ADDMU: Detection of Far-Boundary Adversarial Examples with Data and Model Uncertainty Estimation

Fan Yin
University of California, Los Angeles
fanyin20@cs.ucla.edu

Yao Li
University of North Carolina, Chapel Hill
yaoli@email.unc.edu

Cho-Jui Hsieh
University of California, Los Angeles
chohsieh@cs.ucla.edu

Kai-Wei Chang
University of California, Los Angeles
kwchang@cs.ucla.edu

Abstract

Adversarial Examples Detection (AED) is a crucial defense technique against adversarial attacks and has drawn increasing attention from the Natural Language Processing (NLP) community. Despite the surge of new AED methods, our studies show that existing methods heavily rely on a shortcut to achieve good performance. In other words, current search-based adversarial attacks in NLP stop once model predictions change, and thus most adversarial examples generated by those attacks are located near model decision boundaries. To surpass this shortcut and fairly evaluate AED methods, we propose to test AED methods with **F**ar **B**oundary (**FB**) adversarial examples. Existing methods show worse than random guess performance under this scenario. To overcome this limitation, we propose a new technique, **ADDMU**, **a**dversary **d**etection with **d**ata and **m**odel **u**ncertainty, which combines two types of uncertainty estimation for both regular and FB adversarial example detection. Our new method outperforms previous methods by 3.6 and 6.0 *AUC* points under each scenario. Finally, our analysis shows that the two types of uncertainty provided by **ADDMU** can be leveraged to characterize adversarial examples and identify the ones that contribute most to model's robustness in adversarial training.

1 Introduction

Deep neural networks (DNN) have achieved remarkable performance in a wide variety of NLP tasks. However, it has been shown that DNNs can be vulnerable to adversarial examples (Jia and Liang, 2017; Alzantot et al., 2018; Jin et al., 2020), i.e., perturbed examples that flip model predictions but remain imperceptible to humans, and thus impose serious security concerns about NLP models.

To improve the robustness of NLP models, different kinds of techniques to defend against adversarial examples have been proposed (Li et al., 2021b). In this paper, we study AED, which aims to add a detection module to identify and reject malicious inputs based on certain characteristics. Different from adversarial training methods (Madry et al., 2018a; Jia et al., 2019) which require re-training of the model with additional data or regularization, AED operates in the test time and can be directly integrated with any existing model.

Despite being well explored in the vision domain (Feinman et al., 2017; Raghuram et al., 2021), AED started to get attention in the field of NLP only recently. Many works have been proposed to conduct detection based on certain statistics (Zhou et al., 2019; Mozes et al., 2021; Yoo et al., 2022; Xie et al., 2022). Specifically, Yoo et al. (2022) propose a benchmark for AED methods and a competitive baseline by robust density estimation. However, by studying examples in the benchmark, we find that the success of some AED methods relies heavily on the shortcut left by adversarial attacks: most adversarial examples *are located near model decision boundaries*, i.e., they have small probability discrepancy between the predicted class and the second largest class. This is because when creating adversarial data, the searching process stops once model predictions changed. We illustrate this finding in Section 2.2.

To evaluate detection methods accurately, we propose to test AED methods on both regular adversarial examples and **F**ar-**B**oundary (**FB**)[1] adversarial examples, which are created by continuing to search for better adversarial examples till a threshold of probability discrepancy is met. Results show that existing AED methods perform worse than random guess on FB adversarial examples. Yoo et al. (2022) recognize this limitation, but we find that this phenomenon is more severe than what is reported in their work. Thus, an AED method that works for FB attacks is in need.

[1] Other works may call this 'High-Confidence'. We use the term 'Far-Boundary' to avoid conflicts between 'confidence' and the term 'uncertainty' introduced later.

We propose ADDMU, an uncertainty estimation based AED method. The key intuition is based on the fact that adversarial examples lie off the manifold of training data and models are typically uncertain about their predictions of them. Thus, although the prediction probability is no longer a good uncertainty measurement when adversarial examples are far from the model decision boundary, there exist other statistical clues that give out the 'uncertainty' in predictions to identify adversarial data. In this paper, we introduce two of them: *data uncertainty* and *model uncertainty*. Data uncertainty is defined as the uncertainty of model predictions over neighbors of the input. Model uncertainty is defined as the prediction variance on the original input when applying Monte Carlo Dropout (MCD) (Gal and Ghahramani, 2016) to the target model during inference time. Previous work has shown that models trained with dropout regularization (Srivastava et al., 2014) approximate the inference in Bayesian neural networks with MCD, where model uncertainty is easy to obtain (Gal and Ghahramani, 2016; Smith and Gal, 2018). Given the statistics of the two uncertainties, we apply p-value normalization (Raghuram et al., 2021) and combine them with Fisher's method (Fisher, 1992) to produce a stronger test statistic for AED. To the best of our knowledge, we are the first work to estimate the uncertainty of Transformer-based models (Shelmanov et al., 2021) for AED.

The advantages of our proposed AED method include: 1) it only operates on the output level of the model; 2) it requires little to no modifications to adapt to different architectures; 3) it provides an unified way to combine different types of uncertainties. Experimental results with on four datasets, four attacks, and two models demonstrate that our method outperforms existing methods by 3.6 and 6.0 in terms of AUC scores on regular and FB cases, respectively. We also show that the two uncertainty statistics can be used to characterize adversarial data and select useful data for another defense technique, adversarial data augmentation (ADA).

2 A Diagnostic Study on AED Methods

In this section, we first describe the formulation of adversarial examples and AED. Then, we show that current AED methods mainly act well on detecting adversarial examples near the decision boundary, but are confused by FB adversarial examples.

2.1 Formulation

Adversarial Examples. Given an NLP model $f : \mathcal{X} \rightarrow \mathcal{Y}$, a textual input $x \in \mathcal{X}$, a predicted class from the candidate classes $y \in \mathcal{Y}$, and a set of boolean indicator functions of constraints, $\mathcal{C}_i : \mathcal{X} \times \mathcal{X} \rightarrow \{0,1\}, i = 1, 2, \cdots, n$. An (untargeted) adversarial example $x^* \in \mathcal{X}$ satisfies:

$$f(x^*) \neq f(x), \mathcal{C}_i(x, x^*) = 1, i = 1, 2, \cdots, n.$$

Constraints are typically grammatical or semantic similarities between original and adversarial data. For example, Jin et al. (2020) conduct part-of-speech checks and use Universal Sentence Encoder (Cer et al., 2018) to ensure semantic similarities between two sentences.

Adversarial Examples Detection (AED) The task of AED is to distinguish adversarial examples from natural ones, based on certain characteristics of adversarial data. We assume access to 1) the victim model f, trained and tested on clean datasets $\mathcal{D}_{train}$ and $\mathcal{D}_{test}$; 2) an evaluation set $\mathcal{D}_{eval}$; 3) an auxiliary dataset $\mathcal{D}_{aux}$ contains only clean data. $\mathcal{D}_{eval}$ contains equal number of adversarial examples $\mathcal{D}_{eval-adv}$ and natural examples $\mathcal{D}_{eval-nat}$. $\mathcal{D}_{eval-nat}$ are randomly sampled from $\mathcal{D}_{test}$. $\mathcal{D}_{eval-adv}$ is generated by attacking a disjoint set of samples from $\mathcal{D}_{eval-nat}$ on $\mathcal{D}_{test}$. See Scenario 1 in Yoo et al. (2022) for details. We use a subset of $\mathcal{D}_{train}$ as $\mathcal{D}_{aux}$. We adopt an unsupervised setting, i.e., the AED method is not trained on any dataset that contains adversarial examples.

2.2 Diagnose AED Methods

We define examples *near model decision boundaries* to be those whose output probabilities for the predicted class and the second largest class are close. Regular iterative adversarial attacks stop once the predictions are changed. Therefore, we suspect that regular attacks are mostly generating adversarial examples near the boundaries, and existing AED methods could rely on this property to detect adversarial examples.

Figure 1 verifies this for the state-of-the-art unsupervised AED method (Yoo et al., 2022) in NLP, denoted as **RDE**. Similar trends are observed for another baseline. The X-axis shows two attack methods: TextFooler (Jin et al., 2020) and Pruthi (Pruthi et al., 2019). The Y-axis represents the probability difference between the predicted class and the second largest class. Average probability differences of natural examples (Natural), and three types of

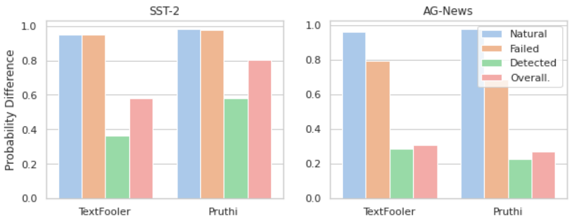

Figure 1: The probability difference between the predicted class and the second largest class on natural examples, adversarial examples that the detector failed, succeed, and in total. The X-axis is the attack. The Y-axis is the difference. Correctly detected adversarial examples have relatively small probability difference.

	RDE		DIST	
Data-attack	**Regular**	**FB**	**Regular**	**FB**
SST2-TF	72.8/86.5	45.0/81.5	73.4/87.9	26.3/81.6
SST2-Pruthi	55.1/80.6	30.8/72.6	61.4/85.3	26.5/74.6
Yelp-TF	79.2/89.6	44.6/82.7	80.3/90.6	64.3/86.2
Yelp-Pruthi	64.8/88.0	47.9/85.2	72.2/89.2	55.2/84.9

Table 1: F1/AUC scores of two SOTA detection methods on **Regular** and **FB** adversarial examples. RDE and DIST perform worse than random guess (F1=50.0) on FB adversarial examples.

adversarial examples are shown: RDE fails to identify (Failed), successfully detected (Detected), and overall (Overall). There is a clear trend that successfully detected adversarial examples are those with small probability differences while the ones with high probability differences are often mis-classified as natural examples. This finding shows that these AED methods identify examples near the decision boundaries, instead of adversarial examples.

To better evaluate AED methods, we propose to avoid the above shortcut by testing detection methods with FB adversarial examples, which are generated by continuously searching for adversarial examples until a prediction probability threshold is reached. We simply add another goal function to the adversarial example definition to achieve this while keep other conditions unchanged:

$$f\left(x^{*}\right) \neq f\left(x\right), p\left(y=f\left(x^{*}\right) \mid x^{*}\right) \geq \epsilon$$
$$\mathcal{C}_i\left(x, x^{*}\right)=1, i=1,2,\cdots,n.$$

$p\left(y=f\left(x^{*}\right) \mid x^{*}\right)$ denotes the predicted probability for the adversarial example. ϵ is a manually defined threshold. We illustrate the choice of ϵ in Section 4.1. In Table 1, it shows that the existing competitive methods (RDE and DIST) get lower than random guess F1 scores when evaluated with FB adversarial examples.

	Grammar		Semantics	
Data	**Regular**	**FB**	**Regular**	**FB**
SST-2	1.117	1.129	3.960	3.900
Yelp	1.209	1.233	4.113	4.082

Table 2: Quality checks for FB adversarial examples. The results on each dataset are averaged over examples from three attacks: TextFooler, BAE, Pruthi, and their FB versions. The quality of adversarial examples do not degrade much with the FB version of attacks.

2.3 Quality Check for FB Attacks

We show that empirically, even we search for more steps and stronger FB adversarial examples, the quality of adversarial examples do not significantly degrade. We follow Morris et al. (2020b) to evaluate the quality of FB adversarial examples in terms of grammatical and semantic changes, and compare them with regular adversarial examples. We use a triple (x, x_{adv}, x_{FB-adv}) to denote the original example, its corresponding regular adversarial and FB adversarial examples. For grammatical changes, we conduct an automatic evaluation with LanguageTool (Naber et al., 2003) to count grammatical errors and report the relative increase of errors of perturbed examples w.r.t. original examples. For semantic changes, we do a human evaluation using Amazon MTurk [2]. We ask the workers to rate to what extent the changes to x preserve the meaning of the sentence, with scale 1 ('Strongly disagree') to 5 ('Strongly agree'). Results are summarized in Table 2. The values are averaged over three adversarial attacks, 50 examples for each. We find that the FB attacks have minimal impact on the quality of the adversarial examples.

3 Adversary Detection with Data and Model Uncertainty (ADDMU)

Given the poor performance of previous methods on FB attacks, we aim to build a detector that can handle not only regular but also FB adversarial examples. We propose ADDMU, an uncertainty estimation based AED method by combing two types of uncertainty: model uncertainty and data uncertainty. We expect the adversarial examples to have large values for both. The motivation of using uncertainty is that models can still be uncertain about their predictions even when they assign a high probability of predicted class to an example.

[2] We pay workers 0.05 dollars per HIT. Each HIT takes approximately 15 seconds to finish. So, we pay each worker 12 dollars per hour. Each HIT is assigned three workers.

We describe the definitions and estimations of the two uncertainties, and how to combine them.

3.1 Model Uncertainty Estimation

Model uncertainty represents the uncertainty when predicting a single data point with randomized models. Gal and Ghahramani (2016) show that model uncertainty can be extracted from DNNs trained with dropout and inference with MCD without any modifications of the network. This is because the training objective with dropout minimizes the Kullback-Leibler divergence between the posterior distribution of a Bayesian network and an approximation distribution. We follow this approach and define the model uncertainty as the softmax variance when applying MCD during test time.

Specifically, given a trained model f, we do N_m stochastic forward passes for each data point x. The dropout masks of hidden representations for each forward pass are i.i.d sampled from a Bernolli distribution, i.e., $z_{lk} \sim \textit{Bernolli}\,(p_m)$ where p_m is a fixed dropout rate for all layers, z_{lk} is the mask for neuron k on layer l. Then, we can do a Monte Carlo estimation on the softmax variance among the N_m stochastic softmax outputs. Denote the probability of predicting the input as the i-th class in the j-th forward pass as p_{ij} and the mean probability for the i-th class over N_m passes as $\bar{p}_i = \frac{1}{N_m}\sum_{j=1}^{N_m} p_{ij}$, the model uncertainty (MU) can be computed by

$$MU\,(x) = \frac{1}{|\mathcal{Y}|}\sum\nolimits_{i=1}^{|\mathcal{Y}|}\frac{1}{N_m}\sum\nolimits_{j=1}^{N_m}\left(p_{ij}-\bar{p}_i\right)^2.$$

3.2 Data Uncertainty Estimation

Data uncertainty quantifies the predictive probability distribution of a fixed model over the neighborhood of an input point.

Specifically, similar to the model uncertainty estimation, we do N_d stochastic forward passes. But instead of randomly zeroing out neurons in the model, we fix the trained model and construct a stochastic input for each forward pass by masking out input tokens, i.e., replacing each token in the original input by a special token with probability p_d. The data uncertainty is estimated by the mean of (1 $-$ maximum softmax probability) over the N_d forward passes. Denote the N_d stochastic inputs as $x_1, x_2, \cdots, x_{N_d}$, the original prediction as y, and the predictive probability of the original predicted class as $p_y\,(\cdot)$, the Monte Carlo estimation on data uncertainty (DU) is:

$$DU\,(x) = \frac{1}{N_d}\sum\nolimits_{i=1}^{N_d}\left(1-p_y\,(x_i)\right).$$

3.3 Aggregate Uncertainties with Fisher's Method

We intend to aggregate the two uncertainties described above to better reveal the low confidence of model's prediction on adversarial examples. We first normalize the uncertainty statistics so that they follow the same distribution. Motivated by Raghuram et al. (2021) where the authors normalize test statistics across layers by converting them to p-values, we also adopt the same method to normalize the two uncertainties. By definition, a p-value computes the probability of a test statistic being at least as extreme as the target value. The transformation will convert any test statistics into a uniformly distributed probability. We construct empirical distributions for MU and DU by calculating the corresponding uncertainties for each example on the auxiliary dataset $\mathcal{D}_{aux}$, denoted as T_{mu}, and T_{du}. Following the null hypothesis H_0*: the data being evaluated comes from the clean distribution*, we can calculate the p-values based on model uncertainty (q_m) and data uncertainty (q_d) by:

$$q_m\,(x) = \mathbb{P}\left(T_{mu} \geq MU\,(x) \mid H_0\right),$$
$$q_d\,(x) = \mathbb{P}\left(T_{du} \geq DU\,(x) \mid H_0\right).$$

The smaller the values q_m and q_d, the higher the probability of the example being adversarial.

Given q_m and q_d, we combine them into a single p-value using the Fisher's method to do combined probability test (Fisher, 1992). Fisher's method indicates that under the null hypothesis, the sum of the log of the two p-values follows a χ^2 distribution with 4 degrees of freedom. We use q_{agg} to denote the aggregated p-value. Adversarial examples should have smaller q_{agg}, where $\log q_{agg} = \log q_m + \log q_d$.

4 Experiments

We first describe the experimental setup (Section 4.1), then present our results on both regular and FB AED (Section 4.2). Results show that our ADDMU outperforms existing methods by a large margin under both scenarios.

4.1 Experimental Setup

Datasets. We conduct experiments on classification tasks in different domains, including sentiment analysis SST-2 (Socher et al., 2013), Yelp (Zhang et al., 2015), topic classification AGNews (Zhang et al., 2015), and natural language inference SNLI (Bowman et al., 2015). We generate both regular and FB adversarial examples on the test data of each dataset with two word-level attacks: TextFooler (TF) (Jin et al., 2020), BAE (Garg and Ramakrishnan, 2020), and two character-level attacks: Pruthi (Pruthi et al., 2019), and TextBugger (TB) (Li et al., 2019). The numbers of evaluated examples vary among 400 to 4000 across datasets. See Appendix A. For FB adversarial examples, we choose the ϵ so that adversarial examples have approximately equal averaged prediction probability with natural data. Specifically, $\epsilon = 0.9$ for SST-2, Yelp, AGNews, and $\epsilon = 0.7$ for SNLI. The victim models are BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019). More details about these datasets are presented in Appendix A.

Baselines. We compare **ADDMU** with several unsupervised AED methods. 1) **MSP:** Hendrycks and Gimpel (2017) use the Maximum Softmax Probability (MSP) for detection; 2) **PPL:** GPT-2 large (Radford et al., 2019) as a language model to measure the perplexity of the input; 3) **FGWS:** Mozes et al. (2021) measure the difference in prediction probability after replacing infrequent words of the inputs with frequent words and find that adversarial examples have higher performance change; 4) **RDE:** Yoo et al. (2022) fit class conditional density estimation with Kernel PCA (Schölkopf et al., 1998) and Minimum Covariance Determinant (Rousseeuw, 1984) in the feature space and use the density scores; 5) **DIST:** we propose a distance-based baseline that uses the difference between class conditional, averaged K nearest distances. See Appendix B for details.

Unsupervised AED methods assign a value to each evaluated data. Then, a threshold is selected based on the maximum False Positive Rate (FPR) allowed, i.e., the rate of mis-classified natural data.

Implementation Details. For **FGWS** and **RDE**, we follow the hyper-parameters in their papers to reproduce the numbers. For **DIST** and **ADDMU**, we attack the validation set and use those examples to tune the hyper-parameters. See Appendix C for details. Specifically, for **DIST**, we use 600 neighbors. For **ADDMU**, we find $N_m = 10$, $p_m = 0.2$ for MU works well for all datasets. For DU, we find that it is beneficial to ensemble different mask rates for text classification tasks, we set $N_d = 100$ in total, and 25 for each $p_d \in \{0.1, 0.2, 0.3, 0.4\}$ for all the text classification tasks, $N_d = 25$, $p_d = 0.1$ for SNLI.

Metrics. In the main experiments, we select the threshold at maximum FPR=0.1. A lower FPR represents a more practical case where only a small proportion of natural samples are mis-classified as adversarial samples. Following the setup in Xu et al. (2018) and Yoo et al. (2022), we report True Positive Rate (TPR), F1 score at FPR=0.1, and Area Under the ROC curve (AUC).

4.2 Results

Performances of AED methods on BERT are presented in Table 3. We average the results among three runs with different random seeds. See Appendix E for the results on RoBERTa.

Detector performance. Our proposed ADDMU achieves the best performance on both regular and FB adversarial examples under the three metrics (TPR, F1, AUC) on the four datasets, which demonstrates the effectiveness of ADDMU. Further, ADDMU preserves more than 90% of the performance or even achieves better results, e.g SST-2-Pruthi and Yelp-BAE, under FB adversarial attacks, which shows that ADDMU is not affected by FB attacks.

The performances of MSP, DIST, and RDE are severely degraded under FB attacks. This demonstrates that those methods can be fooled and circumvented by carefully designed attacks. Under regular attacks, the performances of RDE and DIST are worse than the baseline MSP in most cases, which simply uses the maximum softmax probability for detection. One explanation is that those class conditional methods are just approximating softmax probabilities so might not be as effective as MSP in detecting near the decision boundary examples.

Finally, PPL and FGWS are also not severely affected by FB attacks. However, FGWS is only applicable to word-level attacks. Also, PPL and FGWS are not effective enough in general.

Ablation study. Data uncertainty (**DU**) and model uncertainty (**MU**) can also be used as features in detection separately. Also, both **RDE** and **DIST** can be enhanced by calculating the average score over the neighborhood of the input using the same random masking technique as used in data uncertainty estimation. We denote them as **RDE-aug** and **DIST-aug**. In this part, we study the effectiveness of uncertainty aggregation and neighbor augmentation by comparing ADDMU with DU and MU, and by comparing RDE and DIST with RDE-aug and DIST-aug. Full results are shown in Appendix F. We show a representative proportion of the results in Table 4. The summary of findings

		SST-2			AGNews			Yelp			SNLI		
Attacks	**Methods**	**TPR**	**F1**	**AUC**	**TPR**	**F1**	**AUC**	**TPR**	**F1**	**AUC**	**TPR**	**F1**	**AUC**
TF	PPL	31.2	44.2	72.4	76.1	81.8	91.1	45.7	58.8	79.3	40.2	53.6	78.0
	FGWS	62.9	72.8	76.5	83.0	86.3	85.5	67.1	72.7	80.6	48.5	55.4	72.2
	MSP	64.0	73.6	88.0	95.2	92.8	97.5	73.9	80.4	90.6	56.7	68.0	83.6
	RDE	62.9	72.8	86.5	96.0	93.2	97.0	72.0	79.2	89.6	46.3	59.3	81.0
	DIST	64.0	73.4	87.9	94.5	92.4	95.9	73.8	80.3	90.6	37.2	50.4	74.5
	ADDMU	**67.1**	**75.8**	**88.8**	**99.2**	**94.9**	**98.6**	**78.7**	**83.5**	**91.6**	**68.9**	**77.0**	**89.7**
TF-FB	PPL	41.9	55.2	80.6	83.3	86.3	93.9	49.9	62.4	81.6	44.1	57.2	79.2
	FGWS	61.8	72.0	77.9	84.8	87.1	88.1	72.2	78.0	89.4	52.1	59.6	78.4
	MSP	31.2	44.2	81.9	82.0	85.4	91.5	66.0	75.0	87.1	26.8	39.2	75.1
	RDE	31.9	45.0	81.5	71.9	79.1	92.5	31.5	44.6	82.7	43.1	56.4	79.6
	DIST	20.7	26.3	81.6	66.6	75.4	91.8	54.8	64.3	86.2	27.2	39.6	69.9
	ADDMU	**62.0**	**72.2**	**88.0**	**97.5**	**94.0**	**97.8**	**72.8**	**79.7**	**89.7**	**53.6**	**65.8**	**87.5**
BAE	PPL	19.7	30.4	66.2	30.9	44.0	71.8	23.6	35.3	70.1	24.8	36.8	68.1
	FGWS	37.6	51.0	64.2	64.7	74.2	72.5	54.9	66.7	68.0	31.2	44.0	67.9
	MSP	45.1	58.3	79.0	96.0	93.4	96.0	68.3	76.7	89.5	41.4	54.7	71.4
	RDE	44.2	57.3	79.3	96.4	**93.7**	96.3	65.2	74.5	89.1	41.7	55.0	76.8
	DIST	44.9	57.3	78.9	94.2	91.9	96.2	68.0	76.2	89.4	36.8	49.7	67.9
	ADDMU	**45.9**	**58.9**	**82.3**	**96.4**	93.5	**97.3**	**72.5**	**79.5**	**90.1**	**48.2**	**61.0**	**81.0**
BAE-FB	PPL	26.0	38.2	70.5	45.5	58.7	79.6	28.5	41.3	73.0	24.9	37.0	67.9
	FGWS	20.4	31.4	57.1	72.6	79.6	78.2	51.9	64.3	65.9	32.9	47.5	63.4
	MSP	12.8	21.1	70.4	79.2	83.8	91.2	69.1	77.2	88.3	18.3	28.6	62.6
	RDE	19.5	30.2	72.5	68.8	77.0	91.2	66.4	75.4	88.1	34.6	47.9	74.0
	DIST	17.7	26.1	70.1	64.9	68.1	91.4	69.7	77.3	88.4	29.5	42.3	62.9
	ADDMU	**51.4**	**64.1**	**84.6**	**83.7**	**85.9**	**94.1**	**76.3**	**81.9**	**90.6**	**34.9**	**48.4**	**76.0**
Pruthi	PPL	29.7	42.9	71.9	31.0	44.0	70.7	35.3	48.7	72.9	54.9	66.6	85.5
	MSP	53.2	65.2	82.6	75.7	81.9	91.5	65.4	74.7	88.7	22.5	33.9	69.2
	RDE	41.4	55.1	80.6	77.4	82.8	92.4	52.6	64.8	88.0	34.6	47.8	76.5
	DIST	55.0	61.4	82.9	77.8	82.0	92.1	66.7	72.2	88.2	23.6	35.2	65.1
	ADDMU	**55.9**	**67.4**	**85.4**	**96.7**	**93.9**	**97.4**	**78.8**	**83.7**	**91.8**	**55.7**	**67.1**	**86.0**
Pruthi-FB	PPL	28.6	41.6	72.3	27.8	40.4	71.6	37.3	50.8	73.3	37.2	50.6	76.3
	MSP	31.1	44.4	73.8	49.4	62.2	84.5	51.5	63.9	85.4	10.2	17.0	64.5
	RDE	20.0	30.8	72.6	59.5	70.4	87.6	34.3	47.9	85.2	31.2	44.2	74.9
	DIST	23.3	26.5	74.6	55.1	61.6	87.2	54.5	55.2	84.9	21.6	32.8	63.3
	ADDMU	**56.2**	**68.7**	**85.8**	**80.4**	**84.9**	**95.0**	**68.7**	**77.0**	**90.7**	**44.9**	**58.0**	**82.5**
TB	PPL	30.8	43.7	76.1	74.0	80.5	90.3	56.9	68.2	84.4	56.0	67.5	84.3
	MSP	72.3	79.0	90.5	95.6	93.0	97.3	70.4	78.1	89.8	66.4	75.1	89.0
	RDE	72.4	79.6	89.6	96.1	93.3	96.9	66.2	75.2	89.2	51.8	64.1	83.0
	DIST	72.4	78.6	90.6	95.6	92.8	96.2	70.2	77.9	90.2	50.7	62.7	82.6
	ADDMU	**73.3**	**80.0**	**90.9**	**99.0**	**94.8**	**98.4**	**70.8**	**78.3**	**91.0**	**69.0**	**77.1**	**90.6**
TB-FB	PPL	36.0	49.4	80.2	82.9	86.0	94.2	60.6	71.1	85.8	48.9	61.6	76.3
	MSP	34.8	48.2	83.0	81.1	84.9	91.2	70.0	77.8	88.4	34.7	48.0	81.5
	RDE	29.5	42.5	82.1	68.9	77.1	91.7	63.9	73.5	88.4	47.8	60.6	82.2
	DIST	34.3	44.0	82.6	63.4	72.9	91.5	69.8	77.6	89.3	40.8	53.9	79.0
	ADDMU	**50.5**	**62.9**	**86.1**	**94.2**	**92.6**	**96.9**	**74.8**	**81.0**	**90.8**	**51.1**	**63.6**	**87.0**

Table 3: Detection performance of regular and FB adversarial examples (*-FB) against BERT on SST-2, AGNews, Yelp, and SNLI. Our proposed ADDMU outperforms other methods by a large margin, especially on FB adversarial examples. We occlude FGWS under character-level attacks, Pruthi and TextBugger, as it is designed for word-level detection. The best performance is bolded. Results are averaged over three runs with different random seeds.

are discussed in the following.

We find that ADDMU, the aggregation of two uncertainties, achieves the best results in 70 out of the 96 metric scores. DU and MU are the best in 12 scores each. This shows that the combination of the two uncertainties provides more information to identify adversarial examples. We also observe that on SNLI, DU values are typically less useful, and thus the combination of DU and MU performs slightly worse than MU. One explanation is that the SNLI task requires more sophisticated neighborhood construction method to generate meaningful neighbors in data uncertainty estimation. Finally, we also notice that RDE-aug and DIST-aug are in general better than RDE and DIST, especially under FB attacks, which demonstrates the effectiveness of neighbor augmentation.

Why do detection results vary among datasets and attacks? Among different attacks, we find that Pruthi is the hardest to detect, followed by BAE. However, there is no obvious difference between detection performances against word-level and character-level attacks. Also, attacks on the sentence pair task (SNLI) are in general harder to

		AGNews			SNLI		
	Method	TPR	F1	AUC	TPR	F1	AUC
TF	RDE	96.0	93.2	97.0	46.3	59.3	81.0
	RDE-aug	97.4	94.0	97.4	41.0	54.3	79.9
	DIST	94.5	92.4	95.9	37.2	50.4	74.5
	DIST-aug	94.0	92.0	96.9	38.3	51.5	75.2
	MU	82.0	85.4	94.5	65.1	74.4	89.1
	DU	98.9	94.6	98.3	59.6	70.3	85.6
	ADDMU	**99.2**	**94.9**	**98.6**	**68.9**	**77.0**	**89.7**

Table 4: Ablation study on effect of uncertainty aggregation and neighbors augmentation against TextFooler.

	F1	Correct	Wrong
SST-2 TF	75.8	0.129	0.360
SST-2 BAE	58.9	0.136	0.597
Yelp TF	83.5	0.211	0.411
Yelp BAE	79.5	0.229	0.425

BAE attack on SST-2, ADDMU fails to detect	
Groundtruth Label changed : Positive → Negative	
Original	Most new movies have a bright sheen.
Attacked	Most new movies have a bad sheen.

Table 5: Why detector performance varies among attacks? This might because attacks already flip groundtruth labels of the examples. We show the detector performance (F1) and the proportion of adversarial examples that have their sentiments changed according to humans on correctly and wrongly detected sets.

detect. Thus, future work could focus more on improving the performance of detecting adversarial examples in sentence pair tasks, like SNLI.

We investigate why the detection performances vary among attacks. Our hypothesis is that attacks on some datasets fail to be imperceptible and have changed the groundtruth label for an input. Thus, these 'adversarial' (can not be called adversarial any more as they do not meet the definition of being imperceptible) examples actually lie close to the training manifold of the target class. Therefore, AED methods find it hard to detect those examples. To verify this assumption, we choose two tasks (SST-2 and Yelp) and two attacks (TF and BAE) to do sentiment analysis. We ask Amazon MTurk workers [3] to re-label *positive* or *negative* for attacked examples. Then, we summarize the proportion of examples that workers assign opposite groundtruth labels in correctly and wrongly detected groups. As shown in Table 5, there is an obvious correlation between bad performance and the number of 'adversarial' examples whose groundtruth labels changed. For example, ADDMU performs weak on detecting BAE attacks on SST-2 (58.9 F1), but it turns out that this is because more than half of the examples already have their groundtruth labels flipped. We give one example in Table 5. This shows that adversarial attacks need to be improved to retain the semantic meaning of the original input.

5 Characterize Adversarial Examples

In this section, we explore how to characterize adversarial examples by the two uncertainties.

MU-DU Data Map Plotting a heatmap with MU on X-axis and DU on Y-axis, we visualize data in terms of the two uncertainties. We show in Figure 2 the heatmaps with natural data, FB and regular adversarial examples generated from three attacks on three datasets (AGNews TF, Yelp BAE, SNLI Pruthi). The performance of ADDMU varies on the three attacks, as shown on the left of Figure 2.

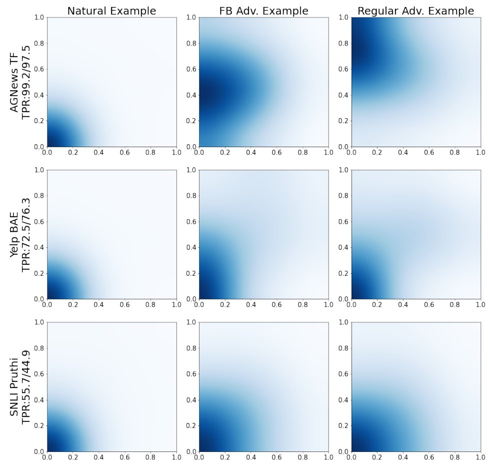

Figure 2: MU-DU heatmaps based on natural and regular/FB adversarial examples generated from three attacks. X-axis: MU value; Y-axis: DU value. Attack types and ADDMU performance are labeled on the left. TPR: Regular Adv./FB Adv.

We find that natural examples center on the bottom left corner of the map, representing low MU and DU values. This phenomenon does not vary across datasets. Whereas for FB and regular adversarial examples, they have larger values on at least one of the two uncertainties. When ADDMU performs best (AGNews TF, the first row), the center of adversarial examples in the MU-DU map is relatively rightward and upward compared to other cases. For maps on the third row, the shadow stretches along the MU axis, indicating that Pruthi examples on SNLI have relatively large MU values.

Identifying Informative ADA Data ADA is another adversarial defense technique, which aug-

[3] Also 0.05 dollar per HIT, but each HIT takes around 10 seconds to finish. Each HIT is assigned three workers.

SST-2 TF	**Clean %**	**#Aug**	**ASR**	**#Query**
BERT	92.8	0	94.31%	98.51
+ All	92.4	11199	87.36%	66.31
+ LDLM	91.7	2800	90.62%	108.06
+ HDLM	92.4	2800	88.59%	111.26
+ LDHM	**92.9**	2799	**85.05%**	115.30
+ HDHM	91.9	2799	87.07%	**119.92**

Table 6: ADA performances of different types of augmented data. We find that adversarial examples with low DU and high MU are most useful for ADA.

ments the training set with adversarial data and re-train the victim model to improve its robustness. In this part, we show that our ADDMU provides information to select adversarial data that is more beneficial to model robustness. We test it with TF on SST-2. The procedure is as follows: since SST-2 only has public training and validation sets, we split the original training set into training (80%) and validation set (20%), and use the original validation set as test set. We first train a model on the new training set. Then, we attack the model on validation data and compute DU and MU values for each adversarial sample. We sort the adversarial examples according to their DU and MU values and split them by half into four disjoint sets: **HDHM** (high DU, high MU), **HDLM** (high DU, low MU), **LDHM** (low DU, high MU), and **LDLM** (low DU, low MU). We augment the clean training set with each of these sets and retrain the model. As a baseline, we also test the performance of augmenting with all the adversarial examples generated from the validation set (All). We report clean accuracy (**Clean %**), the number of augmented data (**#Aug**), attack success rate (**ASR**), and the average query number (**#Query**) for each model.

The results are in Table 6. We find that the most helpful adversarial examples are with *low DU* and *high MU*. Using those samples, we achieve better ASR and clean accuracy than augmenting with the whole validation set of adversarial examples, with only one quarter of the amount of data. It is expected that examples with low DU and low MU are less helpful as they are more similar to the clean data. Similar observations are found in the FB version of TF attacks. We also compare augmentations with regular and FB adversarial examples. See details in Appendix D.

6 Related Work

Adversarial Detection. Adversarial examples detection has been well-studied in the image domain (Feinman et al., 2017; Lee et al., 2018; Ma et al., 2018; Xu et al., 2018; Roth et al., 2019; Li et al., 2021a; Raghuram et al., 2021). Our work aligns with Feinman et al. (2017); Li et al. (2021a); Roth et al. (2019) that introduce uncertainty estimation or perturbations as features to detect adversarial examples. We postpone the details to Appendix G, but focus more on the AED in NLP domain.

In the NLP domain, there are less work exploring AED. Zhou et al. (2019) propose DISP that learns a BERT-based discriminator to defend against adversarial examples. Mozes et al. (2021) propose a word-level detector FGWS that leverages the model confidence drop when replacing infrequent words in the input with frequent ones and surpass DISP. Pruthi et al. (2019) combat character-level attacks with word-recognition models. More recently, Yoo et al. (2022) propose a robust density estimation baseline and a benchmark for evaluating AED methods. There are other works like Xie et al. (2022); Biju et al. (2022) that leverage other features or train a detector. We show limitations of these works on FB adversarial examples and propose our ADDMU that overcomes this limitation.

Other Defenses against Attacks. AED is a category of approaches to defending against adversarial attacks. Other methods are also considered. Jin et al. (2020); Yin et al. (2020); Si et al. (2021) do *ADA* that augments original training datasets with adversarial data for better robustness. Madry et al. (2018b); Miyato et al. (2017); Zhu et al. (2020); Zhou et al. (2020) conduct *adversarial training* which is formulated as a min-max problem. Recently, several works perform certified robustness defense with either interval bound propagation (Huang et al., 2019; Jia et al., 2019; Shi et al., 2020), or randomized smoothness (Ye et al., 2020). In this work, we connect our AED method with ADA by selecting more informative data to augment.

7 Conclusion

We proposed ADDMU, an uncertainty-based approach for both regular and FB AED. We began by showing that existing methods are significantly affected by FB attacks. Then, we show that ADDMU is minimally impacted by FB attacks and outperforms existing methods by a large margin. We further showed ADDMU characterizes adversarial data and provides information on how to select useful augmented data for improving robustness.

8 Limitations

We summarize the limitations of this paper in this section.

1. We only test the AED methods under classification tasks. This is because we find that the attacks on other tasks like language generation are not well-defined, for example what would be the goal function of attacks on a language generation task? Is minimizing the BLEU score sufficient? It is hard to conduct detection when there is no standard for a valid adversarial example. Future work might come up with attacks for diverse tasks first and propose corresponding AED methods.

2. More experiments should be conducted to analyze the FB adversarial examples, including its characteristics and the security concerns it imposes to DNNs. However, given the time and space limitations, we are not able to do that.

3. Our method has slightly more hyperparameters to tune (four in total), and requires a bit more time to finish one detection. But, we confirm that it is in an acceptable range.

References

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2890–2896.

Anish Athalye, Nicholas Carlini, and David A. Wagner. 2018. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *ICML*.

Emil Biju, Anirudh Sriram, Pratyush Kumar, and Mitesh Khapra. 2022. Input-specific attention subnetworks for adversarial detection. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 31–44, Dublin, Ireland. Association for Computational Linguistics.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder for English. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 169–174.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. 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 4171–4186.

Reuben Feinman, Ryan R. Curtin, Saurabh Shintre, and Andrew B. Gardner. 2017. Detecting adversarial samples from artifacts. *ArXiv*, abs/1703.00410.

Ronald Aylmer Fisher. 1992. Statistical methods for research workers. In *Breakthroughs in statistics*, pages 66–70. Springer.

Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. *ArXiv*, abs/1506.02142.

Siddhant Garg and Goutham Ramakrishnan. 2020. BAE: BERT-based adversarial examples for text classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6174–6181.

Dan Hendrycks and Kevin Gimpel. 2017. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *ArXiv*, abs/1610.02136.

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. *ArXiv*, abs/1909.01492.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. *ArXiv*, abs/1707.07328.

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.

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*.

Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. 2018. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *NeurIPS*.

Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, Joe Davison, Mario Šaško, Gunjan Chhablani, Bhavitvya Malik, Simon Brandeis, Teven Le Scao, Victor Sanh, Canwen Xu, Nicolas Patry, Angelina McMillan-Major, Philipp Schmid, Sylvain Gugger, Clément Delangue, Théo Matussière, Lysandre Debut, Stas Bekman, Pierric Cistac, Thibault Goehringer, Victor Mustar, François Lagunas, Alexander Rush, and Thomas Wolf. 2021. Datasets: A community library for natural language processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 175–184, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. 2019. Textbugger: Generating adversarial text against real-world applications. *ArXiv*, abs/1812.05271.

Yao Li, Tongyi Tang, Cho-Jui Hsieh, and Thomas C. M. Lee. 2021a. Detecting adversarial examples with bayesian neural network. *ArXiv*, abs/2105.08620.

Zongyi Li, Jianhan Xu, Jiehang Zeng, Linyang Li, Xiaoqing Zheng, Qi Zhang, Kai-Wei Chang, and Cho-Jui Hsieh. 2021b. Searching for an effiective defender: Benchmarking defense against adversarial word substitution. In *EMNLP*.

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. *ArXiv*, abs/1907.11692.

Xingjun Ma, Bo Li, Yisen Wang, Sarah Monazam Erfani, Sudanthi N. R. Wijewickrema, Michael E. Houle, Grant Robert Schoenebeck, Dawn Xiaodong Song, and James Bailey. 2018. Characterizing adversarial subspaces using local intrinsic dimensionality. *ArXiv*, abs/1801.02613.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018a. Towards deep learning models resistant to adversarial attacks. *ArXiv*, abs/1706.06083.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018b. Towards deep learning models resistant to adversarial attacks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Takeru Miyato, Andrew M. Dai, and Ian J. Goodfellow. 2017. Adversarial training methods for semi-supervised text classification. *arXiv: Machine Learning*.

John Morris, Jin Yong Yoo, and Yanjun Qi. 2020a. TextAttack: Lessons learned in designing python frameworks for NLP. In *Proceedings of Second Workshop for NLP Open Source Software (NLP-OSS)*, pages 126–131.

John X. Morris, Eli Lifland, Jack Lanchantin, Yangfeng Ji, and Yanjun Qi. 2020b. Reevaluating adversarial examples in natural language. *ArXiv*, abs/2004.14174.

Maximilian Mozes, Pontus Stenetorp, Bennett Kleinberg, and Lewis Griffin. 2021. Frequency-guided word substitutions for detecting textual adversarial examples. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 171–186, Online. Association for Computational Linguistics.

Daniel Naber et al. 2003. A rule-based style and grammar checker.

Danish Pruthi, Bhuwan Dhingra, and Zachary C. Lipton. 2019. Combating adversarial misspellings with robust word recognition. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5582–5591.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Jayaram Raghuram, Varun Chandrasekaran, Somesh Jha, and Suman Banerjee. 2021. A general framework for detecting anomalous inputs to dnn classifiers. In *ICML*.

Kevin Roth, Yannic Kilcher, and Thomas Hofmann. 2019. The odds are odd: A statistical test for detecting adversarial examples. In *ICML*.

Peter J. Rousseeuw. 1984. Least median of squares regression. *Journal of the American Statistical Association*, 79:871–880.

Bernhard Schölkopf, Alex Smola, and Klaus-Robert Müller. 1998. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319.

Artem Shelmanov, Evgenii Tsymbalov, Dmitri Puzyrev, K. Fedyanin, Alexander Panchenko, and Maxim Panov. 2021. How certain is your transformer? In *EACL*.

Zhouxing Shi, Huan Zhang, Kai-Wei Chang, Minlie Huang, and Cho-Jui Hsieh. 2020. Robustness verification for transformers. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

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*.

Lewis Smith and Yarin Gal. 2018. Understanding measures of uncertainty for adversarial example detection. *ArXiv*, abs/1803.08533.

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.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15:1929–1958.

Zhouhang Xie, Jonathan Brophy, Adam Noack, Wencong You, Kalyani Asthana, Carter Perkins, Sabrina Reis, Sameer Singh, and Daniel Lowd. 2022. Identifying adversarial attacks on text classifiers. *ArXiv*, abs/2201.08555.

Weilin Xu, David Evans, and Yanjun Qi. 2018. Feature squeezing: Detecting adversarial examples in deep neural networks. *ArXiv*, abs/1704.01155.

Mao Ye, Chengyue Gong, and Qiang Liu. 2020. Safer: A structure-free approach for certified robustness to adversarial word substitutions. In *ACL*.

Fan Yin, Quanyu Long, Tao Meng, and Kai-Wei Chang. 2020. On the robustness of language encoders against grammatical errors. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3386–3403.

KiYoon Yoo, Jangho Kim, Jiho Jang, and Nojun Kwak. 2022. Detection of adversarial examples in text classification: Benchmark and baseline via robust density estimation. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3656–3672, Dublin, Ireland. Association for Computational Linguistics.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 1*, pages 649–657.

Yi Zhou, Xiaoqing Zheng, Cho-Jui Hsieh, Kai-Wei Chang, and Xuanjing Huang. 2020. Defense against adversarial attacks in nlp via dirichlet neighborhood ensemble. *ArXiv*, abs/2006.11627.

Yichao Zhou, Jyun-Yu Jiang, Kai-Wei Chang, and Wei Wang. 2019. Learning to discriminate perturbations for blocking adversarial attacks in text classification. 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)*.

Chen Zhu, Yu Cheng, Zhe Gan, S. Sun, Tom Goldstein, and Jingjing Liu. 2020. Freelb: Enhanced adversarial training for natural language understanding. *arXiv: Computation and Language*.

A Experimental Setup Details

A.1 Datasets and Target Models

We conduct experiments on four datasets, SST-2, Yelp-Polarity, AGNews, and SNLI. Statistics about those datasets are summarized on Table 7. All those datasets are available at Huggingface Datasets (Lhoest et al., 2021). Our target models are BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019). We use the public accessible BERT-base-uncased and RoBERTa-base models fine-tuned on the above datasets provided by TextAttack (Morris et al., 2020a) to benefit reproducibility. The performance of those models are summarized on Table 8

Dataset	Train/Dev/Test	Avg Len	#Labels
SST-2	67.3k/0.8k/-	19.2	2
Yelp	560k/-/38k	152.8	2
AGNews	120k/-/7.6k	35.5	4
SNLI	550k/10k/20k	8.3	3

Table 7: Data Statistics of the four datasets.

Dataset	SST-2	Yelp	AGNews	SNLI
BERT	92.43	96.30	94.20	89.40
RoBERTa	94.04	-	94.70	-

Table 8: BERT-base-uncased and RoBERTa-base accuracy on the four datasets. TextAttack does not have public model for RoBERTa fine-tuned on Yelp and SNLI.

A.2 Attacks and Statistics

We consider four attacks. TextFooler (Jin et al., 2020), BAE (Garg and Ramakrishnan, 2020), Pruthi (Pruthi et al., 2019), and TextBugger (Li et al., 2019). TextFooler and BAE are word-level attacks. Pruthi and TextBugger are character-level attacks. For BAE, we use BAE-R, i.e., replace a word with a substitution. For attacks on SNLI, we only perturb the hypothesis sentence. For FB attacks, as stated in the main paper, we add another goal function to make sure the softmax probability of the attacked class is larger than a threshold ϵ. We select $\epsilon = 0.9$ for SST-2, Yelp, AGNews, and $\epsilon = 0.7$ for SNLI. We implement those attacks with TextAttack, with the default hyperparameter settings. Please refer to the document of TextAttack for details. Here we report the after-attack accuracy (**Adv. Acc**), the attack success rate (**ASR**), the number of queries (**#Query**), and the number of adversarial examples we select (**#Adv**) for each attack on each dataset, as well as for FB attacks. Notice, the total evaluted examples will be twice the number of adversarial examples. See Table 9 and Table 10.

TextFooler	Adv. Acc	ASR%	#Query	#Adv
SST-2	4.5	95.1	95.3	1290
Yelp	6.0	93.8	475.7	738
AGNews	17.7	81.4	333.5	1625
SNLI	3.0	96.7	58.5	2222
BAE	Adv. Acc	ASR%	#Query	#Adv
SST-2	38.3	58.9	60.8	412
Yelp	44.9	53.7	319.9	1039
AGNews	81.5	14.3	122.5	278
SNLI	32.5	64.0	43.4	1605
Pruthi	Adv. Acc	ASR%	#Query	#Adv
SST-2	59.2	36.0	326.9	111
Yelp	86.4	11.5	1678.1	1036
AGNews	84.5	11.1	792.0	239
SNLI	23.2	74.4	103.4	1846
TextBugger	Adv. Acc	ASR%	#Query	#Adv
SST-2	28.9	68.7	49.3	221
Yelp	16.3	83.3	350.1	738
AGNews	20.2	79.2	123.4	1088
SNLI	4.5	95.0	41.9	2225

Table 9: Statistics about attacks. We report the adversarial accuracy (Adv. Acc), attack success rate (ASR%), the number of queries (#Query), and the number of adversarial examples examined.

B DIST

We propose the **DIST** baseline, which is a distance-based detector motivated by Ma et al. (2018). We also find that the Local Intrinsic Dimension value proposed in Ma et al. (2018) does not work well when detecting NLP attacks. The DIST method leverages the whole training set as $\mathcal{D}_{aux}$. Then, it selects the K-nearest neighbors of the evaluated point from each class of $\mathcal{D}_{aux}$ and calculates the average distance between the neighbors and the evaluated point, denote as $d_1, d_2, \cdots, d_k$, where k is the number of classes. Suppose the evaluated point has predicted class i. Then, it uses the difference between the distance of class i and the minimum of the other classes to do detection. i.e., $d_i - min\,(d_k)$, where $k \neq i$. The intuition is that since adversarial examples are generated from the original class, they might still be closer to training data in the original class, which is $min\,(d_k)\,, k \neq i$.

TF-FB	Adv. Acc	ASR%	#Query	#Adv
SST-2	6.54	94.8	108.4	295
Yelp	6.2	93.7	496.0	1027
AGNews	22.0	77.4	365.7	1604
SNLI	8.3	91.4	69.6	2068
BAE-FB	Adv. Acc	ASR%	#Query	#Adv
SST-2	45.3	52.2	64.3	164
Yelp	50.2	48.8	323.4	333
AGNews	87.6	9.7	119.5	202
SNLI	46.8	51.3	44.5	1347
Pruthi-FB	Adv. Acc	ASR%	#Query	#Adv
SST-2	68.9	27.3	326.4	90
Yelp	89.4	9.1	1681.0	134
AGNews	89.8	7.4	791.4	158
SNLI	47.2	50.9	103.8	1323
TB-FB	Adv. Acc	ASR%	#Query	#Adv
SST-2	35.3	62.6	53.0	207
Yelp	18.4	81.3	369.4	1025
AGNews	53.1	45.3	191.1	948
SNLI	18.1	81.2	50.1	2093

Table 10: Statistics about FB attacks. We report the adversarial accuracy (Adv. Acc), attack success rate (ASR%), the number of queries (#Query), and the number of adversarial examples examined.

C Implementation Details

For DIST and ADDMU, we tune the hyperparameters with an attacked validation set. For datasets with an original train/validation/test split (SNLI), we simply attacked the examples in the validation set and select 100 of them to help the tuning. For datasets without an original split, like SST-2, Yelp, and AGNews, we randomly held out 100 examples from the training set and attack them to construct a set for hyperparameter tuning. For DIST, we select the number of the neighbors from {100, 200, $\cdots$, 1000}. For ADDMU, we select N_m and N_d from {10, 20, 80, 100}, and choose p_m and p_d from {0.1, 0.2, 0.3, 0.4}. In our preliminary experiment, we find that ensemble different p_d values also help. So, we also consider ensemble different p_d values in combinations {(0.1, 0.2), (0.1, 0.2, 0.3, 0.4)}. We also find that augment the model uncertainty estimation with some neighborhood data is helpful, so for the model uncertainty value, we actually average over 10 neighborhood data with 0.1 mask rate.

D Selecting Useful data with Uncertainty values

In this section, we present the results of selecting useful data for ADA using DU and MU values for FB version of TF, shown in Table 11. Similar to the regular version, we find that the most useful data are still those with low data uncertainty and high model uncertainty. We achieve better ASR and the number of queries using only one quarter of data compared to the full augmentation. In Table 12, we show the attack success rate of four settings. 1) Augment with FB examples to defend against regular attack; 2) Augment with FB examples to defend against FB attack; 3) Augment with regular examples to defend against regular attack; 4) Augment with regular examples to defend against FB attack. The finding is that augment with FB and regular adversarial examples most benefits its own attacks. This implies that FB attacks might already change the characteristics of regular attacks. We need to defend against them with different strategies.

SST-2 TF	**Clean %**	**#Aug**	**ASR**	**#Query**
BERT	95.8	0	88.66%	118.99
+ All	95.6	11199	77.58%	140.74
+ LDLM	95.6	2800	82.52%	137.50
+ HDLM	95.8	2800	78.25%	142.26
+ LDHM	**95.8**	2799	**75.30%**	**145.79**
+ HDHM	95.3	2799	77.67%	142.42

Table 11: ADA performances for FB version of different types of augmented data. We find that adversarial examples with low DU and high MU are most useful for ADA.

	Regular	FB
Regular	87.2	90.2
FB	82.3	77.1

Table 12: Attack success rate for four settings of augmentation. The columns are the augmented data. The rows are the attack types.

E RoBERTa Results

We conduct adversarial examples detection with RoBERTa-base. The setting is the same as BERT. Through hyperparameters search as described before, for ADDMU, we select $N_m = 20$ and $N_d = 100$, and choose $p_m = 0.1$ and $p_d = 0.1$, without augmentation for MU estimation and no ensemble of various p_d. Table 14 presents the results for RoBERTa-base. ADDMU also outperform other methods with RoBERTa. We combine the ablation table together with the main table for RoBERTa.

F Ablation Study

We present the full results for the ablation study of uncertainty aggregation in Table 13. We also show that our neighborhood construction process in data uncertainty can be used to enhance two baselines RDE and DIST.

G Related work in CV

Feinman et al. (2017) train a binary classifier using density estimation and Bayesian uncertainty estimation as features for detection. Li et al. (2021a) replace DNNs with Bayesian Neural Networks, which enhance the distribution dispersion between natural and adversarial examples and benefit AED. Roth et al. (2019) use *logodds* on perturbed examples as statistics to conduct detection. Further, Athalye et al. (2018) have similar observations with us concerning image attacks. They find that the distance-based feature, *local intrinsic dimension* proposed in Ma et al. (2018) for AED fails when encounters FB adversarial examples.

		SST-2			AGNews			Yelp			SNLI		
Attacks	**Methods**	**TPR**	**F1**	**AUC**	**TPR**	**F1**	**AUC**	**TPR**	**F1**	**AUC**	**TPR**	**F1**	**AUC**
TF	RDE	62.9	72.8	86.5	96.0	93.2	97.0	72.0	79.2	89.6	46.3	59.3	81.0
	RDE-aug	63.6	73.3	86.6	97.4	94.0	97.4	70.1	77.9	89.8	41.0	54.3	79.9
	DIST	64.0	73.4	87.9	94.5	92.4	95.9	73.8	80.3	90.6	37.2	50.4	74.5
	DIST-aug	60.2	70.5	86.5	94.0	92.0	96.9	75.7	81.4	90.8	38.3	51.5	75.2
	MU	51.9	64.2	85.9	82.0	85.4	94.5	71.7	79.0	90.1	65.1	74.4	89.1
	DU	60.6	71.1	87.8	98.9	94.6	98.3	76.3	82.0	90.6	59.6	70.3	85.6
	ADDMU	**67.1**	**75.8**	**88.8**	**99.2**	**94.9**	**98.6**	**78.7**	**83.5**	**91.6**	**68.9**	**77.0**	**89.7**
TF-FB	RDE	31.9	45.0	81.5	71.9	79.1	92.5	31.5	44.6	82.7	43.1	56.4	79.6
	RDE-aug	36.6	50.2	80.4	90.8	90.5	95.9	61.5	71.8	87.8	37.6	51.0	78.9
	DIST	20.7	26.3	81.6	66.6	75.4	91.8	54.8	64.3	86.2	27.2	39.6	69.9
	DIST-aug	50.5	62.4	84.0	81.9	85.3	94.5	64.0	73.5	88.5	29.6	42.3	71.0
	MU	54.4	66.3	85.4	90.7	90.4	96.5	70.7	78.3	89.1	**60.2**	**70.8**	87.0
	DU	55.4	67.1	84.5	97.2	93.9	97.5	70.1	77.9	88.4	31.6	44.6	78.2
	ADDMU	**59.4**	**70.6**	**87.3**	**97.5**	**94.0**	**97.8**	**72.8**	**79.7**	**89.7**	53.6	65.8	**87.5**
BAE	RDE	44.2	57.3	79.3	96.4	93.7	96.3	65.2	74.5	89.1	41.7	55.0	76.8
	RDE-aug	**49.3**	**61.9**	82.4	85.6	87.8	94.3	61.7	71.9	88.5	44.9	57.9	80.3
	DIST	44.9	57.3	78.9	94.2	91.9	96.2	68.0	76.2	89.4	36.8	49.7	67.9
	DIST-aug	38.1	50.7	77.8	86.3	87.9	94.7	66.1	74.8	89.7	38.3	51.6	69.8
	MU	41.7	55.0	78.8	86.7	88.3	94.1	64.6	74.1	88.6	44.4	57.5	76.9
	DU	45.9	58.9	**83.3**	**97.5**	**94.3**	**98.1**	71.5	78.5	89.7	44.7	57.8	80.5
	ADDMU	45.9	58.9	82.3	96.4	93.5	97.3	**72.5**	**79.5**	**90.1**	**48.2**	**61.0**	**81.0**
BAE-FB	RDE	19.5	30.2	72.5	68.8	77.0	91.2	66.4	75.4	88.1	34.6	47.9	74.0
	RDE-aug	48.2	61.0	82.6	63.4	73.4	91.1	66.1	75.1	88.9	40.8	54.2	79.4
	DIST	17.7	26.1	70.1	64.9	68.1	91.4	69.7	77.3	88.4	29.5	42.3	62.9
	DIST-aug	28.7	40.0	72.4	70.3	76.3	91.6	71.5	78.0	89.8	31.5	44.0	65.4
	MU	49.7	62.3	82.3	83.7	86.4	94.0	74.5	80.8	89.9	**36.5**	**49.9**	73.1
	DU	**56.4**	**67.8**	84.4	**84.7**	**87.0**	93.4	74.5	80.8	90.2	22.9	34.5	74.3
	ADDMU	51.4	64.1	**84.6**	83.7	85.9	**94.1**	**76.3**	**81.9**	**90.6**	34.9	48.4	**76.0**
Pruthi	RDE	41.4	55.1	80.6	77.4	82.8	92.4	52.6	64.8	88.0	34.6	47.8	76.5
	RDE-aug	40.5	53.9	78.9	87.4	88.7	94.1	64.7	74.3	88.0	35.4	48.7	77.3
	DIST	55.0	61.4	82.9	77.8	82.0	92.1	66.7	72.2	88.2	23.6	35.2	65.1
	DIST-aug	50.5	61.4	84.1	81.2	84.6	94.1	69.2	75.6	89.5	26.4	38.7	67.4
	MU	48.6	61.4	85.3	89.5	89.9	95.5	77.5	83.1	90.7	**61.8**	**72.0**	**86.8**
	DU	55.7	66.8	82.7	95.8	93.8	97.3	72.4	79.6	88.8	26.6	39.0	74.4
	ADDMU	**55.9**	**67.4**	**85.4**	**96.7**	**93.9**	**97.4**	**78.8**	**83.7**	**91.8**	55.7	67.1	86.0
Pruthi-FB	RDE	20.0	30.8	72.6	59.5	70.4	87.6	34.3	47.9	85.2	31.2	44.2	74.9
	RDE-aug	26.7	39.0	74.5	67.7	76.4	91.8	60.4	71.1	87.0	31.0	44.0	76.0
	DIST	23.3	26.5	74.6	55.1	61.6	87.2	54.5	55.2	84.9	21.6	32.8	63.3
	DIST-aug	25.6	35.0	76.0	69.6	76.3	91.3	59.7	69.6	87.5	23.8	35.4	65.7
	MU	56.2	67.7	85.2	80.3	84.8	94.5	67.9	76.8	91.7	**60.7**	**71.1**	**85.5**
	DU	56.2	68.5	83.1	79.1	83.9	93.5	67.2	75.9	86.4	13.9	22.4	70.3
	ADDMU	**56.2**	**68.7**	**85.8**	**80.4**	**84.9**	**95.0**	**68.7**	**77.0**	**91.7**	44.9	58.0	82.5
TB	RDE	72.4	79.6	89.6	96.1	93.3	96.9	66.2	75.2	89.2	51.8	64.1	83.0
	RDE-aug	54.3	66.1	85.0	95.6	93.0	96.9	61.7	71.9	87.8	45.9	58.9	80.9
	DIST	72.4	78.6	90.6	95.6	92.8	96.2	70.2	77.9	90.2	50.7	62.7	82.6
	DIST-aug	72.9	79.1	89.7	93.0	91.6	96.3	70.5	78.0	90.5	52.0	64.2	83.1
	MU	67.4	76.0	88.9	79.8	84.1	94.5	67.0	75.7	88.9	60.2	70.8	88.6
	DU	**77.8**	**82.9**	90.2	98.4	94.7	98.0	69.3	77.3	89.2	66.9	75.4	88.9
	ADDMU	73.3	80.0	**90.9**	**99.0**	**94.8**	**98.4**	**70.8**	**78.3**	**91.0**	**69.0**	**77.1**	**90.6**
TB-FB	RDE	29.5	42.5	82.1	68.9	77.1	91.7	63.9	73.5	88.4	47.8	60.6	82.2
	RDE-aug	42.0	55.4	80.2	86.6	88.2	94.7	59.6	70.3	87.5	40.7	54.0	80.1
	DIST	34.3	44.0	82.6	63.4	72.9	91.5	69.8	77.6	89.3	40.8	53.9	79.0
	DIST-aug	49.8	59.0	84.6	80.4	84.3	93.6	71.8	78.9	90.4	43.9	57.0	79.8
	MU	55.9	67.4	85.8	91.8	91.0	96.1	72.2	79.4	89.6	**57.7**	**68.8**	87.0
	DU	**58.1**	**69.2**	85.0	94.1	92.2	96.5	72.7	79.6	89.2	40.9	54.2	81.5
	ADDMU	50.5	62.9	**86.1**	**94.2**	**92.6**	**96.9**	**74.8**	**81.0**	**90.8**	51.1	63.6	**87.0**

Table 13: Ablation of detection performance of regular and FB adversarial examples (*-FB) against BERT on SST-2, AGNews, Yelp, and SNLI. We compare ADDMU with soley DU, solely MU, and two enhanced baselines RDE-aug and DIST-aug. The best performance is bolded. Results are averaged over three runs with different random seeds.

		SST-2			AGNews		
Attacks	**Methods**	**TPR**	**F1**	**AUC**	**TPR**	**F1**	**AUC**
TF	PPL	34.0	47.2	73.7	78.2	83.1	92.0
	MSP	71.0	78.5	89.8	93.5	91.9	97.2
	RDE	73.9	80.4	89.8	90.6	90.4	95.5
	RDE-aug	61.3	71.6	87.1	61.7	71.9	87.9
	DIST	70.3	77.9	90.2	94.6	92.5	96.5
	DIST-aug	72.7	78.8	90.1	83.8	86.4	94.6
	MU	78.0	82.9	91.1	98.7	94.6	97.6
	DU	70.1	79.2	89.5	95.9	93.2	97.6
	ADDMU	**78.4**	**83.9**	**91.3**	**98.8**	**94.9**	**98.3**
TF-FB	PPL	43.8	57.0	79.6	84.4	86.9	94.1
	MSP	55.3	66.9	85.0	30.5	43.5	87.6
	RDE	40.5	53.8	84.6	57.0	68.3	88.7
	RDE-aug	46.7	59.7	82.4	48.1	60.9	81.9
	DIST	48.7	58.2	85.1	47.0	59.7	89.8
	DIST-aug	48.7	60.8	85.8	67.4	75.4	90.9
	MU	**55.9**	**66.9**	**89.2**	77.4	82.6	93.5
	DU	55.1	64.2	84.5	88.4	**89.6**	95.7
	ADDMU	54.6	66.8	88.5	**88.6**	89.2	**95.8**
BAE	PPL	17.2	27.1	64.0	38.1	51.5	74.0
	MSP	48.1	60.9	78.6	93.4	91.9	97.2
	RDE	53.8	65.7	80.3	77.2	82.5	93.2
	RDE-aug	53.5	65.5	84.4	52.9	64.9	82.6
	DIST	48.1	60.3	79.7	88.3	88.9	95.0
	DIST-aug	48.5	61.2	80.9	72.4	79.0	91.3
	MU	55.7	66.9	81.8	93.7	92.0	95.9
	DU	52.4	64.0	84.6	92.2	91.2	96.2
	ADDMU	**55.8**	**67.0**	**84.9**	**97.6**	**94.1**	**97.9**
BAE-FB	PPL	27.0	39.6	68.7	34.5	47.8	73.6
	MSP	31.4	44.6	69.8	77.0	82.4	89.8
	RDE	25.8	38.1	72.5	57.0	68.3	89.4
	RDE-aug	40.3	53.8	77.9	61.6	71.9	89.7
	DIST	25.8	31.2	71.2	36.0	47.2	89.9
	DIST-aug	30.8	42.7	73.8	62.0	70.2	89.5
	MU	43.4	56.2	76.5	92.0	91.3	95.0
	DU	37.7	51.3	78.1	79.5	86.9	93.3
	ADDMU	**44.4**	**57.1**	**78.3**	**92.0**	**91.9**	**95.7**
Pruthi	PPL	34.0	47.6	74.4	31.4	44.4	73.9
	MSP	62.0	72.1	83.1	70.2	78.0	93.0
	RDE	57.0	68.3	83.3	61.6	71.9	89.7
	RDE-aug	52.0	64.2	84.4	40.8	54.2	81.1
	DIST	63.0	70.6	83.1	65.5	74.6	92.5
	DIST-aug	52.0	64.6	84.3	72.2	77.5	91.1
	MU	73.0	77.1	**89.3**	**94.5**	**92.5**	97.5
	DU	58.0	68.3	84.9	85.1	87.3	95.5
	ADDMU	**77.0**	**82.4**	88.0	92.5	91.5	**97.6**
Pruthi-FB	PPL	23.4	35.3	71.0	27.9	40.6	71.5
	MSP	40.6	54.2	76.3	12.5	20.5	83.5
	RDE	51.6	64.1	78.4	35.3	48.7	82.1
	RDE-aug	37.5	51.1	75.7	26.5	39.1	74.6
	DIST	43.8	22.8	77.3	25.5	34.4	83.4
	DIST-aug	40.6	47.8	79.6	56.6	67.0	86.1
	MU	64.1	62.9	84.9	72.8	78.4	92.8
	DU	39.1	51.1	81.1	61.8	71.6	90.9
	ADDMU	**64.1**	**74.5**	**89.1**	**75.7**	**82.1**	**93.4**
TB	PPL	45.5	58.6	81.2	76.7	82.2	91.2
	MSP	74.7	81.1	91.8	91.4	90.8	96.8
	RDE	76.8	82.4	92.0	86.0	87.8	84.5
	RDE-aug	60.6	71.2	88.4	57.4	68.6	85.6
	DIST	76.3	80.7	91.5	93.4	91.7	96.0
	DIST-aug	77.3	80.1	91.8	80.1	84.0	93.9
	MU	78.3	82.3	92.4	98.3	94.4	97.3
	DU	74.2	80.4	90.7	94.0	92.1	97.0
	ADDMU	**78.8**	**82.9**	**92.4**	**98.3**	**94.5**	**97.9**
TB-FB	PPL	42.7	55.9	81.6	78.9	83.7	92.6
	MSP	57.5	69.4	86.3	29.8	42.7	87.6
	RDE	52.0	64.3	86.2	47.7	60.6	87.3
	RDE-aug	45.6	58.6	81.8	43.5	56.8	78.3
	DIST	48.5	56.5	86.1	45.2	58.0	89.4
	DIST-aug	48.0	59.7	85.5	60.0	70.6	89.4
	MU	58.5	70.0	**89.7**	84.2	86.8	95.0
	DU	53.9	64.2	84.6	81.9	84.5	92.8
	ADDMU	**64.9**	**74.2**	87.7	**85.5**	**88.3**	**96.8**

Table 14: Detection performance of regular and FB adversarial examples (*-FB) against RoBERTa on SST-2, AGNews. Our proposed ADDMU outperforms other methods. The best performance is bolded. Results are averaged over three runs with different random seeds.