ADAPTERS MIXUP: Mixing Parameter-Efficient Adapters to Enhance the Adversarial Robustness of Fine-tuned Pre-trained Text Classifiers

Anonymous ACL submission

Abstract

Existing works show that augmenting the training data of pre-trained language models (PLMs) for classification tasks fine-tuned via parameter-efficient fine-tuning methods (PEFT) using both clean and adversarial examples can enhance their robustness under adversarial attacks. However, this adversarial training paradigm often leads to performance degradation on clean inputs and requires frequent re-training on the entire data to account for *new, unknown attacks*. To overcome these challenges while still harnessing the benefits of adversarial training and the efficiency of PEFT, this work proposes a novel approach, called ADPMIXUP, that combines two paradigms: *(1) fine-tuning through adapters* and *(2) adversarial augmentation via mixup* to dynamically leverage existing knowledge from a set of pre-known attacks for robust inference. Intuitively, ADPMIXUP fine-tunes PLMs with multiple adapters with both clean and pre-known adversarial examples and intelligently mixes them up in different ratios during prediction. Our experiments show ADPMIXUP achieves the best trade-off between training efficiency and robustness under both *pre-known and unknown* attacks, compared to existing baselines on five downstream tasks across six varied black-box attacks and 2 PLMs. All source code will be available.

1 Introduction

PEFT exemplified by adapter methods, offers a promising solution to mitigate fine-tuning costs for PLMs. PEFT involves injecting a small set of parameters into specific locations within a PLM, activating only these parameters while freezing the remainder during training. This approach significantly reduces the number of trainable parameters to as little as 0.1% of the original count, while maintaining competitive performance on downstream tasks (Wang et al., 2022). Adversarial training includes textual adversarial examples during training

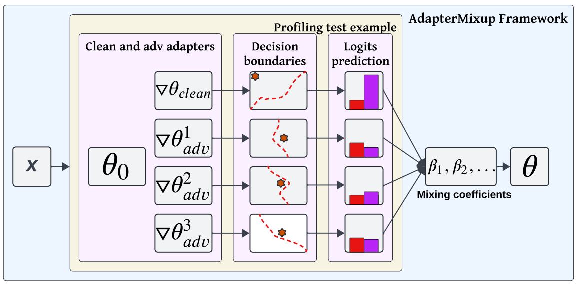

Figure 1: ADPMIXUP Framework: Final model θ is achieved by dynamically mixing the adapter weights across clean and adversarial with different coefficients $\beta_1, \beta_2, \ldots$. The *dash red lines* are the decision boundaries of different fine-tuning models, that when mixed in a certain way can result in robust inference.

to enhance adversarial robustness for neural network models, including PLMs (Goodfellow et al., 2015a; Miyato et al., 2018; Zhu et al., 2020). However, fine-tuning PLMs via this paradigm to be robust against various types of adversarial perturbations is computationally expensive due to the necessity of independently re-training the models with different perturbations to accommodate different types of attack methods. In addition, adversarial training often decreases performance on clean examples (Xu et al., 2021).

To further improve adversarial training, Miyato et al. (2018) introduces Mixup, which trains a model on virtual examples constructed via linear interpolation between two random examples from the training set and their labels. Mixup also helps to improve model robustness under a variety of mixing styles, including mixing between original examples, between original examples and their adversarial examples, and between only adversarial examples (Si et al., 2021). However, Mixup shares the same inefficiency with adversarial training in practice as we need to retrain entire models every time we need to accommodate new types of attacks. It is also unknown how Mixup can be efficiently applied to fine-tuning PLMs via PEFT. In fact, there is limited number of research addressing PLMs's generalization capabilities and adversarial robustness (Nguyen and Le, 2024) when fine-tuned via PEFT, not to mention that many of existing defense methods are not specifically designed for PEFT, highlighting a critical gap in the literature. These observations prompt a crucial question: *"How can we use PEFT with PLMs on downstream tasks that can achieve better trade-off among accuracy, adversarial robustness, and computational complexity and also withstand a variety of new, unknown attack methods?"* To answer this question, we seek to investigate how to incorporate adversarial data augmentation training to improve PLMs' adversarial robustness without sacrificing performance on clean examples, while making minimal changes to complex PLMs during fine-tuning to minimize computational overhead with PEFT.

To tackle this, this work presents a novel approach, called ADPMIXUP, that combines two key paradigms: *(1) fine-tuning through PEFT, often referred to as adapters* (Houlsby et al., 2019; Hu et al., 2022) and *(2) adversarial augmentation via mixup* (Miyato et al., 2018). Intuitively, ADPMIXUP fine-tunes PLMs with multiple adapters with both clean and pre-known adversarial examples and mix them up in different ratios for robust inference (Fig. 1). This new adapters mixup paradigm allows ADPMIXUP to work well in practice when the attack methods by which possible adversarial examples are generated are unknown.

Our contributions are summarized as follows.

1. Provide an analysis of the connections between data augmentation methods adversarial training, Mixup, and model augmentation methods including ModelSoup and PEFT via Adapters;
2. Propose ADPMIXUP that combines adversarial training, Mixup, and Adapters to achieve the best trade-off between training efficiency, and predictive performance under both clean and adversarial examples generated via pre-known and unknown attacks on five classification datasets;
3. ADPMIXUP also achieves the best trade-off in generalizability under both clean and adversarial examples, and superior efficient space and runtime complexity in practice.
4. ADPMIXUP also enables the profiling of potential adversarial examples by characterizing them into pre-known attacks, allowing more interpretable analysis of risk analysis in practice.

2 Related Work

2.1 Training with Data Augmentation

Let denote $f(\cdot;\theta)$ a PLM parameterized by θ, (x_i, y_i) an arbitrary clean input.

Adversarial Training Goodfellow et al. (2015a). Adversarial augmentation training optimizes θ on both clean and adversarial examples to improve f's adversarial robustness by minimizing the loss:

$$\alpha L(f(x;\theta), y) + (1-\alpha) \max_{\delta \in S} L(f(x+\delta;\theta), y), \quad (1)$$

where δ is the adversarial perturbation, α controls how much the loss L is updated towards the adversarial and clean version of the training input x and label y, and S is the set of allowed perturbations.

Mixup. Adversarial augmentation training helps enhance the adversarial robustness of NNs models. However, studies such as Xie et al. (2019) observe a consistent instability, often leading to a reduction in the trained models' performance on clean examples. This is a result of the substantial gap between clean and adversarial examples that can introduce non-smooth learning trajectories during training (Si et al., 2021). To address this, *Mixup* (Zhang et al., 2018a) was proposed as a data augmentation method via linear interpolation to tackle a model's sensitivity to adversarial examples, and its instability in adversarial training. Mixup trains a model on virtual examples constructed via linear interpolation between two random examples from the training set and their labels (Zhang et al., 2018b). While Mixup has been proposed as suitable for continuous data, its application to text data raises questions about its natural fit. Defining a process of "convex combination" between two texts is mathematically feasible, yet the resulting text may *lack grammatical correctness or semantic coherence*. This challenges Mixup's utility as a viable method for enhancing the robustness of PLMs. In practice, *Mixup also shares the same inefficiency with adversarial training as we need to fine-tune a PLM on entire datasets to accommodate new types of attacks*.

2.2 Training with Model Augmentation

Model Soup. Model Soup averages model weights of a pool of k models $\{f_1, f_2, \ldots, f_k\}$ to achieve better robustness without incurring runtime required to make k inference passes as seen in classical ensemble learning (Wortsman et al., 2022).

In principle, Model Soup is similar to Stochastic Weight Averaging (Izmailov et al., 2018) which averages model weights along an optimization trajectory. Particularly, given two model weights θ_1 and θ_2, Model Soup with a coefficient $\alpha \in [0, 1]$ results in a single model with parameters:

$$\theta_\alpha = \alpha\theta_1 + (1 - \alpha)\theta_2 \tag{2}$$

Adapter. Adapters or PEFT help fine-tuning PLMs on downstream tasks or with new domains efficiently (Houlsby et al., 2019; Hu et al., 2022). Some works such as (Pfeiffer et al., 2020) also propose to use not only one but also multiple adapters to further enhance the generalizability of the fine-tuned models on not one but multiple domains. Given f with a PLM parameter θ_0, fine-tuning f via adapters on two domains D_1 and D_2 results in two sufficiently small adapter weights $\nabla\theta_1$ and $\nabla\theta_2$, respectively. This corresponds to two distinct models $f(\cdot; \theta_0 + \nabla\theta_1)$ and $f(\cdot; \theta_0 + \nabla\theta_2)$ during inference. Since $\nabla\theta_1$ and $\nabla\theta_2$ are designed to be very small in size compared to θ_0, this approach *helps achieve competitive performance compared to fully fine-tuning all model parameters θ_0 with only a small fraction of the cost.*

3 Motivation

In this section, we demonstrate how data augmenting methods are linked to the model augmentation methods, which underpins the rationale for our ADPMIXUP framework.

3.1 Adversarial Training versus Mixup

Adversarial Training helps enhance the adversarial robustness of NNs by jointly optimizing θ on both clean and adversarial data following the Eq. (1). When $\alpha \leftarrow 1$, Eq. (1) converges to conventional training on only clean examples, resulting in $\theta \leftarrow \theta_{\text{clean}}$. When $\alpha \leftarrow 0$, it converges to adversarial training with only adversarial examples, resulting in $\theta \leftarrow \theta_{\text{adv}}$ (Madry et al., 2018).

Mixup. Mixup (Zhang et al., 2018a) is used to regularize NNs f to favor simple linear behavior in between training examples by training f on convex combinations of pairs of examples and their labels. Exploiting this property of Mixup, (Si et al., 2021) proposes to adapt Mixup to augment training examples by interpolating not only between clean but between clean and adversarial samples. Given two pairs of samples (x_i, y_i) and its adversarial sample (x_i^*, y_i^*), their Mixup interpolation results in:

$$\begin{aligned}(\overline{x}, \overline{y}) &= \text{Mixup}((x_i; y_i), (x_i^*; y_i^*)) \\ &= [\lambda x_i + (1-\lambda)x_i^*; \lambda y_i + (1-\lambda)y_i^*],\end{aligned} \tag{3}$$

where λ is the interpolation coefficient. When λ=1, the Mixup produces only clean samples, resulting in a trained model with parameter θ=θ_{clean}. When λ=0, the Mixup produces only adversarial examples, resulting in the trained model with parameter θ=θ_{adv}. From Eq. (1), Mixup with adversarial examples under $\lambda \leftarrow 0$ or $\lambda \leftarrow 1$ converges to adversarial training with $\alpha \leftarrow 0$ and $\alpha \leftarrow 1$, respectively.

3.2 ModelSoup on Adapter

Model Soup is used to averaging weights of multiple fine-tuned models improves generalization without increasing inference time (Wortsman et al., 2022). However, in Model Soup, when the model weights are substantially different, averaging them would result in conflicting or contradicting information acquired during *pre-training*, leading to poor performance. Model Soup's authors also advocate the selection of sub-models in decreasing order of their validation accuracy on the same task for optimal results, showing that the sub-models should sufficiently converge or, they are close in parameter space, especially for PLMs (Neyshabur et al., 2020).

To pursue a harmonized optimization trajectory, we want θ_1 and θ_2 to exhibit substantial similarity, differing only in a few parameters responsible for their expertise. This is the case of Adapters, as we can decompose θ_1=θ_0+$\nabla\theta_1$, θ_2=θ_0+$\nabla\theta_2$ where the sizes of $\nabla\theta_1$, $\nabla\theta_2$ are minimal compared to θ_1 or θ_2 (§2.2). Hence, this motivates us to adopt adapters to maximize the similarity in optimization trajectories between two sub-models, enabling the training of a merged model that is more competitive. Moreover, merging adapters are also more efficient, only requiring fine-tuning a small set of additional parameters $\nabla\theta_1$, $\nabla\theta_2$ and not the whole θ_1 and θ_2. Therefore, to get a single language model that generalizes well on the two tasks following parametrized model:

$$f(\cdot; \theta_0 + [\beta\nabla\theta_1 + (1-\beta)\nabla\theta_2]) \tag{4}$$

where β is the weighting factor when averaging the adapters. If $\beta = 1$, the *Model Soup on Adapters* boils down to the θ_1 mode and conversely $\beta = 0$ corresponds to the θ_2 mode.

4 AdpMixup: Mixup of Adapters with Adversarial Training

From § 3, we learned that the mixed model achieved by Model Soup on the Adapters can achieve performance close to the model training with Mixup data augmentation. Therefore, instead of doing Mixup on text samples which is not intuitive in practice, AdpMixup allows us to do Mixup on the model weight but still preserve the effectiveness of the data augmentation method.

Let's define two adapters $\theta_{clean}, \theta_{adv}$ trained on clean and adversarial data, respectively. We have model in *clean mode* $f(\cdot;\theta_0{+}\nabla\theta_{clean})$, and $f(\cdot;\theta_0{+}\nabla\theta_{\text{adv}})$ is the model in *adversarial mode*. Prediction after mixing the two adapters via Mixup can then be formulated as:

$$f(\cdot;\theta_0 + [\beta\nabla\theta_{clean} + (1-\beta)\nabla\theta_{\text{adv}}]), \quad (5)$$

where β is the Mixup coefficient. When $\beta{=}1$, AdpMixup boils down to the *clean mode* and conversely $\beta{=}0$ corresponds to the *adversarial mode*.

4.1 Choosing β dynamically

Given a PLM θ_0, clean adapter $\nabla\theta_{clean}$ and adversarial adapter $\nabla\theta_{adv}$, we want to find the optimal β for every sample during inference based on entropy to measure uncertainty.

Specifically, given $P_{clean}(x)$ is the probability of prediction of clean model θ_{clean} on example x. Then the entropy H measures the expected information content of the prediction $P_{clean}(x)$ is computed following:

$$H(P_{clean}(x)) = -\sum_{p(x)} p(x)\log(p(x)), \quad (6)$$

where $p(x)$ is probability prediction of $P_{clean}(x)$ over classification label. Since the prediction $P_{clean}(x)$ will be close to the uniform distribution on adversarial example. Therefore, the entropy of the prediction of the clean model should be high on adversarial examples. As a consequence, if the test samples are close to the clean set, β should be close to 1, and vice versa if the test samples are close to the adversarial set, β should be close to 0 (Eq. 5).

4.2 Pre-knowing one adversarial attack

Measure how much clean adapter contributed to mix model. Let denote $H(P_{clean}(x_1))$, $H(P_{clean}(x_2))$, $\ldots$, $H(P_{clean}(x_k))$ is the set of entropy of clean prediction over 100 samples train clean dataset. $\max_{\text{clean}}$ and $\min_{\text{clean}}$ are the maximum and minimum entropy, respectively. With test example $x_i (i \in [0,k])$, we estimate the contribution of a clean adapter by the maximum normalization following:

$$\alpha_i^{clean} = \frac{max_{clean} - H(P_{clean}(x_i))}{max_{clean} - min_{clean}}. \quad (7)$$

and the mixed model is used to predict x_i is computed following:

$$\theta_i = \theta_0 + [\alpha_i^{clean}\nabla\theta_{clean} + (1-\alpha_i^{clean})\nabla\theta_{adv}] \quad (8)$$

Intuitively, if example x_i is a clean example, $H(P_{clean}(x_i))$ will be low, α_i^{clean} will be high, and the mixed model θ_i will close with the clean model. On the opposite, if example x_i is an adversarial example, $H(P_{clean}(x_i))$ will be high, α_i^{clean} will be low, and the mixed model θ_i will close with the adversarial model. To summarize, α_i^{clean} controls how much a clean adapter contributes to the mixed model.

Measure how much adversarial adapter contributed to mix model. Similarly, we compute α_i^{adv} by using minimum normalization with the set of entropy of the adversarial model over 100 adversarial training samples $H(P_{adv}(x_1))$, $H(P_{adv}(x_2))$, $\ldots$, $H(P_{adv}(x_k))$. α_i^{adv} controls how much an adversarial adapter contributes to the mixed model. Then the mixed model is used to predict x_i is computed following:

$$\theta_i = \theta_0 + [\alpha_i^{adv}\nabla\theta_{clean} + (1-\alpha_i^{adv})\nabla\theta_{adv}] \quad (9)$$

In summary, let denote the mixing coefficient $\beta^i = (\alpha_i^{clean} + \alpha_i^{adv})/2$, and average the RHS terms in the Eq. 8 and 9:

$$\theta_i = \theta_0 + [\beta^i\nabla\theta_{clean} + (1-\beta^i)\nabla\theta_{adv}] \quad (10)$$

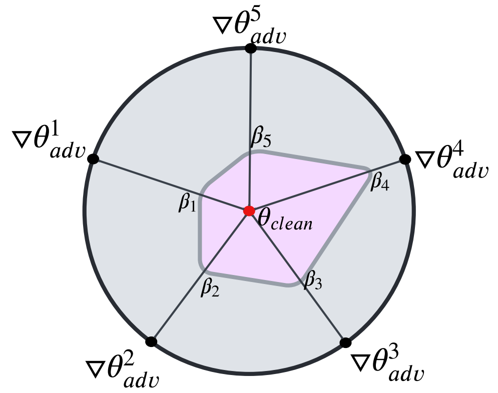

Figure 2: By choosing the coefficients β dynamically, AdpMixup allows us to profile the regions of combination weight. θ_0 represents the pre-trained weight of the language model, while the gray area illustrates all possible combinations between clean and adversarial adapters. The pink area denotes the potential robust combinations of adapter weights.

4.3 Pre-knowing *m* adversarial attack (m>1)

In scenarios where m adversarial attacks are identified, we can construct m pairs of clean and adversarial adapters for each example during inference. If $m = 2$, for every example on the evaluation set, we have two pairs of ($\nabla\theta_{clean}$, $\nabla\theta^1_{adv}$) and ($\nabla\theta_{clean}$, $\nabla\theta^2_{adv}$) with 2 coefficient (β_1, β_2). Specifically, for every sample $x_i (i \in [0, k])$, we have two mixed model which are formulated as:

$$\theta^1_i = \theta_0 + [\beta^i_1 \nabla\theta_{clean} + (1 - \beta^i_1)\nabla\theta^1_{adv}] \quad (11)$$

$$\theta^2_i = \theta_0 + [\beta^i_2 \nabla\theta_{clean} + (1 - \beta^i_2)\nabla\theta^2_{adv}] \quad (12)$$

The final mixed model for sample x_i, utilizing one clean adapter and two known adversarial adapters, is computed as follows:

$$\theta_i = (\theta^1_i + \theta^2_i)/2 = \theta_0 + \frac{(\beta^i_1 + \beta^i_2)}{2}\nabla\theta_{clean} + \frac{(1 - \beta^i_1)}{2}\nabla\theta^1_{adv} + \frac{(1 - \beta^i_2)}{2}\nabla\theta^2_{adv} \quad (13)$$

Generalizing for m>2, for every sample x_i, the final mixed model is computed as $\frac{\sum_{l=1}^{l=m} \theta^l_i}{m}$, where θ^l_i represents the prediction from the l-th adversarial adapter for sample x_i.

$$\theta_i = \theta_0 + \frac{\sum_{l=1}^{l=m} \beta^i_l}{m}\nabla\theta_{clean} + \frac{\sum_{l=1}^{l=m}(1 - \beta^i_l)}{m}\nabla\theta^l_{adv} \quad (14)$$

As a results, ADPMIXUP utilizes the entropy of model predictions as a metric to quantify the contribution of each adapter, potentially impacting the final mixed model for every new incoming sample. The visualization of the profiling weight for each incoming input is depicted in Fig. 2.

5 Experiment Set-up

Datasets and Models. We evaluate the effectiveness of ADPMIXUP on the GLUE benchmark dataset (Wang et al., 2019) across 5 tasks. We present the average clean and adversarial accuracy on the test set. We evaluate our algorithm on the BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019) because they share the same architecture with the latter models and at the same time standard for benchmarking in existing works (Si et al., 2021). We use the popular Houlsby adapter (Houlsby et al., 2019) as the PEFT method for efficient fine-tuning. We refer the readers to § A.1 and § A.2 (Appendix) for details of the benchmark datasets and the hyper-parameter configurations for fine-tuning our models, respectively.

Victim Models and Attack Methods. Our experimentation involves two Victim Models, namely BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019). We employ 4 different types of word-based text attackers, namely TextFooler (TF) (Jin et al., 2020), PW (Ren et al., 2019), BAE (Garg and Ramakrishnan, 2020), PS (Zang et al., 2020) and 2 types of character-based text attackers, namely DeepWordBug (DW) (Gao et al., 2018) and TextBugger (TB) (Li et al., 2019). They all observed superior effectiveness in attacking state-of-the-art PLMs while preserving as much as possible the original semantic meanings. Notably, all the attack algorithms are black-box attackers–i.e., they can query the target models' predictions but their parameters or gradients, making our evaluation practical. We refer the readers to A.3 for details of setting up the attackers.

Baselines. We compare ADPMIXUP with several baselines as follows.

- ***Base Models***: θ_{clean} (denoted as ***CleanOnly***), θ_{adv} (denoted as ***AdvOnly***) are two models trained with only clean examples and with only adversarial examples, respectively.
- ***AdvTrain*** (Miyato et al., 2018) where we train a single model on the augmentation of clean and adversarial data.
- ***ModelSoup*** (Wortsman et al., 2022) where we average the weights of whole models independently trained on clean and adversarial data.
- ***AdapterSoup*** (Chronopoulou et al., 2023) where we average the weights of adapters independently trained on clean and adversarial data.

6 Results

6.1 Defend Against Pre-Known Attacks

Table 1 shows results when the attackers are known in advance. ADPMIXUP outweighs augmentation methods in both data and model space in terms of averaged performance on clean and adversarial inputs across all types of attacks. Unlike adversarial training, ADPMIXUP's performance remains more or less the same with models trained with only clean examples. Although its performance under attacks was still below the model trained on *only* adversarial examples, it achieves *the best trade-off between with and without attacks* across all settings. Appendix A.4 provides more details.

Analysis #1: Mixing two whole, large independent clean and adversarial models results in a signifi-

	Methods	RoBERTa			BERT		
		Clean	Adv	Avg	Clean	Adv	Avg
Word-based	*CleanOnly*	91.7	49.7	70.7	84.0	50.0	67.0
	AdvOnly	55.7	69.8	62.8	61.3	69.2	65.3
	AdvTrain	89.8	65.6	77.7	81.8	61.1	71.5
	ModelSoup	77.1	59.8	68.5	70.0	54.4	62.2
	AdapterSoup	90.6	66.7	78.7	82.4	64.6	73.5
	AdpMixup	**91.6**	**71.4**	**81.5**	**83.2**	**66.4**	**74.8**
Character-based	*CleanOnly*	91.7	59.3	75.5	84.0	50.6	67.3
	AdvOnly	53.1	78.7	65.9	50.0	73.2	61.6
	AdvTrain	89.2	71.6	80.4	82.3	66.9	74.6
	ModelSoup	69.2	64.8	67.0	67.4	57.4	62.4
	AdapterSoup	90.4	72.9	81.7	82.3	70.2	76.3
	AdpMixup	**91.8**	**76.6**	**84.2**	**83.1**	**71.0**	**77.1**

Table 1: Average model performance over 5 datasets of independent clean and adversarial training, traditional adversarial training with RoBERTa, BERT under 6 different types of text adversarial attack. ***Bold**: the best average under clean and adversarial examples.*

Methods	RoBERTa				BERT			
	TB→DW		DW→TB		TB→DW		DW→TB	
	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv
CleanOnly	91.7	58.5	91.7	60.2	84.0	46.0	84.0	55.0
AdvOnly	53.5	67.2	52.7	71.0	42.6	72.3	57.4	69.6
AdvTrain	89.0	63.9	89.4	67.6	82.2	66.3	82.4	66.9
ModelSoup	68.9	53.5	69.5	54.9	66.9	56.1	67.9	53.3
AdapterSoup	89.2	67.6	90.5	70.7	81.2	70.4	82.5	68.9
AdpMixup	**91.7**	**68.0**	**91.8**	**71.3**	**82.5**	**71.1**	**82.8**	**69.4**

Table 2: Cross-attack evaluation between character-based TextBugger (TB) and DeepWordBug (DW). ***Bold**: the best average under clean and adversarial examples.*

cant decline in both generalization and adversarial robustness. This could happen because such independent models trained on datasets of different distributions may not converge to the same optimal trajectory. Thus, when combining them, the final mixed *ModelSoup* model will not represent the optimal solution, as also shown in (Wortsman et al., 2022). This confirms our analysis in §3.2.

Analysis #2: AdpMixup ***achieves better results on RoBERTa compared with BERT.*** AdpMixup exhibits a larger decline in both clean and adversarial robustness on BERT compared to RoBERTa. This discrepancy may be attributed to the size of the adapter, which is 64 for RoBERTa, considerably smaller than the 256 in BERT. Consequently, RoBERTa allocates a smaller portion of weights across clean and adversarial classifiers compared to BERT. This limited weight sharing enables RoBERTa to achieve competitive performance compared to ensemble learning, as discussed in § 3.2.

	Pre-Known Atk	TF		BA		PW		PS	
	Target Atk	PS	PW	PS	PW	TF	BA	TF	BA
RoBERTa	*CleanOnly*	50.3	56.1	50.3	56.1	46.7	45.4	46.7	45.4
	AdvOnly	65.5	68.2	64.4	66.4	66.8	69.0	62.8	63.5
	AdvTrain	60.2	62.8	59.7	61.2	63.3	62.7	57.7	55.9
	ModelSoup	56.0	56.2	54.7	55.8	53.5	52.5	50.8	50.6
	AdapterSoup	63.7	65.7	63.6	65.6	65.5	69.5	61.5	62.5
	AdpMixup	**65.1**	**67.2**	**64.5**	**67.1**	**66.4**	**70.4**	**62.0**	**63.5**
BERT	*CleanOnly*	51.3	51.1	51.3	51.1	53.5	45.8	53.5	45.8
	AdvOnly	63.6	63.5	57.7	56.4	67.6	65.8	58.1	57.2
	AdvTrain	54.8	58.7	52.5	51.9	62.1	53.3	56.5	51.4
	ModelSoup	51.1	51.4	45.6	43.9	53.0	50.2	47.4	45.6
	AdapterSoup	59.3	62.0	55.4	54.3	64.6	62.9	57.1	55.9
	AdpMixup	**61.3**	**63.6**	**57.1**	**55.5**	**65.1**	**63.9**	**58.1**	**57.6**

Table 3: Cross attack evaluation among word-based methods. ***Bold**: the best average under clean and adversarial examples.*

6.2 Defend with m=1 Pre-Known Attack

In this scenario, we train the adapter on one adversarial dataset which is generated by one type of adversarial attack, and then evaluate its performance on adversarial datasets which are generated by different adversarial attacking algorithms.

Intra-type Settings. Table 2 and 3 present the average clean and adversarial robustness scores between character-based, and word-based across 5 downstream tasks. Due to computational limitations, in Table 3, for each word-based attack method, we randomly selected two word-based methods as the target attacks. Overall, AdpMixup achieves the best trade-off performance with and without attacks in utilizing the pre-known adversarial knowledge of one attacker to defend another unknown one. We refer readers to Appendix A.5 for detailed results.

Inter-type Settings. Table 4 shows the model performance when trained on character-based adversarial datasets and evaluated on word-based adversarial datasets, and vice versa. Overall, injecting knowledge of adapters learned from character adversarial perturbations makes better improvement performance on word-based adversarial examples compared to knowledge learned from word-based.

6.3 Defend with m>1 Pre-Known Attacks

Table 5 shows the cross-attack evaluations when increasing the number of known adversarial attacks.

Analysis #1: AdpMixup ***effectively utilizes pre-known attacks to defend against unknown ones.*** Increasing the number of pre-known attacks m from 1 to 3 leads to an improvement in robust-

	Attacker	Charac→Word					Word→Charac		
		Clean	TF	BAE	PS	PW	Clean	DW	TB
RoBERTa	*CleanOnly*	91.7	46.8	45.6	50.3	56.1	91.7	58.5	60.0
	AdvOnly	53.1	62.0	58.2	60.8	64.7	55.7	64.9	66.4
	AdvTrain	89.2	56.2	51.8	56.1	65.4	89.8	62.4	62.8
	ModelSoup	69.2	52.8	43.5	46.1	47.4	81.6	57.2	58.0
	AdapterSoup	91.4	59.8	60.1	59.6	64.4	90.6	63.1	64.2
	ADPMIXUP	**91.8**	**61.6**	**61.5**	**60.7**	**64.4**	**91.6**	**64.5**	**66.1**
BERT	*CleanOnly*	84.0	53.5	45.8	57.4	51.1	84.0	46.0	55.0
	AdvOnly	50.0	63.2	65.3	65.7	65.5	61.3	63.9	63.7
	AdvTrain	82.3	56.8	52.7	56.3	59.9	81.8	57.8	60.8
	ModelSoup	67.4	52.0	49.0	47.2	49.6	70.0	47.9	51.0
	AdapterSoup	82.3	61.9	61.5	63.6	62.1	82.4	59.8	61.0
	ADPMIXUP	**83.1**	**62.8**	**62.0**	**64.2**	**65.0**	**83.2**	**63.0**	**63.1**

Table 4: Average Cross Pre-Known Character and Pre-Known Word attack. *In **bold** means best average performance with and without attack.*

	Attacker	m=1		m=2		m=3	
		Clean	Adv	Clean	Adv	Clean	Adv
RoBERTa	*CleanOnly*	91.7	49.7	91.7	49.7	91.7	49.7
	AdvOnly	55.7	65.8	55.8	66.8↑	55.1	68.6↑
	AdvTrain	89.9	60.4	90.2	61.4↑	86.1	64.0↑
	ModelSoup	81.6	53.8	67.5	49.7↓	60.5	46.4↓
	AdapterSoup	90.6	64.7	87.3	62.1↓	76.4	59.8↓
	ADPMIXUP	**91.6**	**65.7**	**89.6**	**68.0**↑	**91.6**	**69.1**↑
BERT	*CleanOnly*	84.0	49.9	84.0	49.9	84.0	49.9
	AdvOnly	61.1	61.4	41.5	64.0↑	44.3	65.4↑
	AdvTrain	81.4	55.0	80.0	57.2↑	77.8	59.3↑
	ModelSoup	70.8	48.5	63.5	44.6↓	52.9	42.6↓
	AdapterSoup	82.4	59.1	76.5	56.5↓	68.9	52.9↓
	ADPMIXUP	**83.3**	**60.3**	**80.3**	**60.8**↑	**83.6**	**63.5**↑

Table 5: Cross Attack Evaluation between Word-based methods when knowing more than one adversarial attack. *In **bold** means best average performance with and without attack.* ↑/↓ denotes the increase/decrease from preceding m pre-known attacks.

ness for RoBERTa under attacks from 65.8% with $m = 1$ to 66.8% and 68.6% with m is equal to 2 and 3, respectively. In addition, ADPMIXUP demonstrates sustained competitive performance on clean data while benefiting from enhanced performance on adversarial datasets when subjected to an increased number of text adversarial attacks. This resilience may be attributed to ADPMIXUP strategy of selecting different weight configurations for the clean and adversarial adapters, effectively harnessing insights from adversarial adapters to boost overall model performance.

Analysis #2: Model generalization decreases when increasing the number of known m attacks. When m increases from 1 to 3, *AdvOnly*, *AdvTrain*, *ModelSoup*, *AdapterSoup* show a significant drop in model generalization on both RoBERTa and

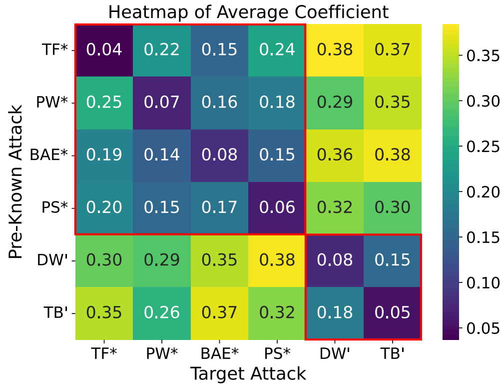

Figure 3: Average coefficient β of ADPMIXUP with $m = 1$ pre-known attack during inference on 100 test examples with RoBERTa against different attack methods. The lower the score, the more the adversarial adapter weight contributes to the mixed models. * and ' denote word-based and character-based attacks, respectively. Red rectangles denote attacks of the same type (word or character-based).

BERT (Table 5). This may stem from adversarial training inducing a shift in data distribution. Adversarial examples often deviate from the statistical distribution of clean data. Consequently, the training process might prioritize learning features and patterns specific to adversarial examples, diverging from the underlying data distribution of clean samples (Goodfellow et al., 2015b).

7 Discussion

Flexibility. Compared to the baselines, ADPMIXUP is able to leverage recent state-of-the-art PEFT methods with superior performance compared to Adapters (Houlsby et al., 2019), such as LoRA (Hu et al., 2022), AdaMix (Wang et al., 2022). Consequently, ADPMIXUP exhibits modular properties, enabling the defense against new types of adversarial attacks by conveniently training a new adapter corresponding to the specific new attack and subsequently merging them.

Profiling adversarial examples via analyzing β. Fig. 3 shows heatmap of average mixing coefficient β with m=1 pre-known attack. For every pre-known attack, we use ADPMIXUP to compute the set of mixing coefficient β to be used during inference on 100 samples generated from 6 types of target attack. Overall, the weights from word-based adversarial adapters contribute *more* to the final mixed model than the those from character-based adversarial adapters when the target attack is word-based. Similar observations can be made

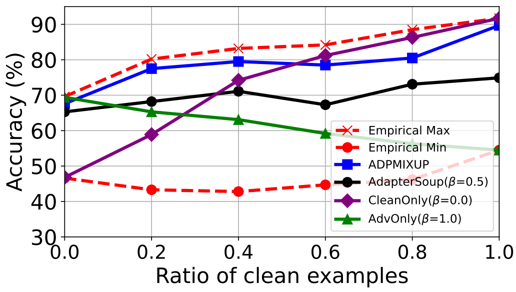

Figure 4: Average model accuracy (clean and adversarial) across 5 domain tasks under m=1 pre-known attack method at various ratios of clean examples.

Method	Notation	Training	Space	Inference
CleanOnly	$f(x,\theta)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$
AdvOnly	$f(x,\theta')$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$
ModelSoup	$f(x,\cup_{i=1}^{m}\theta_i)$	$\mathcal{O}(m)$	$\mathcal{O}(m)$	$\mathcal{O}(1)$
AdvTrain	$f(x,\cup_{i=1}^{m}\theta'_i)$	$\mathcal{O}(m)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$
AdapterSoup	$f(x,\theta_0\cup_{i=1}^{m}\nabla\theta_i)$	$\sim\mathcal{O}(1)$	$\sim\mathcal{O}(1)$	$\sim\mathcal{O}(1)$
ADPMIXUP	$f(x,\theta_0\cup_{i=1}^{n}\nabla\theta_i)$	$\sim\mathcal{O}(\mathbf{1})$	$\sim\mathcal{O}(\mathbf{1})$	$\mathcal{O}(\mathbf{m})$

Table 6: Theoretical complexity during training and inference on a single example. θ_i represents an individual model. m is the number of pre-known attacks. θ_0 is the pre-trained weight that is shared across models. $\nabla\theta_i$ is the adapter trained for task i-th.

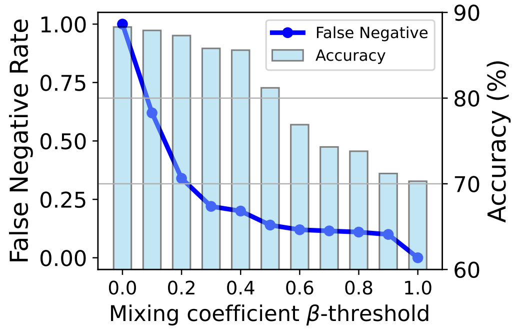

Figure 5: Trade-off between predictive accuracy (*bar*) and false negative rate in detecting adversarial examples (*line*) of ADPMIXUP with RoBERTa, assuming a *conservative* 15% ratio of adversarials out of 1K test inputs.

with character-based attacks. In other words, ADPMIXUP enables interpretable and intuitive characterization of unknown target attacks by attributing them to the suitable set of pre-known attacks.

Empirical Min and Empirical Max. Fig. 4 shows the average accuracy of ADPMIXUP across 5 tasks with m=1 pre-known attack under various ratios of clean examples. For each specific ratio of clean examples, we scan all the the coefficients $\beta\in[0,1]$ with step size of 0.1 to find ones that result in the best and the worst performance. AdapterSoup's performance (β=0.5 fixed) remains more stable than CleanOnly (β=0.0 fixed) and AdvOnly (β=1.0 fixed) as the ratio of clean examples increases. However, their performance are very far away from the empirical optimal performance. ADPMIXUP (dynamic β) automatically finds the suitable coefficient β, *achieving much closer performance to the empirical optimal* β. This further demonstrates the effectiveness of ADPMIXUP's intuition and design of using entropy to dynamically calculate the best set of β for robust inference.

Theoretical and empirical computational complexity. Table 6 shows that ADPMIXUP has near optimal complexities in terms of space and training time compared to the baselines due to marginal additional computations required to accommodate m adapters. During inference, ADPMIXUP requires $\mathcal{O}(\mathbf{m})$ complexity to calculate all the mixing coefficients β (Eq. 14). However, in practice, ADPMIXUP does *not always* need to use all m adapter heads if the input is clean, as there are usually much less number of adversarial examples. Thus, to further reduce the runtime during inference, we can set a threshold on calculated β on the clean adapter head to detect if an input is a potential adversarial example. With coefficient threshold β is set to 0.4, ADPMIXUP has a false negative rate on detecting adversarial examples of 0.25, and the accuracy of ADPMIXUP only drops 2.7% (88.3%→85.6%) (Fig. 5). This help reduces the runtime significantly as it only needs to use all m adapters *maximum* about 30% of the time. This makes ADPMIXUP's runtime complexity ($\sim\mathcal{O}(0.3m)$) closer or even better than $\mathcal{O}(1)$ when $m\leq 3$, making ADPMIXUP's overall complexity practical in real-life, given that the ratio of adversarial examples can be much less than 15% as used in Fig. 5.

8 Conclusion

This work provides a new framework for improving model generalization and robustness of PLMs under adversarial attacks by combining adversarial augmentation via Mixup and parameter-efficient fine-tuning via adapters. Our findings highlight the utility of adapters in empowering PLMs to achieve competitive performance in terms of generalization and robustness under both pre-known and unknown adversarial attacks with minimal additional computational complexity. Additionally, ADPMIXUP provides extra interpretability into profiling and analyzing potential adversarial examples in practice.

Limitation

Primarily, ADPMIXUP use weight average (Wortsman et al., 2022) to compute the weight of the final mixed model based on the mixing coefficient β. Consequently, future works could investigate the applicability of our findings to these alternative model merging approaches. Furthermore, our exploration focused solely on one BERT and RoBERTa on the natural language understanding tasks. As a result, a valuable avenue for future research would involve extending our analysis to encompass the emerging text generation tasks, particularly within the context of the current transformer-based language model like complex GPT-family models.

Broader Impacts and Ethics Statement

We expect no ethical concerns regarding the artifacts and real-life applications of this work.

References

Alexandra Chronopoulou, Matthew E Peters, Alexander Fraser, and Jesse Dodge. 2023. Adaptersoup: Weight averaging to improve generalization of pretrained language models. In *EACL*.

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

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

Siddhant Garg and Goutham Ramakrishnan. 2020. Bae: Bert-based adversarial examples for text classification. In *EMNLP*.

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

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

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *ICML*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *ICLR*.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. 2018. Averaging Weights Leads to Wider Optima and Better Generalization. In *Uncertainty in Artificial Intelligence*.

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. In *AAAI*.

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

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A Robustly Optimized BERT Pretraining approach. In *arXiv*.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards deep learning models resistant to adversarial attacks. In *ICLR*.

Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. 2018. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. In *IEEE transactions on pattern analysis and machine intelligence*.

Behnam Neyshabur, Hanie Sedghi, and Chiyuan Zhang. 2020. What is being transferred in transfer learning? In *NeurIPS*.

Tuc Nguyen and Thai Le. 2024. Generalizability of mixture of domain-specific adapters from the lens of signed weight directions and its application to effective model pruning. In *ACL*.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2020. Adapterfusion: Non-destructive task composition for transfer learning. In *arXiv*.

Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. 2019. Generating Natural Language Adversarial Examples through Probability Weighted Word Saliency. In *ACL*.

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 *ACL-IJCNLP*.

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*.

Yaqing Wang, Sahaj Agarwal, Subhabrata Mukherjee, Xiaodong Liu, Jing Gao, Ahmed Hassan Awadallah, and Jianfeng Gao. 2022. AdaMix: Mixture-of-adaptations for parameter-efficient model tuning. In *EMNLP*.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. 2022. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *ICML*.

Cihang Xie, Mingxing Tan, Boqing Gong, Jiang Wang, Alan Yuille, and Quoc V Le. 2019. Adversarial examples improve image recognition. In *arXiv*.

Han Xu, Xiaorui Liu, Yaxin Li, Anil Jain, and Jiliang Tang. 2021. To be robust or to be fair: Towards fairness in adversarial training. In *ICML*.

Yuan Zang, Fanchao Qi, Chenghao Yang, Zhiyuan Liu, Meng Zhang, Qun Liu, and Maosong Sun. 2020. Word-level textual adversarial attacking as combinatorial optimization. In *ACL*.

Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. 2018a. mixup: Beyond empirical risk minimization. In *ICLR*.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. 2018b. mixup: Beyond empirical risk minimization. In *ICLR*.

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 Appendix

A.1 Dataset statistics

Table 7 shows the number of instances for each dataset divided by training and test set and linguistic statistics over 5 evaluation datasets.

A.2 Training details

Tables 8, 9 show detailed hyper-parameter in our experiments.

A.3 TextAttack Configuration

For the *TextFooler* attack, we set the minimum embedding cosine similarity between a word and its synonyms as 0.85, and the minimum Universal Sentence Encoder (USE) similarity is 0.84. For the *BAE* word-based attack, we set the threshold for cosine similarity of USE as 0.94, and the window size is 15. For the *PS* we set the maximum number of iteration times to 10 and the population size to 60. For the *DeepWordBug*, we set the maximum difference in edit distance to a constant 30 for each sample. For the *TextBugger*, we set top-5 nearest neighbors in a context-aware word vector space, and the semantic similarity threshold for USE is set as 0.8.

A.4 Detailed Evaluation Results

Table 10, 11 show detailed results on the generalization and adversarial robustness of RoBERTa, BERT.

A.5 Detailed Cross Attack Evaluation

Average cross-attack evaluation. Table 12 and 13 show average cross-attack evaluation from word-base to character-base and vice versa.

From Word-based to Character-based attack. Table 14, 15 show detailed cross evaluation from word-based to character-based attack of RoBERTa and BERT.

From Character-based to Word-based attack. Table 16, 17, 18, 19 show detailed cross evaluation from character-based to word-based attack of RoBERTa and BERT.

Between Word-based attacks. Tables 20, 21, 22, 23 show detailed cross attack evaluations between Word-based methods on RoBERTa, BERT.

Between Character-based attacks. Tables 24, 25, 26 and 27 show detailed cross-attack evaluations between Word-based methods on RoBERTa, BERT.

A.6 Detailed Cross Attack Evaluation when know m>1 adversarial attacks

Tables from 28 to 31 show average model generalization and adversarial robustness across tasks when utilized in more than 1 adversarial attack.

A.7 Detail analysis on space and time complexity

Detail time and space analysis of different methods can be seen in Table 6.

Data Source	# Training Example	# Test Example	Average Document Length	Average Sentence Length	Average # Sentences per Document
MRPC	3,668	408	21.9	21.1	1.0
QNLI	104,743	5,463	18.2	18.0	1.0
RTE	2,490	277	26.2	18.1	1.4
SST	67,347	872	10.4	10.4	1.0
IMDB	22,500	2,500	233.8	21.6	10.8

Table 7: Number of instances for each dataset divided by training and test set and linguistic statistics.

Task	Learning rate	epoch	train batch size	evaluation batch size
$\text{BERT}_{\text{BASE}}$				
MRPC	2e-5	3	16	8
QNLI	3e-5	5	32	8
RTE	2e-5	3	16	8
SST2	2e-5	3	32	8
IMDB	5e-5	5	16	8
$\text{RoBERTa}_{\text{LARGE}}$				
MRPC	3e-5	10	32	16
QNLI	2e-4	5	32	16
RTE	3e-5	10	32	16
SST2	2e-5	5	32	16
IMDB	5e-5	5	32	16

Table 8: Hyperparameter configurations for fully finetuning on various tasks.

Task	Learning rate	epoch	batch size	warmup	weight decay	adapter size
$\text{BERT}_{\text{BASE}}$						
MRPC	4e-4	5	32	0.06	0.1	256
QNLI	4e-4	20	32	0.06	0.1	256
RTE	4e-4	5	32	0.06	0.1	256
SST2	4e-4	10	32	0.06	0.1	256
IMDB	4e-4	5	32	0.06	0.1	256
$\text{RoBERTa}_{\text{LARGE}}$						
MRPC	3e-4	5	64	0.6	0.1	64
QNLI	3e-4	20	64	0.6	0.1	64
RTE	3e-4	5	64	0.6	0.1	64
SST2	3e-4	10	64	0.6	0.1	64
IMDB	3e-4	5	64	0.6	0.1	64

Table 9: Hyperparameter configurations for adapter finetuning on various tasks.

Methods	MRPC		QNLI		RTE		SST2		IMDB	
	Clean	Attack	Clean	Attack	Clean	Attack	Clean	Attack	Clean	Attack
TextFooler										
CleanOnly	90.0	51.1	94.8	56.0	85.2	33.6	95.9	42.4	92.5	50.8
AdvOnly	68.4	68.6	49.4	66.4	52.7	76.2	51.0	59.0	51.1	76.9
AdvTrain	87.8	64.1	92.0	64.4	84.5	62.7	95.2	56.5	90.6	75.7
ModelSoup	87.3	53.9	67.0	65.0	85.1	56.8	94.8	51.5	89.2	67.7
ADPMIXUP	**89.9**	**68.6**	**94.4**	**66.7**	**85.0**	**76.6**	**96.3**	**58.9**	**92.6**	**76.8**
PW										
CleanOnly	90.0	60.3	94.8	63.2	85.2	35.4	95.9	67.7	92.5	54.1
AdvOnly	68.2	67.4	52.9	88.6	51.3	72.6	50.9	74.6	54.6	90.8
AdvTrain	87.3	66.8	92.2	79.1	82.9	63.7	95.3	74.2	89.1	68.4
ModelSoup	71.1	66.1	93.7	74.8	62.5	62.1	95.4	51.5	89.6	60.7
ADPMIXUP	**89.3**	**67.9**	**94.7**	**82.4**	**85.1**	**71.9**	**95.8**	**74.7**	**92.5**	**84.3**
BAE										
CleanOnly	90.0	55.5	94.8	50.9	85.2	39.4	95.9	29.3	92.5	52.9
AdvOnly	68.4	65.8	51.5	60.4	51.3	69.1	40.7	67.2	53.0	91.2
AdvTrain	88.7	60.1	94.1	58.0	79.1	62.0	94.2	55.2	91.3	70.2
ModelSoup	72.1	62.3	88.8	56.6	75.3	61.4	90.1	51.3	86.9	65.3
ADPMIXUP	**89.4**	**65.1**	**94.5**	**61.6**	**84.9**	**68.3**	**96.4**	**66.8**	**92.9**	**86.3**
PS										
CleanOnly	90.0	57.2	94.8	60.6	85.2	37.8	95.9	42.4	92.5	53.4
AdvOnly	68.4	63.2	77.2	67.2	52.3	70.4	49.7	69.0	50.2	90.9
AdvTrain	88.3	60.4	93.3	65.5	81.6	66.7	96.6	65.5	90.8	72.3
ModelSoup	70.1	56.2	87.2	60.9	58.5	62.2	85.1	41.5	82.4	67.2
ADPMIXUP	**89.3**	**62.9**	**94.6**	**68.2**	**85.8**	**69.5**	**96.6**	**68.9**	**92.8**	**81.3**
DeepWordBug										
CleanOnly	90.0	66.3	94.8	65.8	85.2	52.7	95.9	52.6	92.5	55.1
AdvOnly	66.4	76.3	55.4	73.1	51.6	73.6	39.9	79.9	50.0	93.2
AdvTrain	88.5	70.2	93.3	68.0	79.4	66.7	93.9	73.5	91.7	78.6
ModelSoup	68.4	70.0	61.0	64.8	55.6	67.7	84.6	62.3	78.0	69.6
ADPMIXUP	**90.6**	**75.9**	**94.7**	**71.3**	**84.9**	**71.9**	**96.5**	**78.9**	**92.4**	**84.2**
TextBugger										
CleanOnly	90.0	65.8	94.8	65.5	85.2	47.3	95.9	60.3	92.5	61.0
AdvOnly	62.5	79.2	50.2	79.1	52.7	71.9	50.9	71.4	51.2	89.0
AdvTrain	88.2	73.2	94.5	72.2	80.9	67.3	90.7	69.6	90.9	76.3
ModelSoup	77.2	55.7	56.4	73.2	61.0	58.9	75.2	55.0	74.6	70.2
ADPMIXUP	**89.7**	**77.5**	**95.2**	**78.6**	**85.0**	**70.9**	**96.3**	**70.9**	**92.4**	**85.4**

Table 10: Model performance of independent clean and adversarial training, traditional adversarial training with RoBERTa under TextFooler, and PW textual attack. In **bold** means better performance compared to Adversarial training. In red means performance is worse than Adversarial training.

Methods	MRPC		QNLI		RTE		SST2		IMDB	
	Clean	Attack	Clean	Attack	Clean	Attack	Clean	Attack	Clean	Attack
TextFooler										
CleanOnly	83.3	64.8	90.5	62.9	65.0	35.1	92.5	52.5	88.9	52.0
AdvOnly	35.6	66.9	75.3	88.9	46.9	58.7	23.6	68.6	60.1	87.3
AdvTrain	78.7	68.9	86.8	73.2	62.1	46.6	92.5	67.9	86.7	55.5
ModelSoup	68.5	57.9	76.6	62.4	54.5	40.9	83.2	55.9	68.9	59.4
ADPMIXUP	**82.1**	**68.0**	**89.6**	**84.2**	**64.1**	**55.3**	**92.3**	**67.9**	**87.9**	**83.2**
PW										
CleanOnly	83.3	60.5	90.5	59.8	65.0	39.5	92.5	40.2	88.9	55.6
AdvOnly	54.7	74.4	76.6	93.2	46.2	56.7	19.3	82.6	58.5	84.3
AdvTrain	78.9	70.9	85.3	73.3	59.9	50.4	90.5	72.3	87.9	65.6
ModelSoup	65.4	68.3	79.8	73.2	55.3	51.6	81.7	63.4	68.4	57.2
ADPMIXUP	**81.2**	**72.0**	**88.3**	**84.5**	**63.6**	**55.3**	**91.7**	**77.1**	**88.2**	**73.6**
BAE										
CleanOnly	83.3	60.8	90.5	53.1	65.0	37.9	92.5	27.7	88.9	49.6
AdvOnly	78.7	65.8	87.6	59.5	59.6	56.7	88.0	35.5	55.6	81.4
AdvTrain	79.7	66.4	90.2	56.8	63.2	53.6	91.7	35.1	88.7	60.1
ModelSoup	71.7	60.1	68.8	40.1	53.1	40.2	86.4	31.3	53.6	50.8
ADPMIXUP	**82.0**	**67.3**	**90.2**	**58.2**	**64.6**	**55.1**	**92.1**	**38.5**	**88.6**	**72.8**
PS										
CleanOnly	83.3	56.6	90.5	64.9	65.0	33.9	92.5	40.4	88.9	51.3
AdvOnly	76.5	78.2	79.2	64.4	60.6	44.8	86.4	57.1	55.8	78.9
AdvTrain	80.4	71.9	89.5	67.4	63.5	41.4	93.0	59.3	85.8	65.2
ModelSoup	71.0	68.6	83.3	59.4	52.7	37.2	85.6	48.7	72.3	61.3
ADPMIXUP	**81.3**	**75.2**	**90.4**	**65.3**	**64.7**	**43.5**	**92.1**	**57.1**	**87.8**	**73.5**
DeepWordBug										
CleanOnly	83.3	51.3	90.5	52.5	65.0	33.6	92.5	40.3	88.9	52.2
AdvOnly	75.2	75.4	50.8	71.8	54.5	66.4	47.4	67.0	58.9	75.3
AdvTrain	81.6	67.7	90.6	62.9	61.4	56.3	91.0	63.5	87.2	71.1
ModelSoup	68.9	53.0	76.9	53.0	54.5	58.4	86.4	48.3	52.6	60.9
ADPMIXUP	**82.4**	**72.5**	89.3	**68.2**	**63.1**	**64.3**	**91.4**	**66.0**	**88.0**	**74.1**
TextBugger										
CleanOnly	83.3	66.0	90.5	62.0	65.0	37.8	92.5	50.9	88.9	59.3
AdvOnly	38.7	76.0	52.1	69.0	45.1	65.7	23.2	76.6	53.9	88.8
AdvTrain	80.6	73.3	90.4	67.8	57.8	54.2	93.5	70.0	88.7	82.3
ModelSoup	70.3	66.0	81.5	59.4	54.5	55.3	74.9	46.6	53.3	73.2
ADPMIXUP	**82.0**	**74.7**	**90.1**	**68.0**	**63.5**	**63.3**	**92.6**	**74.1**	**88.5**	**84.4**

Table 11: Model performance of independent clean and adversarial training, traditional adversarial training with BERT under TextFooler, and PW textual attack. In **bold** means better performance compared to Adversarial training. In red means performance is worse than Adversarial training.

	Attacker	TF→DW		TF→TB		BA→DW		BA→TB		PS→DW		PS→TB		PW→DW		PW→TB	
	Method	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv
RoBERTa	*CleanOnly*	91.7	58.5	91.7	60.0	91.7	58.5	91.7	60.0	91.7	58.5	91.7	60.0	91.7	58.5	91.7	60.0
	AdvOnly	54.5	68.0	54.5	67.1	53.0	63.3	53.0	61.5	59.6	65.7	59.6	67.5	55.6	62.5	55.6	69.3
	AdvTrain	90.0	64.7	90.0	63.2	89.5	61.4	89.5	58.5	90.1	63.0	90.1	63.8	89.4	60.3	89.4	65.8
	ModelSoup	84.7	59.7	84.7	59.2	82.6	57.4	82.6	52.6	76.7	57.0	76.7	58.9	82.5	54.6	82.5	61.1
	ADPMIXUP	**91.6**	**66.4**	**91.6**	**65.9**	**91.6**	**64.0**	**91.6**	**62.6**	**91.8**	**65.6**	**91.8**	**67.0**	**91.5**	**62.0**	**91.5**	**69.0**
BERT	*CleanOnly*	84.0	46.0	84.0	55.0	84.0	46.0	84.0	55.0	84.0	46.0	84.0	55.0	84.0	46.0	84.0	55.0
	AdvOnly	48.3	64.5	48.3	63.8	73.9	56.3	73.9	56.8	71.7	61.1	71.7	62.3	51.1	73.5	51.1	71.8
	AdvTrain	81.4	57.6	81.4	60.9	82.7	52.5	82.7	53.6	82.4	55.5	82.4	60.5	80.5	65.6	80.5	68.1
	ModelSoup	70.3	48.5	70.3	54.3	66.7	41.9	66.7	41.5	73.0	47.6	73.0	50.1	70.1	53.5	70.1	58.2
	ADPMIXUP	**83.2**	**63.1**	**83.2**	**63.0**	**83.5**	**55.1**	**83.5**	**55.8**	**83.3**	**61.1**	**83.3**	**62.1**	**82.6**	**72.6**	**82.6**	**71.3**

Table 12: Cross Attack Evaluation From Word-based to Character-based methods

Model	Attacker	DW→TF		DW→BAE		DW→PS		DW→PW		TB→TF		TB→BAE		TB→PS		TB→PW	
	Method	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv
RoBERTa	*CleanOnly*	91.7	46.8	91.7	45.6	91.7	50.3	91.7	56.1	91.7	46.8	91.7	45.6	91.7	50.3	91.7	56.1
	AdvOnly	52.7	62.7	52.7	60.2	52.7	62.3	52.7	66.0	53.5	61.3	53.5	56.1	53.5	59.3	53.5	63.4
	AdvTrain	89.4	56.4	89.4	51.5	89.4	54.7	89.4	61.4	89.0	56.0	89.0	52.1	89.0	57.4	89.0	59.4
	ModelSoup	69.5	53.5	69.5	44.9	69.5	46.2	69.5	44.3	68.9	52.0	68.9	42.0	68.9	46.0	68.9	50.5
	ADPMIXUP	**91.8**	**61.5**	**91.8**	**62.1**	**91.8**	**62.4**	**91.8**	**65.7**	**91.7**	**61.6**	**91.7**	**60.9**	**91.7**	**59.8**	**91.7**	**63.1**
BERT	*CleanOnly*	84.0	53.5	84.0	45.8	84.0	57.4	84.0	51.1	84.0	53.5	84.0	45.8	84.0	57.4	84.0	51.1
	AdvOnly	57.4	63.3	57.4	64.8	57.4	66.9	57.4	66.2	42.6	63.0	42.6	65.7	42.6	64.5	42.6	64.8
	AdvTrain	82.4	56.7	82.4	52.1	82.4	56.6	82.4	60.5	82.2	56.8	82.2	53.2	82.2	55.9	82.2	59.2
	ModelSoup	67.9	54.8	67.9	49.8	67.9	47.1	67.9	50.0	66.9	49.2	66.9	48.2	66.9	47.3	66.9	49.2
	ADPMIXUP	**82.8**	**63.5**	**82.8**	**61.3**	**82.8**	**66.0**	**82.8**	**66.5**	**83.3**	**62.0**	**83.3**	**62.7**	**83.3**	**62.3**	**83.3**	**63.5**

Table 13: Cross Attack Evaluation From Character-based to Word-based methods

Methods	**DeepWordBug**										**TextBugger**									
	MRPC		QNLI		RTE		SST2		IMDB		MRPC		QNLI		RTE		SST2		IMDB	
	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv
CleanOnly	90.0	66.3	94.8	65.8	85.2	52.7	95.9	52.6	92.5	55.1	90.0	65.8	94.8	65.5	85.2	47.3	95.9	60.3	92.5	61.0
Clean + TextFooler																				
AdvOnly	68.4	70.0	49.4	66.6	52.7	67.7	51.0	60.8	51.1	74.9	68.4	72.6	49.4	67.0	52.7	64.3	51.0	56.5	51.1	75.3
AdvTrain	87.8	68.5	92.0	64.9	84.5	66.2	95.2	59.2	90.6	64.8	87.8	70.4	92.0	64.0	84.5	57.6	95.2	53.2	90.6	70.9
ModelSoup	87.3	64.2	67.0	60.6	85.1	61.7	94.8	52.8	89.2	59.2	87.3	67.5	67.0	62.0	85.1	54.3	94.8	50.4	89.2	62.0
ADPMIXUP	**89.9**	**70.3**	**94.4**	**66.5**	**85.0**	**67.7**	**96.3**	**59.5**	**92.6**	**68.2**	**89.9**	**69.4**	**94.4**	**67.3**	**85.0**	**64.3**	**96.3**	**55.0**	**92.6**	**73.6**
Clean + BAE																				
AdvOnly	68.4	70.0	51.5	68.8	51.3	66.7	40.7	53.9	53.0	57.3	68.4	72.6	51.5	65.7	51.3	60.7	40.7	52.5	53.0	56.2
AdvTrain	88.7	68.9	94.1	67.8	79.1	64.7	94.2	50.8	91.3	54.7	88.7	70.2	94.1	62.6	79.1	57.6	94.2	50.1	91.3	52.1
ModelSoup	72.1	63.7	88.8	63.5	75.3	62.2	90.1	47.5	86.9	50.1	72.1	65.8	88.8	50.6	75.2	51.6	90.1	47.2	86.9	48.0
ADPMIXUP	**89.4**	**70.7**	**94.5**	**68.4**	**84.9**	68.0	**96.4**	56.1	**92.9**	56.9	**89.4**	**71.8**	**94.5**	**66.4**	**84.9**	**64.5**	**96.4**	**53.1**	**92.9**	**56.1**
Clean + PS																				
AdvOnly	68.4	70.0	77.2	73.7	52.3	66.2	49.7	65.5	50.2	53.2	68.4	72.6	77.2	75.4	52.3	64.3	49.7	65.9	50.2	59.2
AdvTrain	88.3	67.5	93.3	69.3	81.6	63.7	96.6	63.3	90.8	51.3	88.3	70.1	93.3	70.8	81.6	58.0	96.6	63.0	90.8	57.0
ModelSoup	70.1	60.0	87.2	60.5	58.5	60.2	85.1	57.0	82.4	47.5	70.1	62.1	87.2	68.8	58.5	53.4	85.1	58.8	82.4	51.3
ADPMIXUP	**89.3**	**68.9**	**94.6**	**72.8**	**85.8**	**68.1**	**96.6**	**65.0**	**92.8**	**53.0**	**89.3**	**71.4**	**94.6**	**75.3**	**85.8**	**64.6**	**96.6**	**64.9**	**92.8**	**58.9**
Clean + PW																				
AdvOnly	68.2	70.0	52.9	56.5	51.3	68.7	50.9	60.8	54.6	56.3	68.2	2.6	52.9	85.7	51.3	62.9	50.9	61.9	54.6	63.2
AdvTrain	87.3	67.0	92.2	54.3	82.9	67.7	95.3	58.2	89.1	54.1	87.3	71.7	92.2	83.3	82.9	60.3	95.3	57.0	89.1	56.8
ModelSoup	71.1	66.3	93.7	45.9	62.5	63.2	95.4	50.6	89.6	46.8	71.1	69.2	93.7	76.0	62.5	55.4	95.4	52.5	89.6	52.6
ADPMIXUP	**89.3**	**69.0**	**94.7**	**56.9**	**85.1**	**68.0**	**95.8**	**59.9**	**92.5**	**56.0**	**89.3**	**73.0**	**94.7**	**85.5**	**85.1**	**64.7**	**95.8**	**60.9**	**92.5**	**60.7**

Table 14: Cross Evaluation (from Word-based to Character-based) with RoBERTa

Methods	DeepWordBug										TextBugger									
	MRPC		QNLI		RTE		SST2		IMDB		MRPC		QNLI		RTE		SST2		IMDB	
	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv
CleanOnly	83.3	51.3	90.5	52.5	65.0	33.6	92.5	40.3	88.9	52.2	83.3	66.0	90.5	62.0	65.0	37.8	92.5	50.9	88.9	59.3
Clean + TextFooler																				
AdvOnly	35.6	78.6	75.3	61.4	46.9	58.0	23.6	64.6	60.1	59.9	35.6	75.8	75.3	62.3	46.9	53.8	23.6	60.4	60.1	66.9
AdvTrain	78.7	74.2	86.8	58.4	62.1	50.0	92.5	48.7	86.7	56.7	78.7	74.1	86.8	57.8	62.1	51.0	92.5	57.9	86.7	63.8
ModelSoup	68.5	64.5	76.6	53.5	54.5	42.9	83.2	32.8	68.9	48.7	68.5	71.3	76.6	54.2	54.5	46.6	83.2	43.8	68.9	55.6
ADPMIXUP	**82.1**	**76.9**	**89.6**	**60.2**	**64.1**	**56.8**	**92.3**	**62.4**	**87.9**	**59.0**	**82.1**	**75.3**	**89.6**	**61.3**	**64.1**	**53.1**	**92.3**	**59.3**	**87.9**	**65.8**
Clean + BAE																				
AdvOnly	78.7	69.6	87.6	49.0	59.6	59.7	88.0	46.6	55.6	56.4	78.7	64.9	87.6	44.8	59.6	59.4	88.0	48.1	55.6	66.9
AdvTrain	79.7	68.7	90.2	44.2	63.2	49.6	91.7	45.4	88.7	54.4	79.7	62.5	90.2	41.6	63.2	53.0	91.7	45.9	88.7	64.8
ModelSoup	71.7	58.6	68.8	40.7	53.1	35.7	86.4	32.1	53.6	42.5	71.7	45.5	68.8	35.4	53.1	38.2	86.4	42.8	53.6	45.8
ADPMIXUP	**82.0**	**69.2**	**90.2**	**48.7**	**64.6**	**55.3**	**92.1**	**46.4**	**88.6**	**55.9**	**82.0**	**64.0**	**90.2**	**44.5**	**64.6**	**57.1**	**92.1**	**47.0**	**88.6**	**66.3**
Clean + PS																				
AdvOnly	76.5	79.7	79.2	58.7	60.6	53.8	86.4	55.8	55.8	57.3	76.5	74.4	79.2	59.3	60.6	52.2	86.4	59.2	55.8	66.5
AdvTrain	80.4	74.2	89.5	55.3	63.5	47.5	93.0	47.5	85.8	53.1	80.4	74.4	89.5	56.4	63.5	51.4	93.0	56.7	85.8	63.8
ModelSoup	**71.0**	78.0	83.3	51.5	52.7	32.4	85.6	31.9	72.3	44.2	71.0	64.7	**83.3**	58.5	52.7	35.5	85.6	42.6	72.3	49.3
ADPMIXUP	**81.3**	**79.5**	**90.4**	**57.9**	**64.7**	**54.6**	**92.1**	**56.3**	**87.8**	**57.1**	**81.3**	**73.9**	**90.4**	**59.0**	**64.7**	**52.1**	**92.1**	**59.0**	**87.8**	**66.3** s
Clean + PW																				
AdvOnly	54.7	79.4	76.6	86.6	46.2	59.7	19.3	73.8	58.5	68.2	54.7	74.7	76.6	85.9	46.2	56.2	19.3	66.4	58.5	75.7
AdvTrain	78.9	74.8	85.3	83.9	59.9	48.3	90.5	56.4	87.9	64.8	78.9	72.1	85.3	86.5	59.9	51.0	90.5	59.6	87.9	71.4
ModelSoup	65.4	76.8	79.8	68.8	55.3	37.8	81.7	38.2	68.4	45.8	65.4	73.8	79.8	73.4	55.3	41.3	81.7	49.1	68.4	53.5
ADPMIXUP	**81.2**	**79.2**	**88.3**	**86.0**	**63.6**	**57.5**	**91.7**	**72.3**	**88.2**	**68.1**	**81.2**	**74.8**	**88.3**	**86.3**	**63.6**	**55.4**	**91.7**	**65.2**	**88.2**	**74.8**

Table 15: Cross Attack Evaluation (from Word-based to Character-based) with BERT

Methods	TextFooler										BAE									
	MRPC		QNLI		RTE		SST2		IMDB		MRPC		QNLI		RTE		SST2		IMDB	
	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv
CleanOnly	90.0	51.1	94.8	56.0	85.2	33.6	95.9	42.4	92.5	50.8	90.0	55.5	94.8	50.9	85.2	39.4	95.9	29.3	92.5	52.9
Clean + DeepWordBug																				
AdvOnly	66.4	58.6	55.4	64.6	51.6	75.7	39.9	52.5	50.0	61.9	66.4	65.1	55.5	54.5	51.6	66.3	39.9	55.2	50.0	59.7
AdvTrain	88.5	57.8	93.3	63.5	79.4	51.4	93.9	51.0	91.7	58.3	88.5	59.6	93.3	54.4	79.4	55.4	93.9	37.9	91.7	50.3
ModelSoup	68.4	39.1	61.0	55.2	55.6	74.3	84.6	48.5	78.0	50.6	68.4	40.4	61.0	42.9	55.6	67.4	84.6	29.3	78.0	44.6
ADPMIXUP	**90.6**	**58.9**	**94.7**	**64.1**	**84.9**	**74.0**	**96.5**	**51.5**	**92.4**	**59.2**	**90.6**	**66.1**	94.7	**53.9**	**84.9**	**73.4**	**96.5**	**60.1**	**92.4**	**57.2**
Clean + TextBugger																				
AdvOnly	62.5	60.2	50.2	65.3	52.7	75.7	50.9	48.5	51.2	56.9	62.5	64.4	50.2	53.3	52.7	69.1	50.9	39.7	51.2	54.2
AdvTrain	88.2	55.6	94.5	65.0	80.9	62.2	90.7	46.0	90.9	54.3	88.2	62.3	94.5	51.0	80.9	62.9	90.7	32.8	90.9	51.5
ModelSoup	77.2	50.0	56.4	55.0	61.0	56.8	75.2	53.0	74.6	46.3	77.2	48.6	56.4	42.8	61.0	47.4	75.2	27.6	74.6	43.4
ADPMIXUP	**89.7**	**57.3**	**95.2**	**64.9**	**85.0**	**74.8**	**96.3**	**55.4**	**92.4**	**55.8**	**89.7**	**63.9**	**95.2**	**53.0**	**85.0**	**73.5**	**96.3**	**59.7**	**92.4**	**54.6**

Table 16: Cross Attack Evaluation (from Character-based to Word-based) with RoBERTa (1)

Methods	PS					PW				
	MRPC	QNLI	RTE	SST2	IMDB	MRPC	QNLI	RTE	SST2	IMDB
CleanOnly	57.2	60.6	37.8	42.4	53.4	60.3	63.2	35.4	67.7	54.1
Clean + DeepWordBug										
AdvOnly	64.5	60.6	68.9	52.0	65.4	68.3	68.9	72.3	54.1	65.2
AdvTrain	61.8	54.5	48.1	49.3	59.6	64.6	67.5	56.9	56.6	61.6
ModelSoup	38.8	50.3	58.1	40.2	43.6	38.1	45.7	40.3	44.7	52.5
ADPMIXUP	**63.9**	**59.0**	**68.5**	**59.6**	**63.1**	**68.5**	**66.3**	**72.9**	**56.8**	**64.0**
Clean + TextBugger										
AdvOnly	64.5	58.2	68.9	47.2	57.8	64.6	67.3	72.3	54.6	58.3
AdvTrain	62.1	59.6	60.7	49.8	54.7	62.8	63.3	66.7	49.2	54.8
ModelSoup	50.7	40.0	58.5	42.8	42.1	55.6	45.5	61.0	40.7	49.6
ADPMIXUP	**63.2**	**59.4**	**69.3**	**50.8**	**56.4**	**66.1**	**65.4**	**72.8**	**54.4**	**56.9**

Table 17: Cross Attack Evaluation (from Character-based to Word-based) with RoBERTa (2)

Methods	TextFooler										BAE									
	MRPC		QNLI		RTE		SST2		IMDB		MRPC		QNLI		RTE		SST2		IMDB	
	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv
CleanOnly	83.3	64.8	90.5	62.9	65.0	35.1	92.5	52.5	88.9	52.0	83.3	60.8	90.5	53.1	65.0	37.9	92.5	27.7	88.9	49.6
Clean + DeepWordBug																				
AdvOnly	75.2	75.5	50.8	57.7	54.5	61.5	47.4	58.0	58.9	63.6	75.2	74.5	50.8	60.9	54.5	60.7	47.4	70.0	58.9	56.8
AdvTrain	81.6	67.6	90.6	56.1	61.4	46.2	91.0	54.3	87.2	59.5	81.6	68.5	90.6	58.8	61.4	47.8	91.0	31.9	87.2	53.7
ModelSoup	68.9	71.3	76.9	51.3	54.5	57.2	86.4	46.4	52.6	47.6	68.9	66.8	76.9	48.3	54.5	48.0	**86.4**	37.1	52.6	48.7
ADPMIXUP	**82.4**	**74.6**	**89.3**	**58.4**	**63.1**	**63.5**	**91.4**	**58.4**	**88.0**	**62.5**	**82.4**	**72.5**	**89.3**	**60.0**	**63.1**	**61.8**	**91.4**	**56.3**	**88.0**	**55.8**
Clean + TextBugger																				
AdvOnly	38.7	77.3	52.1	62.3	45.1	60.1	23.2	59.1	53.9	56.9	38.7	72.4	52.1	62.9	45.1	62.9	23.2	67.2	53.9	63.2
AdvTrain	80.6	69.6	90.4	57.8	57.8	48.1	93.5	53.6	88.7	54.8	80.6	64.4	90.4	58.0	57.8	49.6	93.5	33.3	88.7	60.8
ModelSoup	70.3	71.0	81.5	53.5	54.5	42.5	74.9	44.3	53.3	35.7	70.3	57.1	81.5	48.6	54.5	42.1	74.9	41.5	53.3	52.8
ADPMIXUP	**82.0**	**76.2**	**90.1**	**61.8**	**63.5**	**57.8**	**92.6**	**58.2**	**88.5**	**56.0**	**82.0**	**70.3**	**90.1**	**61.8**	**63.5**	**58.3**	**92.6**	**61.3**	**88.5**	**61.8**

Table 18: Cross Attack Evaluation (from Character-based to Word-based) with BERT (1)

Methods	PS					PW				
	MRPC	QNLI	RTE	SST2	IMDB	MRPC	QNLI	RTE	SST2	IMDB
CleanOnly	56.6	64.9	33.9	40.4	51.3	60.5	59.8	39.5	40.2	55.6
Clean + DeepWordBug										
AdvOnly	73.3	73.6	62.8	68.4	56.3	75.4	70.4	61.3	66.8	56.9
AdvTrain	66.8	70.5	48.1	43.3	53.1	75.8	69.7	52.9	50.5	53.5
ModelSoup	59.0	55.1	35.2	44.7	41.5	52.9	53.5	**57.6**	42.1	43.8
ADPMIXUP	**71.5**	**76.4**	**69.4**	**56.8**	**56.0**	**78.6**	**69.3**	**72.8**	**56.9**	**54.9**
Clean + TextBugger										
AdvOnly	69.3	74.6	65.6	61.5	51.4	72.9	73.0	60.9	58.3	58.7
AdvTrain	66.4	66.7	48.6	49.1	48.9	71.5	67.0	51.7	52.5	54.3
ModelSoup	56.9	57.3	41.2	40.9	40.3	63.2	54.6	41.8	43.0	44.5
ADPMIXUP	**68.2**	**71.4**	**59.8**	**59.7**	**52.5**	**73.6**	**71.5**	**58.2**	**57.1**	**57.3**

Table 19: Cross Attack Evaluation (from Character-based to Word-based) with BERT (2)

Attacker	PS										PW									
	MRPC		QNLI		RTE		SST2		IMDB		MRPC		QNLI		RTE		SST2		IMDB	
	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv
CleanOnly	90.0	57.2	94.8	60.6	85.2	37.8	95.9	42.4	92.5	53.4		60.3		63.2		35.4		67.7		54.1
Clean + TextFooler																				
AdvOnly	68.4	63.2	49.4	59.1	52.7	68.1	51.0	57.2	51.1	79.8	68.4	67.2	49.4	64.5	52.7	71.8	51.0	57.2	51.1	80.3
AdvTrain	87.8	60.7	92.0	56.4	84.5	60.7	95.2	52.0	90.6	71.1	87.8	64.9	92.0	61.6	84.5	64.1	95.2	56.3	90.6	67.1
ModelSoup	87.3	57.9	67.0	53.1	85.1	63.1	94.8	41.5	89.2	64.2	87.3	54.4	67.0	54.5	85.1	61.8	94.8	52.6	89.2	57.8
ADPMIXUP	**89.9**	**64.2**	**94.4**	**59.9**	**85.0**	**68.6**	**96.3**	**56.8**	**92.6**	**76.8**	**89.9**	**67.1**	**94.4**	**64.9**	**85.0**	**72.0**	**96.3**	**57.1**	**92.6**	**73.7**
Clean + BAE																				
AdvOnly	68.4	63.2	51.5	58.5	51.3	63.7	40.7	55.0	53.0	81.4	68.4	67.2	51.5	65.7	51.3	68.7	40.7	51.5	53.0	78.8
AdvTrain	88.7	60.8	94.1	56.9	79.1	61.5	94.2	51.1	91.3	68.3	88.7	64.3	94.1	61.8	79.1	67.2	94.2	48.7	91.3	64.1
ModelSoup	72.1	57.9	88.8	50.4	75.3	57.8	90.1	44.5	86.9	62.8	72.1	56.4	88.8	57.9	75.3	63.6	90.1	45.8	86.9	55.4
ADPMIXUP	**89.4**	**62.9**	**94.5**	**58.9**	**84.9**	**68.3**	**96.4**	**55.4**	**92.9**	**76.8**	**89.4**	**68.9**	**94.5**	**64.2**	**84.9**	**72.0**	**96.4**	**53.4**	**92.9**	**76.9**

Table 20: Cross Attack Evaluation between Word-based methods on RoBERTa (1)

Methods	TextFooler										BAE									
	MRPC		QNLI		RTE		SST2		IMDB		MRPC		QNLI		RTE		SST2		IMDB	
	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv
CleanOnly	90.0	51.1	94.8	56.0	85.2	33.6	95.9	42.4	92.5	50.8	90.0	55.5	94.8	50.9	85.2	39.4	95.9	29.3	92.5	52.9
Clean + PW																				
AdvOnly	68.2	58.6	52.9	90.5	51.3	68.9	50.9	55.5	54.6	60.6	68.2	65.8	52.9	86.6	51.3	68.6	50.9	44.8	54.6	79.4
AdvTrain	87.0	56.4	92.6	82.4	83.4	64.3	95.5	54.0	91.6	59.2	87.0	61.6	92.6	78.9	83.4	63.1	95.5	41.4	91.6	68.6
ModelSoup	71.1	47.0	93.7	74.3	62.5	46.2	95.4	50.5	89.6	49.4	71.1	53.0	93.7	62.1	62.5	61.4	95.4	29.3	89.6	56.8
ADPMIXUP	**89.3**	**58.0**	**94.7**	**87.9**	**85.1**	**70.4**	**95.8**	**55.8**	**92.5**	**61.0**	**89.3**	**66.4**	**94.7**	**83.2**	**85.1**	**73.0**	**95.8**	**52.4**	**92.5**	**76.8**
Clean + PS																				
AdvOnly	68.4	58.6	77.2	67.3	52.3	74.3	49.7	52.5	50.2	61.4	68.4	65.9	77.2	55.3	52.3	73.1	49.7	44.8	50.2	78.2
AdvTrain	88.3	56.7	93.3	67.1	81.6	56.8	96.6	49.5	90.8	58.6	88.3	63.0	93.3	53.6	81.6	53.1	96.6	39.7	90.8	69.9
ModelSoup	70.1	48.6	87.2	56.7	58.5	60.3	85.1	42.5	82.4	46.1	70.1	55.8	87.2	50.9	58.5	68.0	85.1	24.1	82.4	54.1
ADPMIXUP	**89.3**	**58.0**	**94.6**	**65.2**	**85.8**	**74.2**	**96.6**	**52.8**	**92.8**	**59.6**	**89.3**	**64.3**	**94.6**	**55.2**	**85.8**	**73.5**	**96.6**	**49.5**	**92.8**	**74.9**

Table 21: Cross Attack Evaluation between Word-based methods on RoBERTa (2)

Methods	PS										PW									
	MRPC		QNLI		RTE		SST2		IMDB		MRPC		QNLI		RTE		SST2		IMDB	
	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv
CleanOnly	83.3	56.6	90.5	64.9	65.0	33.9	92.5	40.4	88.9	51.3	83.3	60.5	90.5	59.8	65.0	39.5	92.5	40.2	88.9	55.6
Clean + TextFooler																				
AdvOnly	35.6	69.5	75.3	64.8	46.9	60.7	23.6	63.3	60.1	59.7	35.6	69.5	75.3	63.9	46.9	56.7	23.6	62.7	60.1	64.7
AdvTrain	78.7	66.8	86.8	60.3	62.1	43.2	92.5	47.3	86.7	56.4	78.7	74.4	86.8	59.4	62.1	47.9	92.5	51.4	86.7	60.3
ModelSoup	68.5	63.5	76.6	51.2	54.5	45.4	83.2	46.9	68.9	48.7	68.5	70.0	76.6	50.3	54.5	43.3	83.2	42.2	68.9	51.4
ADPMIXUP	**82.1**	**68.5**	**89.6**	**63.7**	**64.1**	**54.9**	92.3	**61.4**	**87.9**	**58.2**	**82.1**	**72.8**	**89.6**	**62.8**	**64.1**	**55.8**	**92.3**	**61.9**	**87.9**	**64.6**
Clean + BAE																				
AdvOnly	78.7	65.7	87.6	43.1	59.6	60.7	88.0	57.1	55.6	61.8	78.7	63.1	87.6	44.1	59.6	60.9	88.0	50.9	55.6	63.2
AdvTrain	79.7	66.1	90.2	41.0	63.2	52.5	91.7	44.4	88.7	58.6	79.7	58.6	90.2	41.9	63.2	50.8	91.7	47.5	88.7	60.9
ModelSoup	71.7	58.6	68.8	29.1	53.1	41.5	86.4	48.0	53.6	50.7	71.7	52.7	68.8	34.0	53.1	39.9	86.4	41.7	53.6	51.3
ADPMIXUP	**82.0**	**68.7**	**90.2**	**42.8**	**64.6**	**58.5**	**92.1**	**55.8**	**88.6**	**59.7**	**82.0**	**63.3**	**90.2**	**44.0**	**64.6**	**57.7**	**92.1**	**49.7**	**88.6**	**62.7**

Table 22: Cross Attack Evaluation between Word-based methods on BERT (1)

Methods	TextFooler										BAE									
	MRPC		QNLI		RTE		SST2		IMDB		MRPC		QNLI		RTE		SST2		IMDB	
	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv
Clean Only	83.3	64.8	90.5	62.9	65.0	35.1	92.5	52.5	88.9	52.0	83.3	60.8	90.5	53.1	65.0	37.9	92.5	27.7	88.9	49.6
Clean + PW																				
AdvOnly	54.7	73.1	76.6	92.8	46.2	58.2	19.3	66.1	58.5	47.9	54.7	54.8	76.6	89.4	46.2	59.4	19.3	74.8	58.5	50.8
AdvTrain	78.9	71.0	85.3	85.9	59.9	46.2	90.5	61.4	87.9	45.8	78.9	49.2	85.3	86.5	59.9	48.2	90.5	35.3	87.9	47.5
ModelSoup	65.4	66.6	79.8	71.2	55.3	39.9	81.7	47.5	68.4	39.6	65.4	52.7	79.8	66.3	55.3	39.7	81.7	51.8	68.4	40.6
ADPMIXUP	**81.2**	**71.9**	**88.3**	**90.2**	**63.6**	**52.6**	**91.7**	**64.2**	**88.2**	**46.5**	81.2	**54.0**	88.3	**89.0**	63.6	**55.8**	91.7	**71.6**	88.2	**49.3**
Clean + PS																				
AdvOnly	76.5	64.5	79.2	56.0	60.6	57.7	86.4	57.7	55.8	54.8	76.5	56.5	79.2	55.8	60.6	50.4	86.4	70.1	55.8	53.2
AdvTrain	80.4	71.0	89.5	52.9	63.5	49.5	93.0	56.4	85.8	52.5	80.4	63.4	89.5	53.9	63.5	51.3	93.0	37.1	85.8	51.4
ModelSoup	71.0	66.3	88.3	44.4	52.7	36.5	85.6	42.3	72.3	47.4	71.0	55.1	88.3	50.1	52.7	38.4	85.6	40.7	72.3	43.5
ADPMIXUP	**81.3**	**70.9**	**90.4**	**54.7**	**64.7**	**53.8**	92.1	**56.5**	**87.8**	**54.6**	**81.3**	**65.2**	**90.4**	**54.8**	64.7	**51.2**	**92.1**	**64.2**	**87.8**	**52.8**

Table 23: Cross Attack Evaluation between Word-based methods on BERT (2)

Methods	TextBugger									
	MRPC		QNLI		RTE		SST2		IMDB	
	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv
CleanOnly	90.0	65.8	94.8	65.5	85.2	47.3	95.9	60.3	92.5	61.0
Clean + DeepWordBug										
AdvOnly	66.4	71.3	55.4	72.7	51.6	65.6	39.9	65.0	50.0	80.2
AdvTrain	88.5	71.3	93.3	70.0	79.4	57.6	93.9	63.4	91.7	75.6
ModelSoup	68.4	37.6	61.0	57.3	55.6	57.9	84.6	56.8	78.0	65.1
ADPMIXUP	**90.6**	**72.8**	**94.7**	**71.6**	**84.9**	**65.5**	**96.5**	**66.9**	**92.4**	**79.5**

Table 24: Cross Attack Evaluation between Character-based methods on RoBERTa (1)

Methods	DeepWordBug									
	MRPC		QNLI		RTE		SST2		IMDB	
	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv
CleanOnly	90.0	66.3	94.8	65.8	85.2	52.7	95.9	52.6	92.5	55.1
Clean + TextBugger										
AdvOnly	62.5	70.5	50.2	69.8	52.7	65.7	50.9	58.8	51.2	71.4
AdvTrain	88.2	67.6	94.5	67.6	80.9	61.1	90.7	56.7	90.9	66.4
ModelSoup	77.2	48.4	56.4	66.8	61.0	47.3	75.2	45.6	74.6	59.3
ADPMIXUP	**89.7**	**69.9**	**95.2**	**71.8**	**85.0**	**67.5**	**96.3**	**60.3**	**92.4**	**70.6**

Table 25: Cross Attack Evaluation between Character-based methods on RoBERTa (2)

Methods	TextBugger									
	MRPC		QNLI		RTE		SST2		IMDB	
	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv
CleanOnly	83.3	66.0	90.5	62.0	65.0	37.8	92.5	50.9	88.9	59.3
Clean + DeepWordBug										
AdvOnly	75.2	74.1	50.8	66.4	54.5	63.3	47.4	69.6	58.9	74.5
AdvTrain	81.6	77.7	90.6	62.7	61.4	56.2	91.0	66.6	87.2	71.4
ModelSoup	68.9	56.0	76.9	55.2	54.5	51.4	86.4	50.7	52.6	53.3
ADPMIXUP	**82.4**	**75.9**	**89.3**	**65.2**	**63.1**	**62.8**	**91.4**	**69.0**	**88.0**	**73.9**

Table 26: Cross Attack Evaluation between Character-based methods on BERT (1)

Methods	DeepWordBug									
	MRPC		QNLI		RTE		SST2		IMDB	
	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv
CleanOnly	83.3	51.3	90.5	52.5	65.0	33.6	92.5	40.3	88.9	52.2
Clean + TextBugger										
AdvOnly	38.7	80.9	52.1	67.6	45.1	68.1	23.2	69.7	53.9	75.3
AdvTrain	80.6	77.4	90.4	65.4	57.8	54.6	93.5	62.1	88.7	72.2
ModelSoup	70.3	69.4	81.5	51.6	54.5	66.4	74.9	34.6	53.3	58.7
ADPMIXUP	**82.0**	**80.1**	**90.0**	**66.9**	**59.8**	**67.5**	**92.0**	**65.8**	**88.7**	**75.1**

Table 27: Cross Attack Evaluation between Character-based methods on BERT (2)

	Methods	(TF, PW)→BAE										(TF, PW)→PS									
		MRPC		QNLI		RTE		SST2		IMDB		MRPC		QNLI		RTE		SST2		IMDB	
		Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv
RoBERTa	*CleanOnly*	90.0	55.5	94.8	50.9	85.2	39.4	95.9	29.3	92.5	52.9	90.0	57.2	94.8	60.6	85.2	37.8	95.9	42.4	92.5	53.4
	AdvOnly	68.4	65.8	49.5	85.2	52.7	73.1	49.1	48.3	52.4	79.9	68.4	63.9	49.5	59.5	52.7	69.2	49.1	57.9	52.4	82.0
	AdvTrain	86.2	62.5	91.5	79.3	81.9	65.2	92.3	42.6	90.2	70.1	86.2	60.9	91.5	56.9	81.9	62.5	92.3	53.1	90.2	73.5
	ModelSoup	51.6	49.3	81.5	63.7	60.4	52.9	76.8	43.2	80.5	54.2	51.6	55.5	81.5	54.2	60.4	56.3	76.8	43.6	80.5	46.9
	ADPMIXUP	89.7	67.0	94.5	84.8	85.7	73.9	95.3	54.8	92.0	77.9	90.2	64.2	94.6	61.8	85.0	68.9	96.4	58.3	92.4	78.1
BERT	*CleanOnly*	83.3	60.8	90.5	53.1	65.0	37.9	92.5	27.7	88.9	49.6	83.3	56.6	90.5	64.9	65.0	33.9	92.5	40.4	88.9	51.3
	AdvOnly	41.7	67.1	70.5	89.6	44.8	60.8	18.7	75.3	52.1	55.6	41.7	70.0	70.5	65.2	44.8	62.7	18.7	70.2	52.1	64.3
	AdvTrain	78.0	62.8	83.9	86.5	58.6	50.3	88.5	40.9	86.3	52.2	78.0	69.0	83.9	61.9	58.6	44.5	88.5	49.5	86.3	56.5
	ModelSoup	64.3	53.4	77.3	60.2	43.7	33.8	63.5	34.9	63.4	43.8	64.3	46.9	77.3	53.8	43.7	39.9	63.5	44.3	63.4	44.5
	ADPMIXUP	82.0	54.3	88.1	90.2	63.7	56.1	91.2	72.9	88.5	49.3	82.7	69.0	89.5	66.9	64.5	57.3	92.5	64.3	88.0	61.8

Table 28: Cross Evaluation between Word-based attacks when knowing 2 adversarial attacks

	Methods	(BAE, PS)→TF										(BAE, PS)→PW									
		MRPC		QNLI		RTE		SST2		IMDB		MRPC		QNLI		RTE		SST2		IMDB	
		Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv
RoBERTa	*CleanOnly*	90.0	51.1	94.8	56.0	85.2	33.6	95.9	42.4	92.5	50.8	90.0	60.3	94.8	63.2	85.2	35.4	95.9	67.7	92.5	54.1
	AdvOnly	68.4	59.6	70.5	68.2	52.1	76.8	49.1	54.5	45.8	62.4	68.4	67.4	70.5	70.3	52.1	71.8	49.1	59.0	45.8	80.3
	AdvTrain	86.2	56.6	91.0	67.5	80.6	59.8	92.5	52.3	89.3	59.8	86.2	65.1	91.0	64.9	80.6	68.3	92.5	50.5	89.3	66.8
	ModelSoup	67.3	44.3	72.1	46.8	50.4	52.5	65.4	44.9	63.9	39.8	67.3	50.3	72.1	50.3	50.4	49.6	65.4	44.2	63.9	50.4
	ADPMIXUP	89.9	58.7	94.6	67.8	85.1	75.1	96.0	53.1	92.9	61.3	89.9	68.0	94.4	66.3	85.0	74.5	95.9	55.6	92.1	78.9
BERT	*CleanOnly*	83.3	64.8	90.5	62.9	65.0	35.1	92.5	52.5	88.9	52.0	83.3	60.5	90.5	59.8	65.0	39.5	92.5	40.2	88.9	55.6
	AdvOnly	44.1	66.5	40.7	64.9	43.7	59.3	18.8	59.9	51.2	56.7	44.1	68.3	40.7	47.7	43.7	64.6	18.8	59.4	51.2	66.8
	AdvTrain	79.4	72.8	88.4	55.8	61.8	52.3	91.5	57.1	84.2	53.4	79.4	61.5	88.4	52.4	61.8	53.3	91.5	49.8	84.2	62.7
	ModelSoup	60.3	67.4	80.2	46.8	50.8	31.1	64.3	43.8	56.4	49.8	60.3	55.7	80.2	35.9	50.8	41.3	64.3	39.8	56.4	43.8
	ADPMIXUP	82.1	71.3	90.0	58.5	65.5	54.9	92.6	57.8	88.2	55.8	83.1	64.0	90.2	46.4	63.1	58.2	92.4	53.2	88.0	64.9

Table 29: Cross Evaluation between Word-based attacks when knowing 2 adversarial attacks

	Methods	(TF, PW, BAE)→PS										(PW, BAE, PS)→TF									
		MRPC		QNLI		RTE		SST2		IMDB		MRPC		QNLI		RTE		SST2		IMDB	
		Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv
RoBERTa	*CleanOnly*	90.0	57.2	94.8	60.6	85.2	37.8	95.9	42.4	92.5	53.4	90.0	51.1	94.8	56.0	85.2	33.6	95.9	42.4	92.5	50.8
	AdvOnly	68.4	64.2	49.1	60.1	52.0	70.4	50.3	59.3	48.2	82.0	68.5	59.5	71.6	70.7	52.7	74.3	50.8	56.5	45.4	63.0
	AdvTrain	86.2	62.5	90.2	57.8	81.9	64.8	91.2	55.7	86.3	78.4	85.1	57.3	88.5	69.1	78.4	63.4	90.1	54.0	86.8	61.7
	ModelSoup	78.3	53.2	59.4	51.6	61.4	57.4	54.3	45.9	41.5	44.8	62.5	40.5	64.9	41.8	45.7	47.9	53.2	40.8	50.6	36.9
	ADPMIXUP	89.9	64.8	94.5	62.5	85.1	69.0	96.1	59.0	92.0	80.2	90.4	59.6	94.2	72.5	85.0	76.2	95.6	55.2	92.6	62.9
BERT	*CleanOnly*	83.3	56.6	90.5	64.9	65.0	33.9	92.5	40.4	88.9	51.3	83.3	64.8	90.1	62.9	64.5	35.1	92.5	52.5	88.9	52.0
	AdvOnly	43.6	69.7	70.5	66.8	46.2	65.3	17.7	67.6	50.9	68.9	57.6	70.3	58.4	64.3	43.3	57.7	17.1	60.2	51.0	57.1
	AdvTrain	75.0	69.8	83.1	64.2	58.1	50.4	88.4	51.2	82.5	59.4	76.3	73.0	87.0	57.4	60.2	53.9	87.4	57.7	82.4	53.0
	ModelSoup	50.6	42.8	60.4	45.8	38.6	40.3	54.2	45.8	51.4	41.8	50.9	54.8	58.3	44.9	45.9	32.5	60.2	41.9	50.6	44.5
	ADPMIXUP	83.0	69.0	89.0	68.0	64.8	60.8	92.0	66.2	88.5	64.9	83.0	72.1	90.1	61.8	65.2	59.8	92.0	58.2	88.5	56.3

Table 30: Cross Evaluation between Word-based attacks when knowing 3 adversarial attacks

	Methods	(TF, BAE, PS)→PW										(TF, PW, PS)→BAE									
		MRPC		QNLI		RTE		SST2		IMDB		MRPC		QNLI		RTE		SST2		IMDB	
		Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv
RoBERTa	*CleanOnly*	90.0	60.3	94.8	63.2	85.2	35.4	95.9	67.7	92.5	54.1	90.0	55.5	94.8	50.9	85.2	39.4	95.9	29.3	92.5	52.9
	AdvOnly	68.4	67.2	49.5	74.5	52.7	72.6	50.3	61.7	53.9	81.2	68.4	65.8	49.5	85.3	52.7	73.9	48.3	49.1	51.6	80.3
	AdvTrain	86.0	66.3	90.1	66.3	75.1	69.8	90.4	53.4	86.3	69.0	83.1	64.0	89.4	81.6	78.9	68.1	89.1	44.8	88.9	72.5
	ModelSoup	61.2	42.7	66.1	43.8	56.3	43.2	70.6	51.0	66.2	44.2	45.7	43.6	76.3	54.9	55.8	50.8	72.1	40.3	67.6	51.9
	ADPMIXUP	90.1	68.4	94.6	68.9	84.9	74.0	95.5	58.9	92.4	80.5	89.9	66.8	94.0	86.0	85.8	73.0	95.8	56.0	93.0	79.6
BERT	*CleanOnly*	83.3	60.5	90.5	59.8	65.0	39.5	92.5	40.2	88.9	55.6	83.5	60.8	90.7	53.1	65.0	37.9	92.5	27.7	88.9	49.6
	AdvOnly	39.7	70.0	40.2	48.6	48.4	66.7	19.2	61.8	50.5	67.8	50.2	58.8	68.4	90.2	42.6	62.7	18.7	75.4	51.4	57.4
	AdvTrain	74.8	63.6	86.3	54.9	58.9	55.8	88.4	51.4	85.1	63.0	76.1	64.2	80.3	87.0	57.1	55.2	85.1	46.8	84.1	54.0
	ModelSoup	56.8	53.9	56.8	36.8	41.9	45.9	60.5	34.7	44.7	40.7	60.1	45.6	64.9	54.8	40.8	31.2	56.8	30.8	54.3	41.7
	ADPMIXUP	83.5	64.2	90.1	49.8	64.2	63.5	91.8	56.4	89.2	65.0	83.2	56.9	88.9	90.0	64.2	59.8	91.9	74.1	88.6	52.5

Table 31: Cross Evaluation between Word-based attacks when knowing 3 adversarial attacks