-level perturbation will be applied to generate adversarial examples.

When investigating how the choice of an attack algorithm to construct the validation set impact the performance of models, we also take two other attack methods of PWWS (Ren et al., 2019b) and DeepWordBug (Gao et al., 2018b) into consideration for comprehensive assessment.

Following (Li et al., 2021), four different metrics are used to evaluate the generation and robustness of the models: (1) Clean accuracy, denoted as *Clean*%, is defined as the model's classification accuracy on a clean test set; (2) Accuracy under attacks, denoted as *Aua*%, is the model's accuracy under some adversarial attack; (3) Attack success rate, denoted as *Suc*%, is calculated as the number of texts successfully perturbed by an attack algorithm divided by the number of all texts attempted; (4) The number of queries, denoted as *Query*%, is the average number of times the attacker queries the model to form a successful attack.

4.2 Baseline methods

We evaluate the proposed method by comparing it with several representative methods. We primarily compare with the following recently proposed

defense methods,

Base fine-tunes a pre-trained BERT on a training set consisting of clean examples.

FreeLB (Zhu et al., 2020) adds norm-bounded adversarial perturbations to the input's word embeddings using a gradient-based method, and enlarges the batch size with diversified adversarial samples under such norm constraints.

FreeLB++ is a variant of FreeLB, which increases the number of ascent steps to further improve the adversarial robustness of models (Li et al., 2021). They demonstrated through extensive experiments that FreeLB and its variant of FreeLB++ outperforms other defense methods including TAVAT (Li and Qiu, 2020) and DNE (Zhou et al., 2021). Therefore, we only report the results produced by FreeLB and FreeLB++ for comparison.

Adversarial Data Augmentation (ADA) is one of the widely used methods (Dong et al., 2021; Si et al., 2021; Zhou et al., 2021). During the training, they replace a word with one of its synonyms that maximizes the prediction loss. By augmenting these adversarial examples with the original training data, the model is robust to such perturbations.

4.3 Results

Table 1 shows the clean accuracy and adversarial robustness achieved by different defense methods under three attack algorithms. We use TextFoolor as the attack algorithm to generate the adversarial examples for validation set construction because it was reported that TextFoolor can generate high-quality, semantics-preserved adversarial examples (Hauser et al., 2021). For a fair comparison, ADA also use TextFoolor to craft the adversarial examples for data augmentation. Unless otherwise specified, we set to 50% the percent of the clean data in the validation set from which the corresponding adversarial examples will be generated. We use WETAR-S to denote the setting where the adversarial examples are constructed by the static method and WETAR-D to that by the dynamic method.

From these numbers, a handful of trends are readily apparent: (1) The proposed WETAR achieved the highest robustness across three text classification datasets under different adversarial attacks over all the baseline methods while suffering little to no performance drop on the clean input data; (2) The models trained with FreeLB++ achieved the better performance than others in clean accuracy. However, the improvement in adversarial robustness is relatively small compared to WETAR; (3) The models trained with ADA method outperformed those trained with other baseline methods in adversarial robustness, but they suffer a significant drop in clean accuracy, especially on SST-2 and MR datasets[2].

WETAR-D performed better than WETAR-S on SST-2 and AGNEWS datasets while the latter outperformed the former on MR dataset. One possible explanation is that the size of MR training set is much smaller than those of SST-2 and AGNEWS. For a given maximum number of training epochs, the number of mini-batches is relatively small when the models are trained on MR dataset. It would be hard for WETAR-D to tune the models sufficiently within a limited number of epochs since WETAR-D introduces the dynamics into the training process and requires more training epochs to converge.

4.4 Impact of the Types of Augmented Adversarial Examples

To better understand the impact of different attack algorithms used to construct the adversarial validation examples on the performance, we report in Table 2 accuracy achieved by WETAR and ADA methods under different attacks on the MR dataset. We found that both WETAR-D and WETAR-S perform better than ADA under almost all attack algorithms. Besides, WETAR is not sensitive to the choice of the attack algorithm used to construct validation set at the training stage, whereas ADA shows to be more sensitive to the type of attack algorithm applied to generate adversarial examples for data augmentation. Although the choice of attack algorithm has little impact on the adversarial robustness, the models trained by WETAR integrated with BERT-Attack achieved slightly better performance on MR dataset.

4.5 Impact of the Proportion of Adversarial Examples in the Validation Set

We conducted some experiments on SST-2 dataset to investigate the impact of different proportions of adversarial examples in the validation set. Figure 1 shows the clean accuracy and accuracy under attack of our WETAR where BERT-Attack was used to generate adversarial examples for validation set construction. We evaluated the adversarial robustness of the resulting models with TextFooler.

[2]In our experiments, we found our implemented ADA can achieve higher robustness than that reported in Li et al. (2021).

Method	Clean	TFL	BTA	TBG	PWWS	DWB
ADA-TFL	85.1	14.3	11.3	34.0	34.1	23.9
ADA-BTA	84.0	8.6	24.1	23.7	24.0	19.7
ADA-TBG	**87.0**	8.7	10.9	29.4	26.3	14.5
Generating Adversarial Examples Statically						
WETAR-S-TFL	86.6	30.4	31.7	48.1	39.8	43.3
WETAR-S-BTA	86.3	**32.1**	**32.4**	48.1	**41.4**	**43.4**
WETAR-S-TBG	86.0	22.6	27.0	43.3	35.2	36.8
Generating Adversarial Examples Dynamically						
WETAR-D-TFL	86.2	24.9	28.4	**48.7**	38.9	39.2
WETAR-D-BTA	86.6	26.0	28.0	45.6	38.2	38.6
WETAR-D-TBG	86.5	29.8	32.1	48.3	40.8	41.2

Table 2: Accuracy achieved with various training methods under different attacks on the MR dataset. Those listed in the rows are training methods, and those in the columns are attacking algorithms. "Clean" denotes the clean accuracy. "TFL", "BTA", "TBG", and "DWB" denotes TextFooler, BERT-Attack, TextBugger and DeepWordBug respectively.

As shown in Figure 1, we found that WETAR in general can provide a great increase in robustness only with little sacrifice in clean accuracy. WETAR is also insensitive to the proportions of adversarial examples that are added into the validation set. However, adding too many adversarial examples to the validation set will hurt the performance of models in both clean accuracy and adversarial robustness. We do get surprised that our method using a validation set that only contains clean examples can make the models more robust against the word substitution-based attacks. We believe that our training method can prevent the models from overfitting to a pre-defined training set, which leads to more robust models.

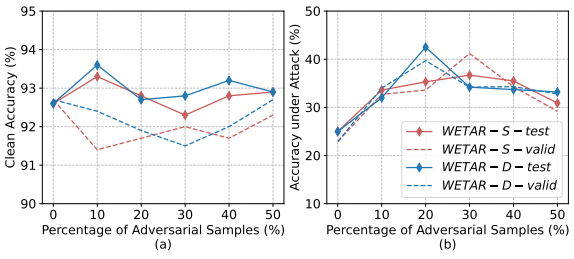


Figure 1: The impact of the proportion of adversarial samples in the validation set on MR dataset. (a) clean accuracy versus various proportions of adversarial samples. (b) the accuracy under attack versus various proportions of adversarial samples.

4.6 Combined Approach

We carried out some experiments to study whether the robustness of models can be further improved by combining WETAR with the data augmentation method. In this combination approach, we add some adversarial examples to the training set as the adversarial data augmentation. During the training process, WETAR will assign the weights to both the clean and adversarial examples based on the gradient direction of a small validation set.

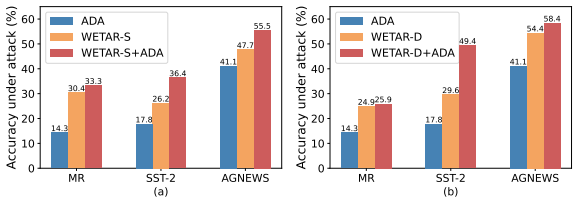


Figure 2: The robustness results achieved by WETAR combined with adversarial data augmentation. (a) WETAR with the static construction method. (b) WETAR with the dynamic construction method.

We show in Figure 2 the experimental results achieved by WETAR with validation set generated statically and dynamically under TextFooler attack on three datasets. Augmenting the training set with adversarial examples can further improve the robustness of models no matter WETAR-S or WETAR-D is used for training. Like the results reported in Section 4.3, WETAR-D outperformed WETAR-S on SST-2 and AGNEWS while the latter performed better than the former on MR dataset.

5 Analysis

We in this section give some analyses on the interpretability of the proposed reweighting method. First, we experimentally analyze the changes in the weight distributions over the training samples produced by our reweighting method. Base on this analysis, we propose an empirical method for model selection. Second, we visualize the weights obtained by the proposed method.

5.1 Weight Distribution

To better understand the changes in the weight distributions over the training examples, we show in Figure 3 the weight distributions produced by WETAR at different epochs. Those weight distributions are obtained by normalizing the weights of all training samples at each epoch. To show whether and how those weight distributions will converge to some distribution, we use the Wasserstein distance to compute the difference between two weight distributions before and after each epoch. We remove all examples with zeros weights when visualizing the distributions.

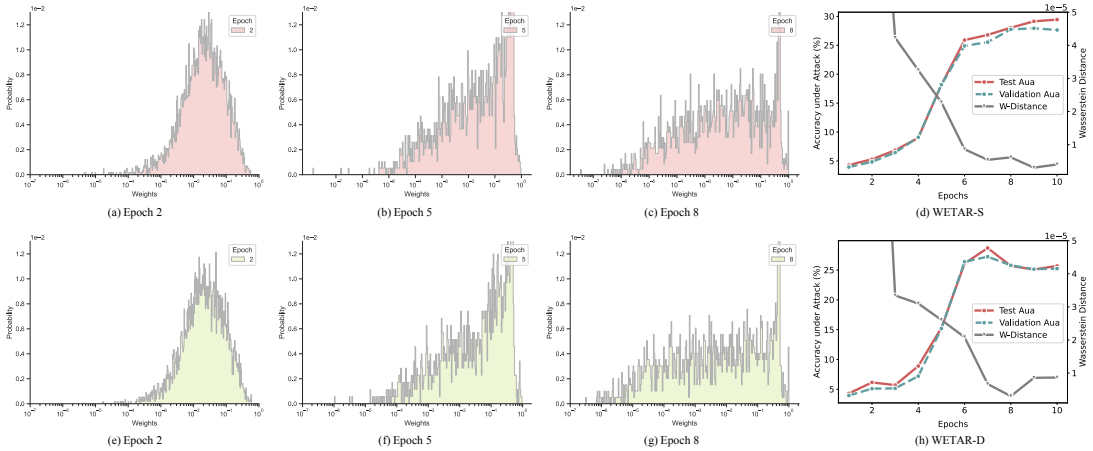


Figure 3: The weight distributions produced by WETAR-S and WETAR-D on MR datasets. Sub-figures (a), (b), and (c) show the weight distributions produced by WETAR-S at epoch 2, 5, and 8 respectively. Sub-figures (e), (f), and (g) give the same distributions produced by WETAR-D at epoch 2, 5, and 8. Sub-figures (d) and (h) plot the curves of accuracy under attack and the Wasserstein distance between two weight distributions at every two epochs yielded by WETAR-S and WETAR-D respectively.

As shown in Figure 3, we found that the performance of robustness generally increases when the differences in the weight distributions shrink as the number of epochs grows. For examples, the weight distribution starts to converge after 8 epochs when WETAR-S is used to train the model. Therefore, we select the model as the final one when the distance between two weight distributions is small (e.g., less than a given threshold). When WETAR-D is applied, there are some drops at the end of training process in adversarial robustness. One possible explanation is that after a long "spear-and-shield" battle, it is hard for any attack algorithm to generate good adversarial examples for a robust model, and the generated adversarial examples after that will go too far from the original ones, which hurts the decision boundary of the model and results in inferior performance in robustness. We give more experimental results about the weight distributions on AGNEWS and SST-2 in Appendix C.

5.2 Visualization

Learning to reweight scheme assigns different weights to examples in a mini-batch under the guidance of the adversarial validation set during the training phase. We illustrate the proposed reweighting process in Figure 4 to provide a better understanding of our method.

In the Figure 4, we plot the representations using t-SNE visualization by analyzing the final hidden states corresponding to `[CLS]` token of the model in the last training epoch. Visualization of more epochs are provided in Appendix D. In the t-SNE analysis, we use the average weights of data points during training as a measure of its importance. We normalized the average weights by the maximum weight this weight distribution could achieve. In Figure 4, the darker the color, the more larger weight this sample point is calculated into the loss function during training, and the more important it is in the training process.

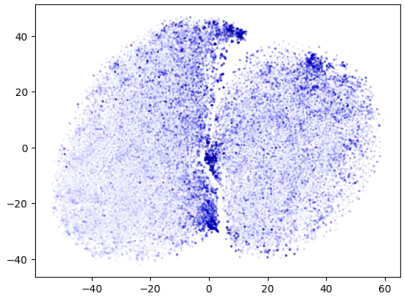


Figure 4: t-SNE visualization of the final hidden states corresponding to `[CLS]` token of SST-2 training examples produced by the model in trained via WETAR.

We found that the samples distributed in one category, while close to the other one have greater non-zero weight values, which indicates they are relatively more important than the others, coinciding with the finding mentioned by Zhang et al. (2021); Kim et al. (2021). We can also observe that more weights will be given to some samples during the training process, making the whole weight distribution sparse. Through the analysis of visualization, we found the proposed mechanism can prevent the

model from being overfitted to some training samples by assigning small weights to them. Guiding the training process via these sparse weight distributions leads to a significant increase in adversarial robustness with no or little drop in clean accuracy.

6 Conclusion

In this study, we propose a defense method against text adversarial attacks by reweighting examples automatically. The algorithm learns to weight training examples in proportion to their contributions to minimize the loss evaluated on a validation set augmented with adversarial examples. The proposed method can directly be applied to any deep learning architecture without any additional hyperparameter search. We showed through extensive experiments that there indeed exists a reweighting mechanism to make the model robust without generating adversarial examples for the entire training set, and our reweighting algorithm performs better than existing defense methods across three different text classification datasets.

Acknowledgements

The authors would like to thank the anonymous reviewers for their valuable comments. This work was supported by National Science Foundation of China (No. 62076068), Shanghai Municipal Science and Technology Major Project (No. 2021SHZDZX0103), and Zhangjiang Lab. Chang is supported in part by Cisco and Sloan fellowship. Hsieh is supported in part by NSF IIS-2008173 and IIS-2048280.

References

Ronan Le Bras, Swabha Swayamdipta, Chandra Bhagavatula, Rowan Zellers, Matthew Peters, Ashish Sabharwal, and Yejin Choi. 2020. Adversarial filters of dataset biases. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1078–1088. PMLR.

Minhao Cheng, Wei Wei, and Cho-Jui Hsieh. 2019. Evaluating and enhancing the robustness of dialogue systems: A case study on a negotiation agent. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3325–3335, Minneapolis, Minnesota. Association for Computational Linguistics.

Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. 2019. Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Xinshuai Dong, Anh Tuan Luu, Rongrong Ji, and Hong Liu. 2021. Towards robustness against natural language word substitutions. In *International Conference on Learning Representations*.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2017. Hotflip: White-box adversarial examples for text classification. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.

Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. 2018a. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 50–56.

Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. 2018b. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 50–56. IEEE.

Edward Grefenstette, Brandon Amos, Denis Yarats, Phu Mon Htut, Artem Molchanov, Franziska Meier, Douwe Kiela, Kyunghyun Cho, and Soumith Chintala. 2019. Generalized inner loop meta-learning.

Jens Hauser, Zhao Meng, Damián Pascual, and Roger Wattenhofer. 2021. Bert is robust! a case against synonym-based adversarial examples in text classification.

Po-Sen Huang, Robert Stanforth, Johannes Welbl, Chris Dyer, Dani Yogatama, Sven Gowal, Krishnamurthy Dvijotham, and Pushmeet Kohli. 2019. Achieving verified robustness to symbol substitutions via interval bound propagation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4083–4093, Hong Kong, China. Association for Computational Linguistics.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031, Copenhagen, Denmark. Association for Computational Linguistics.

Robin Jia, Aditi Raghunathan, Kerem Göksel, and Percy Liang. 2019. Certified robustness to adversarial word substitutions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4129–4142, Hong Kong, China. Association for Computational Linguistics.

Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. 2020. SMART: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online. Association for Computational Linguistics.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is BERT really robust? a strong baseline for natural language attack on text classification and entailment. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8018–8025.

Minseon Kim, Jihoon Tack, Jinwoo Shin, and Sung Ju Hwang. 2021. Entropy weighted adversarial training. In *ICML 2021 Workshop on Adversarial Machine Learning*.

Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1885–1894. PMLR.

Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. 2019. Textbugger: Generating adversarial text against real-world applications. *Proceedings 2019 Network and Distributed System Security Symposium*.

Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. 2020. BERT-ATTACK: Adversarial attack against BERT using BERT. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6193–6202, Online. Association for Computational Linguistics.

Linyang Li and Xipeng Qiu. 2020. Tavat: Token-aware virtual adversarial training for language understanding. *arXiv preprint arXiv:2004.14543*.

Zongyi Li, Jianhan Xu, Jiehang Zeng, Linyang Li, Xiaoqing Zheng, Qi Zhang, Kai-Wei Chang, and Cho-Jui Hsieh. 2021. Searching for an effective defender: Benchmarking defense against adversarial word substitution. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3137–3147, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

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

Takeru Miyato, Andrew M. Dai, and Ian J. Goodfellow. 2017b. Adversarial training methods for semi-supervised text classification. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

John Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020. TextAttack: A framework for adversarial attacks, data augmentation, and adversarial training in NLP. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 119–126, Online. Association for Computational Linguistics.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 115–124, Ann Arbor, Michigan. Association for Computational Linguistics.

Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. 2018. Learning to reweight examples for robust deep learning. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4334–4343. PMLR.

Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. 2019a. Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1085–1097, Florence, Italy. Association for Computational Linguistics.

Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. 2019b. Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 1085–1097.

Motoki Sato, Jun Suzuki, Hiroyuki Shindo, and Yuji Matsumoto. 2018a. Interpretable adversarial perturbation in input embedding space for text. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 4323–4330. International Joint Conferences on Artificial Intelligence Organization.

Motoki Sato, Jun Suzuki, Hiroyuki Shindo, and Yuji Matsumoto. 2018b. Interpretable adversarial perturbation in input embedding space for text. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 4323–4330. ijcai.org.

Wonyoung Shin, Jung-Woo Ha, Shengzhe Li, Yongwoo Cho, Hoyean Song, and Sunyoung Kwon. 2020. Which strategies matter for noisy label classification? insight into loss and uncertainty.

Chenglei Si, Zhengyan Zhang, Fanchao Qi, Zhiyuan Liu, Yasheng Wang, Qun Liu, and Maosong Sun. 2021. Better robustness by more coverage: Adversarial and mixup data augmentation for robust finetuning. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1569–1576, Online. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Wenjie Wang, Fuli Feng, Xiangnan He, Liqiang Nie, and Tat-Seng Chua. 2021. Denoising implicit feedback for recommendation. *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*.

Yisen Wang, Difan Zou, Jinfeng Yi, James Bailey, Xingjun Ma, and Quanquan Gu. 2020. Improving adversarial robustness requires revisiting misclassified examples. In *International Conference on Learning Representations*.

Mao Ye, Chengyue Gong, and Qiang Liu. 2020. SAFER: A structure-free approach for certified robustness to adversarial word substitutions. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3465–3475, Online. Association for Computational Linguistics.

Jingfeng Zhang, Jianing Zhu, Gang Niu, Bo Han, Masashi Sugiyama, and Mohan Kankanhalli. 2021. Geometry-aware instance-reweighted adversarial training. In *International Conference on Learning Representations*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Zhengli Zhao, Dheeru Dua, and Sameer Singh. 2018. Generating natural adversarial examples. In *International Conference on Learning Representations*.

Xiaoqing Zheng, Jiehang Zeng, Yi Zhou, Cho-Jui Hsieh, Minhao Cheng, and Xuanjing Huang. 2020. Evaluating and enhancing the robustness of neural network-based dependency parsing models with adversarial examples. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6600–6610, Online. Association for Computational Linguistics.

Yi Zhou, Xiaoqing Zheng, Cho-Jui Hsieh, Kai-Wei Chang, and Xuanjing Huang. 2021. Defense against synonym substitution-based adversarial attacks via Dirichlet neighborhood ensemble. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5482–5492, Online. Association for Computational Linguistics.

Chen Zhu, Yu Cheng, Zhe Gan, Siqi Sun, Tom Goldstein, and Jingjing Liu. 2020. FreeLB: Enhanced adversarial training for natural language understanding. In *International Conference on Learning Representations*.

A Algorithm

Algorithm 1 Weighting Examples Towards Adversarial Robustness Algorithm

Input: $\mathcal{D}$:training dataset $\{(x_i, y_i)\}_{i=1}^{N}$, $\mathcal{D}^v$: adversarial validation set $\{(x_i^v, y_i^v)\}_{i=1}^{M}$, $\boldsymbol{\theta}$:model parameters, τ:learning rate, $\boldsymbol{\epsilon}$: weights for perturbation, $\boldsymbol{w}$: weights for updating
Output:$\boldsymbol{\theta}$:model weights

1: Initialize $\boldsymbol{\theta}$
2: **for** Iterative step $t = 1, ..., T$ **do**
3: **for** minibatch $\{x_i, y_i\}_{i=1}^{n} \subset \mathcal{D}$ **do**
4: // Initialize $\boldsymbol{\epsilon}$ and minibatch from $\mathcal{D}^v$
5: Sample minibatch $\{x_i^v, y_i^v\}_{i=1}^{m} \subset \mathcal{D}^v$
6: $\boldsymbol{\epsilon} \leftarrow \mathbf{0}$
7: // Calculate $\nabla \boldsymbol{\theta}_t'$ and update meta model
8: $g_{\boldsymbol{\theta}_t'} \leftarrow \nabla_{\boldsymbol{\theta}_t} \sum_{i=1}^{n} \epsilon_i L_{\boldsymbol{\theta}_t}(x_i, y_i)$
9: $\boldsymbol{\theta}_t' = \boldsymbol{\theta}_t - \tau g_{\boldsymbol{\theta}_t'}$
10: // Calculate $g_{\boldsymbol{\epsilon}}$ and weights $\boldsymbol{w}$
11: $g_{\boldsymbol{\epsilon}} \leftarrow \nabla_{\boldsymbol{\epsilon}} \frac{1}{m} \sum_{j=1}^{m} L_{\boldsymbol{\theta}_t'}(\boldsymbol{x}_j^v, y_j^v)$
12: $\tilde{\boldsymbol{w}} \leftarrow \max(-g_{\boldsymbol{\epsilon}}, \mathbf{0})$
13: $\boldsymbol{w} \leftarrow \frac{\tilde{\boldsymbol{w}}_j}{\sum_j \tilde{\boldsymbol{w}}_j + \delta(\boldsymbol{w}_j)}$
14: // Update model with reweighted examples
15: $\nabla \boldsymbol{\theta}_t \leftarrow \nabla_{\boldsymbol{\theta}_t} \sum_{i=1}^{n} w_i L_{\boldsymbol{\theta}_t}(x_i, y_i)$
16: $\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \tau \nabla \boldsymbol{\theta}_t$
17: **end for**
18: **end for**
19: **return** $\boldsymbol{\theta}$

In this section, we provide the algorithm for training process of METAR and describe the whole algorithm process in detail. We would construct an adversarial validation set first and then proceed to the next step of training. In case of METAR-D method, we reconstruct our adversarial validation set after every 1-2 epochs.

As shown in Algorithm 1, we sample a minibatch from the adversarial validation set and initialize weight perturbation $\boldsymbol{\epsilon}$ in line 5-6. In line 8-9, we calculate $\boldsymbol{\theta}_t'$ in order to obtain the gradient $g_{\boldsymbol{\epsilon}}$ of $\boldsymbol{\epsilon}$ by meta-learning algorithm in line 11. We obtain the relative importance $\tilde{\boldsymbol{w}}$ of samples by comparing the magnitude of $-g_{\boldsymbol{\epsilon}}$. Note that we use the opposite direction of $g_{\boldsymbol{\epsilon}}$ to evaluate the relative importance because we do a gradient descent operation in line 9. We then normalize this importance weight $\tilde{\boldsymbol{w}}$ to $\boldsymbol{w}$ and use $\boldsymbol{w}$ to weight training samples in the regular training. In general, the theoretical computational complexity of our algorithm is about three times greater than the regular training method.

B Implementation Details

Table B shows the implementation details about the hyper-parameters we used to train models. "Adversarial Learning Rate" is the parameter setting for standard adversarial training methods. "Proportion ρ for WETAR" means that there are 50% of samples are clean samples in validation batch in each training iteration. In order to make the guidance function of adversarial validation set more obvious, we use relatively larger adversarial validation batch.

Hyper-parameters	**SST-2**	**AGNEWS**	**MR**
Learning Rate	2×10^{-5}	2×10^{-5}	2×10^{-5}
Weight Decay	1×10^{-6}	1×10^{-6}	1×10^{-6}
Batch Size	32	32	8
Epochs	10	10	10
Adversarial Learning Rate	0.03	0.06	0.03
FreeLB Ascent Step	2	3	3
FreeLB++ Ascent Step	10	10	10
Proportion ρ for WETAR	50%	50%	50%
Validation Batch Size	256	256	64

Table 3: The training hyperparameters we selected to train models across three datasets in Table 1.

C Weight Distribution

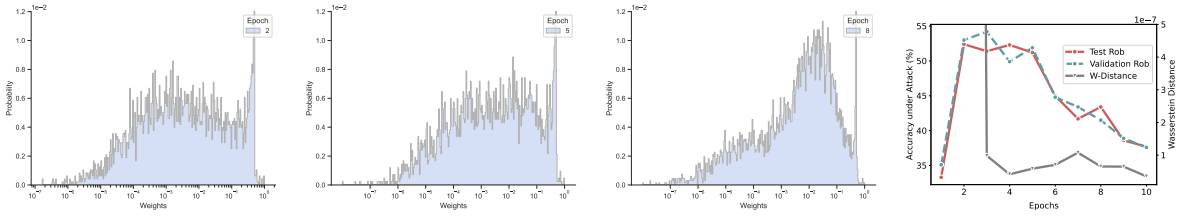
In the section, we first provide the experimental basis for our empirical model selection method with respect to the Wasserstein distance of weight distribution at every two epochs. We also provide detailed weight distributions in this section.

Figure 5 shows the weight distribution of WETAR-D on AGNEWS and SST-2 datasets. We can draw the similar conclusion that our empirical model selection method based on Wasserstein distance could select a relatively robust model.

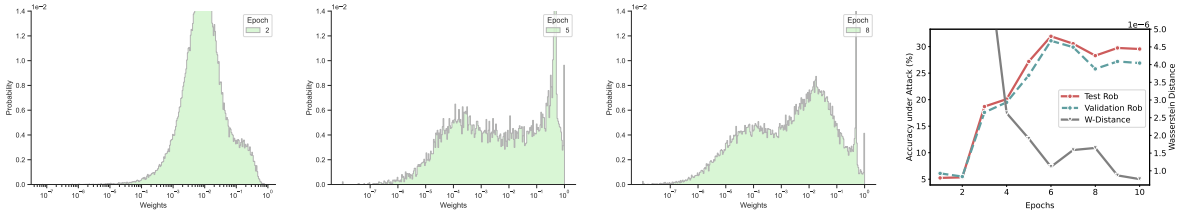
We provide more weight distributions in this section in addition to the above distributions. Figure 6 and 7 show the weight distribution provided by WETAR-D on AGNEWS dataset and SST-2 dataset respectively. Figure 10 and 9 show the weight distribution on MR dataset provided by WETAR-D and WETAR-S respectively.

D Visualization

Figure 8 shows t-SNE results of training examples in the SST-2 dataset. For each sub-figure, the dots are outputs of the last layer of the model for the corresponding epoch, and the relative importance of dots is calculated by the average weights.

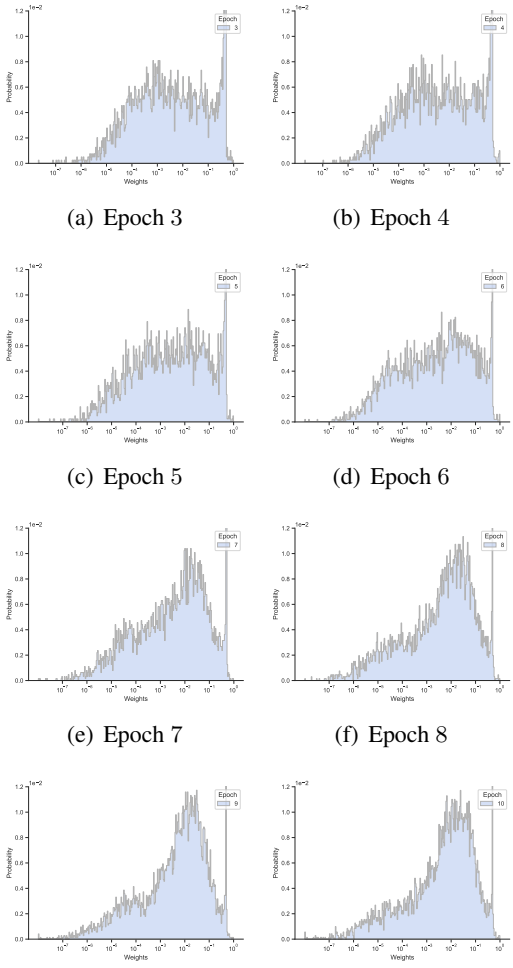


(a) Epoch 2 (b) Epoch 5 (c) Epoch 8 (d) WETAR-D



(e) Epoch 2 (f) Epoch 5 (g) Epoch 8 (h) WETAR-D

Figure 5: The weight distributions produced by WETAR-D on AGNEWS and SST-2 datasets. Sub-figures (a), (b), and (c) show the weight distributions produced by WETAR-D at epoch 2, 5, and 8 on AGNEWS respectively. Sub-figures (e), (f), and (g) give the same distributions produced by WETAR-D at epoch 2, 5, and 8 on SST-2. Sub-figures (d) and (h) plot the curves of accuracy under attack and the Wasserstein distance between two weight distributions at every two epochs respectively.



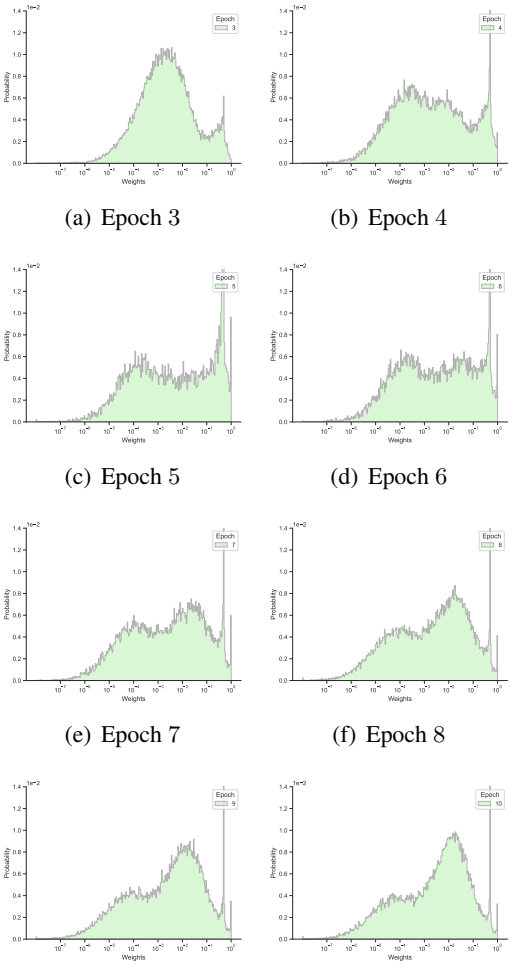
(a) Epoch 3 (b) Epoch 4

(c) Epoch 5 (d) Epoch 6

(e) Epoch 7 (f) Epoch 8

(g) Epoch 9 (h) Epoch 10

Figure 6: Weight distribution provided by WETAR-D on AGNEWS dataset.



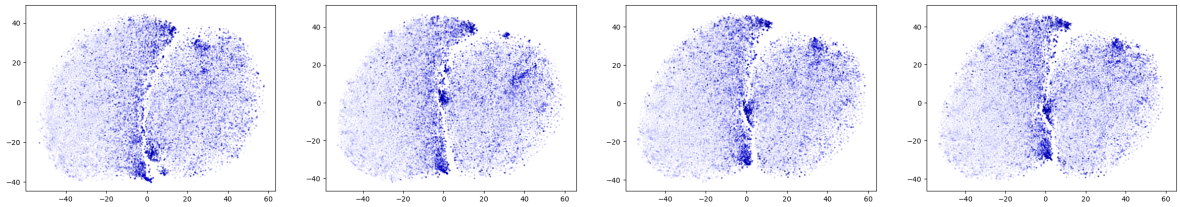
(a) Epoch 3 (b) Epoch 4

(c) Epoch 5 (d) Epoch 6

(e) Epoch 7 (f) Epoch 8

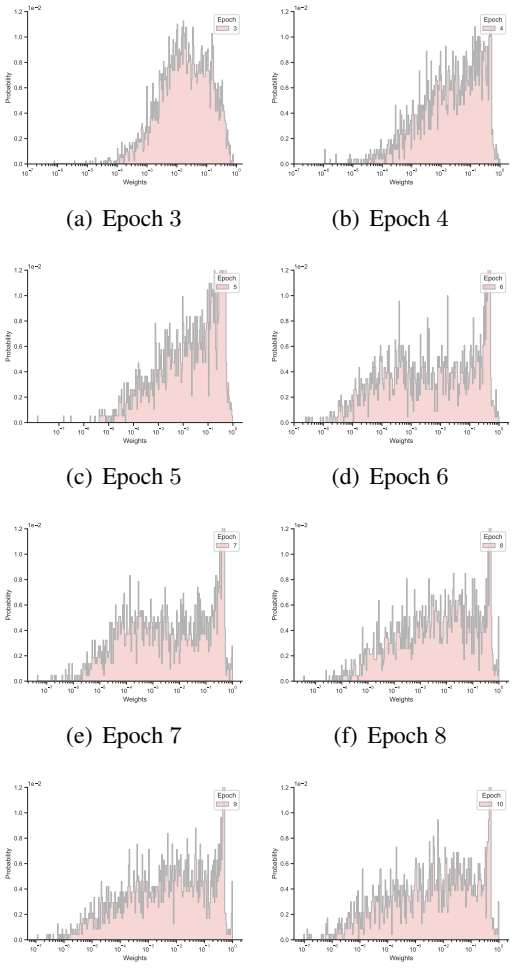
(g) Epoch 9 (h) Epoch 10

Figure 7: Weight distribution provided by WETAR-D on SST-2 dataset.



(a) Epoch 7 (b) Epoch 8 (c) Epoch 9 (d) Epoch 10

Figure 8: t-SNE visualization of the representations of SST-2 training examples produced by the model trained via WETAR.



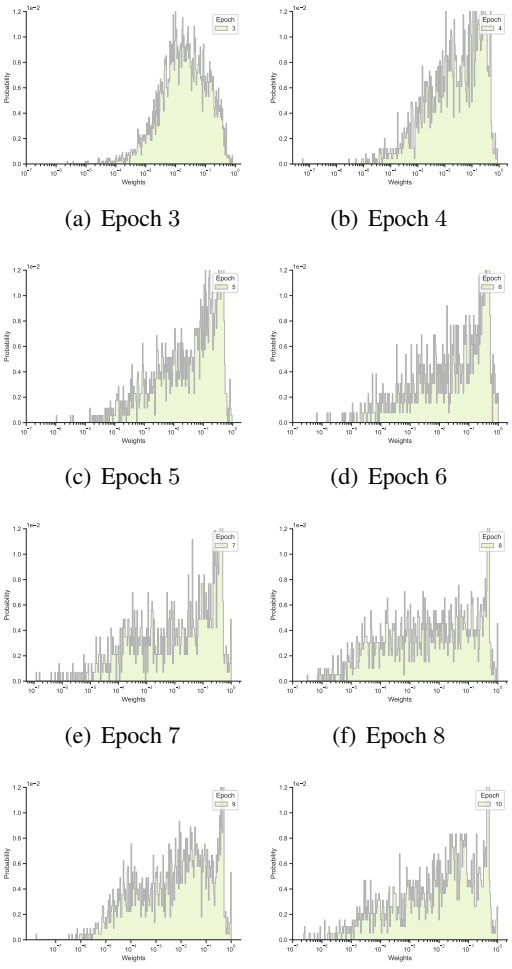
(a) Epoch 3 (b) Epoch 4

(c) Epoch 5 (d) Epoch 6

(e) Epoch 7 (f) Epoch 8

(g) Epoch 9 (h) Epoch 10

Figure 9: Weight distribution provided by WETAR-S on MR dataset.



(a) Epoch 3 (b) Epoch 4

(c) Epoch 5 (d) Epoch 6

(e) Epoch 7 (f) Epoch 8

(g) Epoch 9 (h) Epoch 10

Figure 10: Weight distribution provided by WETAR-D on MR dataset.