Detection of Adversarial Examples in NLP: Benchmark and Baseline via Robust Density Estimation

Anonymous ACL submission

Abstract

Word-level adversarial attacks have shown success in NLP models, drastically decreasing the performance of transformer-based models in recent years. As a counter measure, adversarial defense has been explored, but relatively little efforts have been made to detect adversarial examples. However, detecting adversarial examples in NLP may be crucial for automated task (e.g. review sentiment analysis) that wishes to amass information about a certain population and additionally be a step towards a robust defense system. To this end, we release a dataset for four popular attack methods on three datasets and four NLP models to encourage further research in this field. Along with it, we propose a competitive baseline based on density estimation that has the highest AUC on 21 out of 22 dataset-attack-model combinations.[1]

1 Introduction

Adversarial examples in NLP refer to seemingly innocent texts that alter the model prediction to a desired output, yet remain imperceptible to humans. In recent years, adversarial attacks have shown success in NLP models, drastically decreasing the performance of transformer-based models in sentence classification tasks with increasingly smaller perturbation rate (Jin et al., 2020; Li et al., 2020; Garg and Ramakrishnan, 2020; Ren et al., 2019). In the image domain, two main lines of research exist to counteract adversarial attacks : adversarial example *detection* and *defense*. The goal of detection is to discriminate an adversarial input from a normal input, whereas adversarial defense intends to predict the correct output of the adversarial input. While works defending these attacks have shown some progress in NLP (Zhou et al., 2021; Keller et al., 2021; Jones et al., 2020), only few efforts have been made in detecting them.

[1]https://github.com/anoymous92874838/text-adv-detection

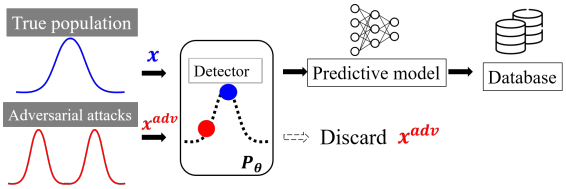

Figure 1: Schematic Diagram of our adversarial detection Framework. We propose a density estimation model to detect adversarial samples.

However, detecting adversarial examples in NLP may be as crucial as defending them by alerting the users to counteract them. In addition, models used for automation of tasks (e.g. review sentiment analysis, news headline classification, etc) are adopted to efficiently gain information about the true data-generating population (e.g. consumers, news media, etc), rather than the adversary. For such applications, attaining outputs of an adversarial input - whether correct or not - may turn out to be harmful for the system. Accordingly, the *discard-rather-than-correct strategy* which simply discards the detected adversarial input would be a good countermeasure. Moreover, being able to detect adversarial examples may be a step towards building a more robust defense model as the popular defense paradigm, adversarial training, usually suffers from degraded performance on normal inputs (Bao et al., 2021). With a competent detection system, the normal and adversarial inputs can be processed by two separate mechanisms as proposed by Zhou et al. (2019).

Existing few works of detecting adversarial examples in NLP (Le et al., 2021; Pruthi et al., 2019) either focus on a single type of attack or is limited to character-level attacks. Le et al. detect a particular type of attack that prepends an identical phrase on all samples, making them inapplicable to various other attack methods, and Pruthi et al. target detecting adversrial misspellings. Aforementioned methods detect adversarial attacks that do not consider either semantics or grammaticality,

SUCCESSFUL	director brian levant, who never strays far from his sitcom roots, skates blithely from one ~~implausible~~[***improbable***] situation to another [..]	😠 → 😀
FAILED	arnold 's ~~jump~~[***leap***] from little screen to big will leave frowns on more than a few faces	😠 → 😠

Table 1: Successful and Failed Adv. Examples in SST2 Dataset (~~original~~[***replaced***])

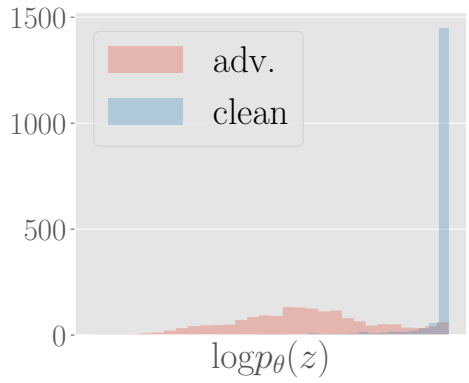

Figure 2: Density estimation using our method in (Data-Attack-Model) = (IMDB-(TF-adj)-BERT). Normal samples are peaked at high likelihood region. Adversarial samples tend to have low likelihood.

which are two key constraints in order to be imperceptible (Morris et al., 2020a). As opposed to this, carefully crafted word-level adversarial attacks can maintain original semantics and remain unsuspicious to human inspectors. Additionally, the works lack uniformity in the experimented attack methods and the experimental settings (Section 2.2). To this end, we release a benchmark for adversarial example detection on four attack methods across four NLP models and three datasets. We also propose a simple but effective baseline that utilizes density estimation in the feature space as shown in Fig. 1.

Inspired by classic works in novelty detection (Bishop, 1994), which utilizes generative models to find anomalies, we fit a parametric density estimation model to the features obtained from a classification model (e.g. BERT) to yield likelihoods of each sample as shown by Fig. 2. However, simply fitting a parametric model suffers from curse of dimensionality characterized by (i) sparse data points and spurious features (ii) and rare outliers that hamper accurate estimation. To tackle these issues, we leverage classical techniques in statistical analysis, namely kernel PCA and Minimum Covariance Determinant, for robust density estimation (RDE).

We validate our method with existing works in NLP and find that our method can be used as a competitive baseline across all the tested attack methods without accessing validation sets of each attack method. Our contributions are two-fold:

- We release a dataset for adversarial example detection on 4 attacks, 3 datasets, and 4 models and the source code for experimenting on various experimental protocols.
- We propose a competitive baseline method that does not require validation sets of each attack method through robust parameter estimation, alleviating problems caused by curse of dimensionality.
- Our method achieves best performance as of AUC on 21 out of 22 dataset-attack-model combinations and best performance as of TPR, F1 , AUC on 17 of them without any assumption on the attacks.

2 Preliminaries

2.1 Adversarial Examples

Given an input space $\mathcal{X}$, a label space $\mathcal{Y}$, a predictive model $\mathcal{F} : \mathcal{X} \to \mathcal{Y}$, and an oracle model $\mathcal{F}^* : \mathcal{X} \to \mathcal{Y}$ a successful adversarial example, x_{adv}, of an input $x \in \mathcal{X}$ satisfies the following:

$$\begin{aligned} \mathcal{F}^*(x) = \mathcal{F}(x) \neq \mathcal{F}(x_{\text{adv}}), \\ C_i(x, x_{\text{adv}}) = 1 \text{ for } i \in \{1, \ldots, c\} \end{aligned} \tag{1}$$

where C_i is an indicator function for the i-th constraint between the perturbed text and the original text, which is 1 when the two texts are indistinguishable with respect to the constraint. The constraints vary from attack algorithms and is crucial for maintaining the original semantics while providing an adequate search space. For instance, Jin et al. (2020) ensure that the embedding of the two sentences have a cosine similarity larger than 0.5 using the Universal Sentence Encoder (Cer et al., 2018, USE).

2.2 Detecting Adversarial Examples

For the purpose of detecting adversarial examples, a dataset, $\mathcal{D}$, consisting of clean samples ($\mathcal{D}_{\text{clean}}$)

Dataset	Task	Classes	Median Length	# of Test Samples Original / Generated	# of Val. Samples Orig. / Gen.
IMDB(Maas et al., 2011)	movie review sentiment classification	2	161	25K / 10K	0 / NA
AG-News(Zhang et al., 2015)	news topic classification	4	44	7.6K / 7.6K	0 / NA
SST-2(Socher et al., 2013)	sentiment classification	2	16	1.8K / 1.8K	0.87K / 0.87K

Table 2: Summary of the benchmark dataset. For datasets without separate validation sets, portion of the test samples could be held out for validation.

and adversarial samples ($\mathcal{D}_{\text{adv}}$) is required. However, how the dataset should be configured has rarely been discussed in detail and the exact implementation varies by works. Here we discuss two main configurations used in the literature. We denote the test set as $\mathcal{X}_t$ and the correctly classified test set as $\mathcal{X}_c \subset \mathcal{X}_t$.

- Scenario 1 : Sample disjoint subsets $\mathcal{S}_1, \mathcal{S}_2 \subset \mathcal{X}_t$. For the correctly classified examples of $\mathcal{S}_1$, adversarial attacks are generated and the successful examples form $\mathcal{D}_{\text{adv}}$. $\mathcal{D}_{\text{clean}}$ is formed from $\mathcal{S}_2$.
- Scenario 2 : Sample subset $\mathcal{S} \subset \mathcal{X}_t$. For the correctly classified examples of $\mathcal{S}$, adversarial attacks are generated and the successful examples form $\mathcal{D}_{\text{adv}}$. $\mathcal{D}_{\text{clean}}$ is formed from $\mathcal{S}$.

Scenario 1 provides more flexibility in choosing the ratio between adversarial samples and clean samples, while in Scenario 2 this is determined by the attack success rate and task accuracy. For instance, an attack with low success rate will have a low adversarial-to-clean sample ratio. In addition, Scenario 2 consists of pairs of adversarial sample and its corresponding clean sample in addition to the incorrect clean samples. A more challenging scenario can be proposed by including failed attacked samples, which may be closer to the real world. Examples of failed and successful samples are provided in Table 1.

A seminal work (Xu et al., 2018) on adversarial example detection in the image domain assumes the first scenario, whereas existing works in NLP (Le et al., 2021; Mozes et al., 2021) only experiment on the second scenario. Our benchmark framework provides the data and tools for experimenting on both. We provide experiment results on both scenarios.

3 Method

3.1 Benchmark

We generate adversarial examples on four models, four types of attacks, and three sentence classification datasets. Since some attacks (Garg and Ramakrishnan, 2020) require hundreds of queries and inference of models per query, vast amount of time is required to create all the adversarial examples (e.g. up to 44 hours for 5,000 examples on the IMDB dataset using TF-adjusted attack). This renders on-the-fly generation and detection of adversarial examples extremely inefficient. Therefore, adversarial examples are created beforehand and sampled according to Section 2.2. Three sentence classification datasets (IMDB, AG-News, SST-2) are chosen to have diverse topics and length. See Table 2 for the summary and the number of generated samples.

We choose two non-transformer-based models (Word-CNN Kim (2014); LSTM Hochreiter and Schmidhuber (1997)) and two transformer-based models (RoBERTa Liu et al. (2019); BERT Devlin et al. (2018)) . Recently, numerous adversarial attacks have been proposed. We choose two widely known attacks called Textfooler (Jin et al., 2020, TF) and Probability Weighted Word Saliency (Ren et al., 2019, PWWS) and a recent approach using BERT to generate attacks called BAE (Garg and Ramakrishnan, 2020). Lastly, we also include a variant of TF called TF-adjusted (Morris et al., 2020a, TF-adj), which enforces a stronger similarity constraint to ensure imperceptibility to humans. All attacks are created using the TextAttack library (Morris et al., 2020b). See Appendix A.1 for the summary of attack methods and Appendix A.5 for a code snippet of using our benchmark.

3.2 Estimating Density and Parameters in Feature Space

Earlier works in novelty detection (Bishop, 1994) have shown that generative models fitted on normal samples are capable of detecting unseen novel samples (e.g. adversarial samples). Since we can assume that the training samples, which were used to train the victim model of a particular task, are available to the victim party, we can similarly design a generative model that estimates input density. However, directly using the inputs is challeng-

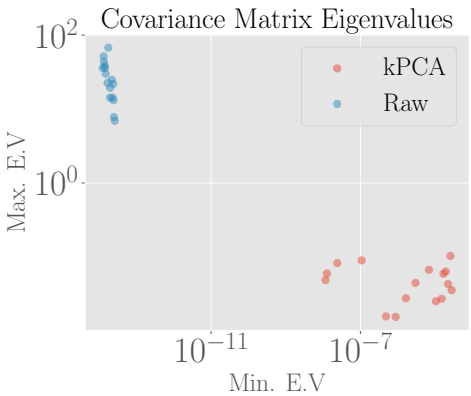

Figure 3: Comparisons of the maximum and minimum eigenvalues of the estimated covariance matrices for RoBERTa and BERT across all datasets and classes (20 samples total). Naive estimation (blue) using raw features leads to extremely ill-conditioned matrices while kPCA (red) alleviates this.

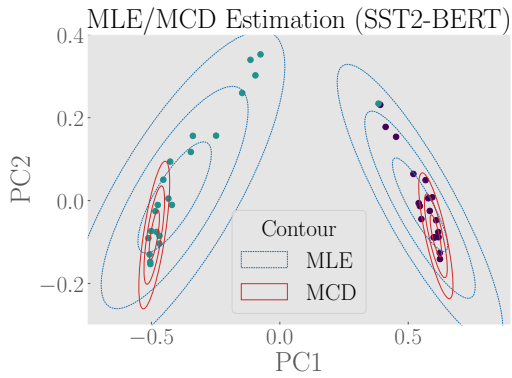

Figure 4: Probability-contours of $\tilde{\Sigma}$ within three standard deviations estimated by MLE and Minimum Covariance Determinant (MCD) of BERT on SST-2 dimensionality reduced by kPCA. Colors of the points indicate class. See Sec. 3.3 for details.

ing as modeling the probability distribution of raw texts is non-trivial. To bypass this, we fit a parametric density estimation model in the feature space (i.e. penultimate layer of the classification model). Since a neural network learns to extract important features of the inputs to distinguish classes, the features can be regarded as representations of the raw inputs. For a pre-trained predictive model $\mathcal{F}$, let $z \in \mathcal{Z} \subset \mathbb{R}^D$ denote the feature given by the feature extractor $\mathcal{H} : \mathcal{X} \to \mathcal{Z}$. Then the entire predictive model can be written as the composition of $\mathcal{H}$ and a linear classifier.

Given a generative model p_θ with mean and covariance as parameters $\theta = (\mu, \Sigma)$, we can use the features of the training samples ($\mathcal{X}_{train}$) to estimate the parameters. Then, novel adversarial samples lying in the unobserved feature space are likely to be assigned a low probability, because the generative model only used the normal samples for parameter fitting. For simplicity, we assume the distributions of the feature z follow a multivariate Gaussian, and thus we model the class conditional probability as $p_\theta(z|y=k) \sim N(\mu_k, \Sigma_k) \propto \exp\{-(z-\mu_k)^T \Sigma_k^{-1} (z-\mu_k)\}$, where y indicates the class of a given task. Then, the maximum likelihood estimate (MLE) is given by the sample mean $\tilde{\mu}_{\text{MLE}} = \frac{1}{N}\sum_{i=1}^{N} z_i$ and sample covariance $\tilde{\Sigma}_{\text{MLE}} = \frac{1}{N}\sum_{i=1}^{N}(z_i - \tilde{\mu}_{\text{MLE}})(z_i - \tilde{\mu}_{\text{MLE}})^T$.

However, accurate estimation of the parameters is difficult with finite amount of samples especially in high dimensions ($D = 768$ for transformer-based models) due to curse of dimensionality, thereby (i) leading to sparse data points and spurious features (ii) and occasional outliers that influence the parameter estimates. In Figure 3, we empirically show that the covariance matrices (blue) of BERT and RoBERTa across all models across all datasets are ill-conditioned, demonstrated by the high maximum eigenvalues and extremely small minimum eigenvalues ($\approx 10^{-12}$). Due to this, the precision matrix is abnormally inflated in certain dimensions and prone to numerical errors during inversion. More analysis regarding the upperbound of this error is provided in Appendix A.2.

In addition, although we have assumed a Gaussian distribution for convenience, the unknown true distribution may be a more general elliptical distribution with thicker tails. This is observed empirically in Figure 4 by visualizing the features into two dimensions by dimensionality reduction. Outliers that are far from the modes of both classes (indicated by color) are present: those that are completely misplaced occasionally exist, while subtle outliers that deviate from the Gaussian distribution assumption are common, which influences the MLE estimation. Thus, to accurately estimate the Gaussian parameters, these outliers should be taken into account. In the following subsection, we tackle these issues through well-known classical techniques.

3.3 RDE using kPCA and Minimum Covariance Determinant

To address the first issue, we first use kernel PCA (Schölkopf et al., 1998, kPCA) to select top P orthogonal basis that best explains the variance of the data, thereby reducing redundant features. Given N centered samples $Z_{\text{train}} \in \mathbb{R}^{N \times D} = [z_1, \ldots, z_N]$, a mapping function $\phi : \mathbb{R}^D \to \mathbb{R}^{D'}$, and its mapping applied to each sample $\Phi(Z_{\text{train}}) \in \mathbb{R}^{N \times D'}$,

kPCA projects the data points to the eigenvectors with the P largest eigenvalues of the covariance $\Phi(Z_{\text{train}})^T\Phi(Z_{\text{train}})$[2]. Intuitively, this retains the most meaningful feature dimensions, which explains the data the most, while reducing spurious features and improve stability of inversion by increasing the condition number as shown in Figure 3. By leveraging non-linear ϕ, we are able to find meaningful non-linear signals in the features as opposed to standard PCA. We use the radial basis function as our kernel. Comparison of performance with standard PCA is provided in Appendix Table A.3. For a complete derivation, please refer to Schölkopf et al. (1997).

However, this does not remove sample-level outliers as shown in Figure 4. Since we have assumed a Gaussian distribution, "peeling off" outliers may be favorable for parameter estimation. A principled way of removing outliers for parameter estimation has been an important research area in multivariate statistics and various methods have been developed for robust covariance estimation (Friedman et al., 2008; Ledoit and Wolf, 2004). Among them, Minimum Covariance Determinant (Rousseeuw, 1984, MCD) finds a subset of $h \leq N$ samples that minimizes the determinant of Σ.[3] As the determinant is proportional to the differential entropy of a Gaussian up to a logarithm (shown in Appendix A.3), this results in a robust covariance estimation consisting of centered data points rather than outliers. For a review, see Hubert et al. (2018). Qualitatively, we observe in Figure 4 that for MLE estimates both classes have their means yanked towards the outliers and the contours are disoriented (Blue). MCD estimates (Red) focus on the high density clusters, which leads to higher performance as will be shown in the experiments.

In summary, we retain informative features by applying kPCA and obtain robust covariance estiamte by using MCD on the train set. Using the estimated robust parameters, we can evaluate the likelihood of a test sample. We treat those with low likelihood as novel (adversarial) samples. Our algorithm is shown in Algorithm 1 in the Appendix. We empirically validate the effectiveness of two techniques and discuss the effect of hyper-parameter P and h in the following sections.

[2]For simplicity, we assume $\Phi(Z_{\text{train}})$ is centered. When this assumption does not hold, slightly modified approach is taken. See Appendix B of Schölkopf et al. (1998) for details.

[3]Although the possible number of subsets is infeasibly large, Rousseeuw and Driessen (1999) propose an iterative method that converges relatively fast for ≈ 4000 samples with 100 dimensions.

4 Experiments

4.1 Experimental Settings

We experiment on the three datasets (IMDB, AGNews, SST-2) and four attack methods described in Section 3.1. Our experiment is mainly based on BERT and RoBERTa as they are widely used competent models for various tasks. Since SST-2 only has 1.8K test samples, TF-adjusted attack was unable to create an adequate number of successful adversarial samples (e.g. 80 samples out of 1.7K). Omitting experiments for these, there are 22 combinations of dataset-attack-model in total.

4.2 Compared Methods

We compare our robust density estimation method (**RDE**) with a recently proposed detection method in NLP called **FGWS** (Mozes et al., 2021) which is a word frequency-based method that assumes that rare words appear more often in adversarial samples. We also verify whether Perplexity (**PPL**) computed by a language model (GPT-2, Radford et al. 2019) is able to distinguish normal and adversarial samples as PPL is often used to compare the fluency of the two samples. **FGWS** implicitly models the input density via word frequencies, while GPT-2 explicitly computes the conditional probability via an auto-regressive tasks. In addition, we adopt Lee et al. (2018) denoted as MLE, which is a density estimation method from the image domain. For further details, see Section 5. We compare MLE with two variants of our method:

- **MLE**: Maximum likelihood estimate of parameters using the raw 768-dimensional features is used for density estimation.
- **RDE(-MCD)** : This is a variant of RDE, in which only kPCA is applied to the features without MCD. The results of applying standard PCA instead of kPCA are reported in Table A.3 of Appendix.
- **RDE** : After applying kPCA, MCD estimate is used. This is the final proposed robust density estimation incorporating both kPCA and MCD.

4.3 Evaluation Metric and Protocol

Following Xu et al. (2018), we report three widely used metrics in adversarial example detection : (1) *True positive rate (*TPR*)* is the fraction of true adversarial samples out of predicted adversarial samples. (2) *F1-score* (f1) measures the harmonic mean of

Models	Methods	Attacks											
		TF			PWWS			BAE			TF-adj		
		TPR	F1	AUC	TPR	F1	AUC	TPR	F1	AUC	TPR	F1	AUC
		IMDB											
BERT	PPL	48.7±0.2	61.4±0.2	76.9±0.2	37.8±0.5	51.2±0.5	71.7±0.2	27.0±0.5	39.4±0.5	67.3±0.1	24.5±0.8	36.5±1.0	67.9±0.3
	FGWS	84.6±0.3	87.1±0.2	87.1±0.3	**88.2**±0.1	**89.0**±0.0	90.8±0.0	62.1±0.3	72.3±0.2	70.9±0.3	72.6±0.9	80.6±0.9	78.4±0.6
	MLE	86.3±1.1	87.9±0.7	94.5±0.2	75.7±1.4	81.5±0.9	92.4±0.2	81.8±1.3	85.3±0.8	93.7±0.2	88.3±1.0	89.1±0.6	95.3±0.2
	RDE(-MCD)	96.3±0.3	93.4±0.2	96.8±0.1	86.9±0.9	88.3±0.5	94.6±0.2	92.5±0.5	91.4±0.3	95.8±0.2	98.2±0.2	94.6±0.2	97.6±0.2
	RDE	**96.6**±0.2	**93.5**±0.1	**97.7**±0.2	87.8±0.4	88.8±0.2	**95.2**±0.2	**93.8**±0.1	**92.1**±0.0	**96.9**±0.2	**98.8**±0.0	**95.0**±0.1	**98.7**±0.2
RoBERTa	PPL	47.8±0.1	60.6±0.1	78.4±0.1	43.5±0.7	56.7±0.6	76.1±0.2	25.9±0.4	38.2±0.5	67.0±0.2	26.6±0.9	39.0±1.1	69.1±0.4
	FGWS	85.1±0.1	87.4±0.1	88.0±0.1	92.1±0.2	91.4±0.2	93.6±0.2	61.5±0.2	71.8±0.1	70.3±0.1	69.2±0.4	78.0±0.1	75.4±0.2
	MLE	80.5±1.0	84.5±0.6	94.0±0.2	76.8±1.3	82.2±0.8	93.3±0.2	75.5±1.5	81.4±0.9	93.1±0.3	86.4±2.3	88.0±1.3	95.3±0.7
	RDE(-MCD)	98.5±0.1	94.5±0.1	97.9±0.1	95.0±0.3	92.7±0.2	96.7±0.1	**95.4**±0.4	**93.0**±0.2	97.0±0.2	98.6±0.4	94.8±0.2	98.1±0.4
	RDE	**98.9**±0.1	**94.7**±0.0	**98.6**±0.1	**95.2**±0.1	**92.8**±0.1	**97.2**±0.1	95.3±0.2	92.9±0.1	**97.6**±0.1	**98.8**±0.4	**95.9**±0.6	**99.0**±0.2
		AG-News											
BERT	PPL	75.7±0.4	81.6±0.2	91.0±0.2	70.8±0.7	78.3±0.5	89.5±0.2	31.2±1.3	44.2±1.4	73.0±0.8	32.8±1.8	45.9±1.9	73.3±0.8
	FGWS	82.4±0.6	85.7±0.3	84.2±0.7	**91.0**±0.1	**90.6**±0.1	90.8±0.3	64.3±0.9	73.8±0.6	71.3±0.4	63.8±1.0	74.3±1.0	71.9±0.7
	MLE	77.8±0.5	82.9±0.3	93.5±0.1	70.4±0.9	78.0±0.6	92.0±0.1	72.7±1.8	79.6±1.2	92.8±0.4	71.0±1.6	78.9±0.9	92.0±0.2
	RDE(-MCD)	**96.2**±0.1	**93.3**±0.0	**97.1**±0.1	89.8±0.8	90.0±0.4	**95.6**±0.1	93.2±0.9	92.1±0.6	96.2±0.3	96.6±1.0	93.6±0.5	96.0±0.1
	RDE	95.8±0.2	93.2±0.1	96.9±0.1	88.7±1.0	89.3±0.6	95.4±0.1	**96.6**±0.1	**93.7**±0.1	**96.9**±0.1	**98.2**±0.6	**95.4**±0.3	**97.5**±0.3
RoBERTa	PPL	77.1±0.5	82.4±0.3	91.8±0.1	72.2±0.8	79.3±0.5	89.6±0.2	37.1±1.4	50.4±1.5	74.7±0.3	31.8±1.3	45.3±1.3	74.3±1.3
	FGWS	78.8±0.5	83.5±0.3	82.2±0.2	**86.6**±0.4	**88.1**±0.2	87.9±0.3	53.3±3.4	65.1±2.7	63.5±2.0	58.9±3.4	69.7±2.6	70.1±0.9
	MLE	82.5±0.3	85.7±0.2	94.1±0.1	78.6±0.5	83.4±0.3	92.9±0.2	68.1±3.1	76.3±2.2	91.5±0.7	65.0±2.3	74.4±1.7	91.2±0.2
	RDE(-MCD)	90.5±0.5	90.3±0.3	**96.1**±0.1	84.1±1.2	86.6±0.7	**94.8**±0.2	77.8±4.1	82.6±2.6	93.9±0.5	82.6±2.7	85.9±1.5	94.5±0.4
	RDE	**92.9**±0.3	**91.6**±0.2	95.7±0.1	84.5±0.8	86.9±0.5	93.9±0.2	**89.3**±2.3	**89.6**±1.3	**95.3**±0.5	**94.4**±0.7	**92.6**±0.4	**96.0**±0.3
		SST-2											
BERT	PPL	31.7±0.6	44.8±0.6	73.1±0.3	29.2±1.3	41.9±1.4	73.4±0.4	22.2±1.6	33.5±2.0	67.0±0.5			
	FGWS	60.8±0.4	72.3±0.3	73.6±0.3	**79.9**±0.6	**84.9**±0.4	**86.7**±0.4	34.7±0.3	48.0±0.3	60.3±0.3			
	MLE	33.3±1.3	46.5±1.4	79.8±0.5	23.2±0.4	34.8±0.6	78.4±0.3	32.6±1.3	45.8±1.5	76.8±0.6			
	RDE(-MCD)	61.3±0.8	71.6±0.6	86.3±0.4	46.6±0.7	59.5±0.6	84.6±0.2	45.4±1.3	58.5±1.1	80.6±0.6			
	RDE	**66.1**±0.8	**75.1**±0.5	**87.7**±0.3	54.3±1.1	66.1±0.9	86.6±0.2	**48.0**±1.4	**60.7**±1.2	**81.0**±0.5			
RoBERTa	PPL	34.7±0.7	48.0±0.7	75.0±0.5	32.5±1.6	45.5±1.7	73.9±0.5	20.0±1.3	30.8±1.6	65.3±0.4			
	FGWS	61.6±0.2	73.0±0.1	73.7±0.1	**80.8**±0.2	**85.6**±0.1	86.4±0.2	36.1±1.0	49.4±1.1	60.0±0.6			
	MLE	44.2±0.6	57.3±0.5	84.4±0.3	33.1±0.8	46.3±0.8	81.9±0.4	37.1±0.5	50.5±0.5	77.9±0.4			
	RDE(-MCD)	63.2±0.2	73.0±0.1	87.8±0.1	53.1±0.7	65.1±0.6	85.4±0.2	45.7±0.7	58.7±0.7	79.3±0.3			
	RDE	**74.1**±0.3	**80.6**±0.2	**90.4**±0.1	67.7±1.1	76.2±0.8	**89.1**±0.0	**52.0**±0.3	**64.3**±0.3	**80.6**±0.1			

Table 3: Adversarial detection results for BERT and RoBERTa on three datasets on Scenario 1. For all metrics, highers mean better.

precision and recall. Since all compared methods are threshold-based methods, we report TPR at a fixed false positive rate (FPR). (3) *Area under ROC (*AUC*)* measures the area under TPR vs. FPR curve. For all three metrics, higher denotes better performance.

Note that whereas AUC considers performance on various FPR's, TPR and F1 is dependent on one particular FPR. In all our experiments, we fixed FPR$= 0.1$, which means 10% of normal samples are predicted to be adversarial samples. This threshold should be chosen depending on the context (i.e. the degree of safety-criticalness). We believe this standard should be elevated as more works are proposed in the future. For IMDB and AG-News, 30% of the test set is held out as validation set, and we subsample out of the test set as described in Section 2.2. For quantitative analysis, we report the mean and its standard error of three repetitions of random seeds for test/validation split and subsampled samples.

4.4 Implementation Details

We choose the feature z to be the output of the last attention layer (before Dropout and fully connected layer) for BERT and RoBERTa. RDE has two main hyper-parameters, namely the number of retained dimensions P of kPCA and the support fraction h of MCD. We fix $P = 100$ for all experiments as we observe the performance is not sensitive to P. For h, we use the default value proposed in the algorithm, which is $\frac{N+P+1}{2N}$. We study the effect of h in Section 4.6. All models are pre-trained models provided by TextAttack and both kPCA and MCD are implemented using scikit-learn (Pedregosa et al., 2011). We use the radial basis function as our kernel. The time required to estimate the parameters of our method is approximately around 25 seconds. For more details, see Appendix A.4.

For FGWS, we use the official implementation[4] and use the held-out validation set of each attack to tune the threshold for word frequency δ as done in the original work. For PPL, we use the HuggingFace (Wolf et al., 2020) implementation of GPT-2 (Radford et al., 2019).

4.5 Results

Main Results

Here we present our main experiments done on Scenario 1. Table 3 demonstrates the results on all datasets and attacks. The highest means out of the five methods are written in bold. Out of the **22** com-

[4]https://github.com/maximilianmozes/fgws

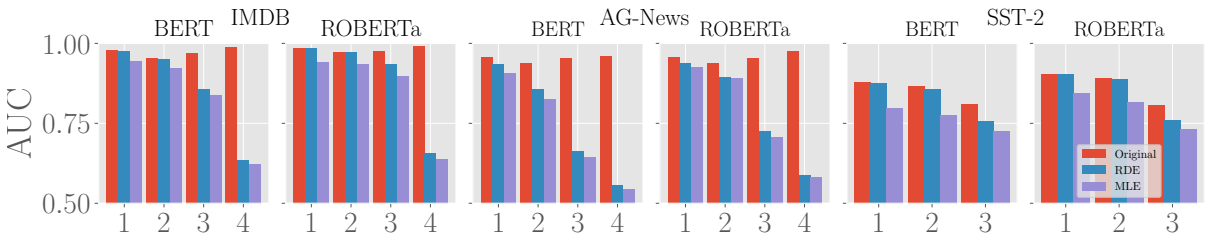

Figure 5: Detection performance (AUC) for scenario including failed adversarial samples. Horizontal axis dentoes the type of attacks methods (TF, PWWS, BAE, TF-adj). The original performance of RDE from Table 3 as an upper bound is provided (red). Blue and purple denotes RDE and MLE including failed adversarial samples, respectively.

binations of dataset-attack-model, RDE achieves the best performance on **21** of them on AUC and **17** of them for all three metrics, which shows the competitiveness of our simple method. The motivation of our method is verified by the large margin of improvement from MLE in almost all cases. Using MCD estimation also further improves performance except in the few cases of AG-News. Large language models (PPL) are able to distinguish between adversarial samples and normal samples in expectation as shown by the higher-than-random metric, but the performance is inadequate to be used as a detector. FGWS generally has a higher performance compared to PPL, but is inferior to MLE in most cases. Note the degradation of performance in FGWS for BAE and TF-adj, which are more subtle attacks with stronger constraint, as FGWS relies on use of rare words. This trend is not observed in feature density-based methods (MLE and RDE).

FGWS outperforms ours on TPR and F1 in five combinations out of 22, but our method has higher AUC on four of them. Interestingly, all the five results are from PWWS attacks, which indicates that our method is relatively susceptible to PWWS. Nonetheless, AUC still remains fairly high: On IMDB and AG-News, the AUC's are all over 0.9. On the other hand, all methods have degraded performance on SST-2, which may be due to shorter sentence lengths. Some examples of ROC curves are presented in Appendix A.6. Improving detection rate in SST-2 is left as a future work. Comparing with BERT, RoBERTa generally has higher performance on IMDB and SST-2, yet BERT outperforms RoBERTa on AG-News on some attacks.

Motivating More Realistic Scenarios In this section, we provide some preliminary results for other more realistic scenarios: (i) Including failed attacks (ii) Imbalanced ratio of clean and adversarial samples. In previous experiments, all failed adversarial attacks were discarded. However, in reality an adversary will probably have no access to the victim model so some attacks will indeed fail to fool the model. While failed adversarial attacks do not pose threat to the task accuracy of the model, it nevertheless may be harmful if the victim wishes to gain information about a certain population by aggregating data such as sentiment in consumer review about a movie. In addition, as active research in attacks have been made in the past few years, more subtle attacks that are imperceptible to humans naturally have lower attack success ratio (e.g. BAE).

Figure 5 provides the detection results of RDE and MLE when distinguishing between normal samples and (failed and successful) adversarial attempts by comparing the AUC's. As an upper bound, we provide the performance of RDE on the *original scenario* without failed adversarial examples in red. As the first two attacks (TF and PWWS) achieve nearly 100% success rate, only few failed adversarial samples are added. Accordingly, the performances for the two attacks show little difference. However, in more subtle attacks (BAE and TF-adj) the performance drastically drops due to the increased failed adversarial samples, yet RDE outperforms MLE by a considerable margin in most cases. We end on this topic by noting that more comprehensive analysis is called for, because in some cases failed adversarial attempts are (nearly) identical to clean samples. So an attack method with low detection rate does not necessarily imply a crafty attack method in this scenario.

In Appendix Table A.4, we provide the results for Scenario 2 described in Section 2.2. The general trend among detection methods and attack methods is similar to Table 3. As noted earlier, for Scenario 2 the ratio of adversarial to clean samples will be low if the attack success rate is low. For instance, in IMDB-(TF-adj)-BERT, the ratio of adversarial to clean samples is around 1:9. Whereas both AUC and TPR are not strongly affected due to the characteristic of the metrics, F1 drastically drops. For

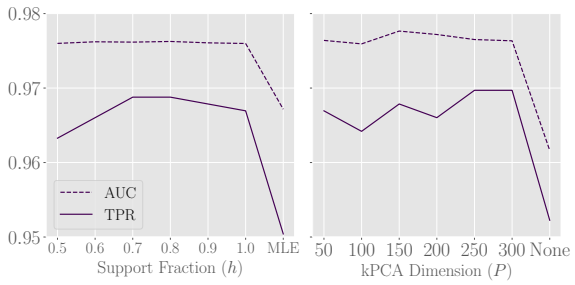

Figure 6: Hyperparameter analysis on IMDB-TF-BERT. Wide range of values all outperform the ablated forms and are relatively stable.

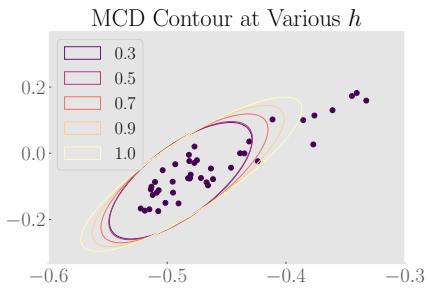

Figure 7: Qualitative example of varying support fraction h on SST2-BERT. Each ellipse represent probability contours of three standard deviations. Higher h leads to wider contour.

instance, for IMDB-(TF-adj)-BERT, RDE achieves 73.7% (as opposed to 95.0% of Scenario 1). On the same set, FGWS achieves 60.1% and MLE achieves 67.6%.

4.6 Discussion

Hyper-parameter Analysis Although the two main hyper-parameters, support fraction (h) of MCD and the dimension (P), were fixed in our experiments, they can be fine-tuned in a separate validation set for optimal performance. We show in Figure 6 the performance of our method on various ranges of h and P on the validation set of IMDB-TF-BERT combination. We set $P = 100$ and h to the default value of the algorithm when tuning for the other parameter. We confirm that RDE has a relatively robust performance across wide ranges of values and improves upon the naive version as shown by the "MLE" and "None", in which MLE estimation and no kPCA were used, respectively.

Qualitative Analysis on Support Fraction The support fraction controls the ratio of original samples to be retained by the MCD estimator, thereby controlling the volume of the contour as shown in Figure 7. Large values of h retain more of the deviating samples and lead to a wider probability contour. We empirically demonstrated in our experiment that using all samples for parameter estimation may be detrimental for adversarial sample detection.

5 Related Works

Detection of adversarial examples is a well-explored field in the image domain. Earlier works have tackled in various ways such as input transformation (Xu et al., 2018), statistical analysis (Grosse et al., 2017), or training a separate binary classifier (Metzen et al., 2017). However, Carlini and Wagner (2017) has shown that an adversary with partial knowledge of the detector can easily nullify it. Meanwhile, early works in novelty detection have shown that a generative model can detect anomaly samples (Bishop, 1994). Following this line of research, Lee et al. (2018) have proposed a method to detect out-of-distribution samples by using features of a neural network for maximum likelihood estimation.

In the NLP domain, few efforts have been made in detecting word-level adversarial examples. Zhou et al. (2019, DISP) utilize a detector trained on adversarial samples for a joint detect-defense system. FGWS (Mozes et al., 2021) outperforms DISP in detection by building on the observation that attacked samples are composed of rare words. FGWS experiements on 2 proposed attack methods. Le et al. (2021) tackle a particular attack method called UniTrigger (Wallace et al., 2019), which pre-pends or appends an identical phrase in all sentences. While the performance is impressive, applying this method to other attacks requires significant adjustment due to the distinct characteristics of UniTrigger. Meanwhile, Pruthi et al. (2019) tackle character-level adversarial examples and compare with spell correctors. Our work is the first to extensively demonstrate experimental results for 4 attack methods including recent attack methods on 3 datasets and propose a competitive baseline.

6 Conclusion

We propose a general framework and benchmark for adversarial example detection in NLP. Along with it, we propose a competitive baseline that does not require training nor validation for each attack method. In the future, an adversary with partial or full knowledge should be considered to motivate further research for stronger detection methods.

References

Theodore Wilbur Anderson. 1962. An introduction to multivariate statistical analysis. Technical report, Wiley New York.

Rongzhou Bao, Jiayi Wang, and Hai Zhao. 2021. Defending pre-trained language models from adversarial word substitutions without performance sacrifice. *arXiv preprint arXiv:2105.14553*.

C.M. Bishop. 1994. Novelty detection and neural network validation. *IEEE Proceedings - Vision, Image and Signal Processing*, 141:217–222(5).

Nicholas Carlini and David Wagner. 2017. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*, pages 3–14.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Céspedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. 2008. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441.

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.

Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick McDaniel. 2017. On the (statistical) detection of adversarial examples. *arXiv preprint arXiv:1702.06280*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Mia Hubert, Michiel Debruyne, and Peter J Rousseeuw. 2018. Minimum covariance determinant and extensions. *Wiley Interdisciplinary Reviews: Computational Statistics*, 10(3):e1421.

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 *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8018–8025.

Erik Jones, Robin Jia, Aditi Raghunathan, and Percy Liang. 2020. Robust encodings: A framework for combating adversarial typos. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2752–2765.

Yannik Keller, Jan Mackensen, and Steffen Eger. 2021. BERT-defense: A probabilistic model based on BERT to combat cognitively inspired orthographic adversarial attacks. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1616–1629, Online. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882.

Thai Le, Noseong Park, and Dongwon Lee. 2021. A sweet rabbit hole by darcy: Using honeypots to detect universal trigger's adversarial attacks. In *59th Annual Meeting of the Association for Comp. Linguistics (ACL)*.

Olivier Ledoit and Michael Wolf. 2004. A well-conditioned estimator for large-dimensional covariance matrices. *Journal of multivariate analysis*, 88(2):365–411.

Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. 2018. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *Advances in neural information processing systems*, 31.

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

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 preprint arXiv:1907.11692*.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. 2017. On detecting adversarial perturbations. *International Conference on Learning Representations*.

John Morris, Eli Lifland, Jack Lanchantin, Yangfeng Ji, and Yanjun Qi. 2020a. Reevaluating adversarial examples in natural language. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 3829–3839.

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

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.

Nikola Mrkšić, Diarmuid O Séaghdha, Blaise Thomson, Milica Gašić, Lina Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting word vectors to linguistic constraints. *arXiv preprint arXiv:1603.00892*.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Princeton. 2010. Princeton university "about wordnet." wordnet.

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, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

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

Peter J Rousseeuw. 1984. Least median of squares regression. *Journal of the American statistical association*, 79(388):871–880.

Peter J Rousseeuw and Katrien Van Driessen. 1999. A fast algorithm for the minimum covariance determinant estimator. *Technometrics*, 41(3):212–223.

Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. 1997. Kernel principal component analysis. In *International conference on artificial neural networks*, pages 583–588. Springer.

Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. 1998. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319.

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

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing nlp. 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 2153–2162.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Weilin Xu, David Evans, and Yanjun Qi. 2018. Feature squeezing: Detecting adversarial examples in deep neural networks. *Network and Distributed Systems Security Symposium*.

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

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

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)*, pages 4904–4913.

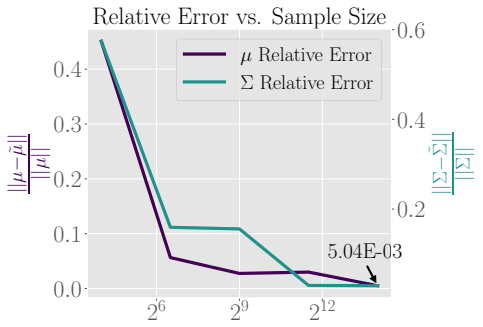

Figure A.1: Left figure shows the relative error of estimating the parameters of 768-dimensional multivariate Gaussian on a toy example. Even with 2^{14} samples, the relative error of μ is on the scale of 10^{-3}

A Appendix

A.1 Success Rate of Attack Methods and Other Statistics

Here we briefly describe each attack method and provide some statistics about the attack results. For Table A.1, word transformation method indicates how candidate replacement words are created. Word importance ranking denotes how the ordering of which word to attack is chosen. For constraints, only those related to embedding was listed and the numbers in parenthesis denote the threshold. Higher threshold signifies stronger constraint. For more details, we refer the readers to Morris et al. (2020b). Table A.2 summarizes the attack results on three dataset for BERT. Results for other models can be found in the released dataset.

A.2 Potential Errors of Parameter Estimation

Accurate estimation of the parameters is difficult with finite amount of samples especially in high dimensions. Here we demonstrate this through a toy example and derive its relationship with the Mahalanobis distance function, which is proportional to the likelihood. Figure A.1 shows that the MLE error remains high even when 2^{14} samples are used to find the parameters of a noise-free normal distribution for both μ and Σ. This, in turn, leads to an inevitably error-prone $\tilde{\mu} = \mu - \epsilon_\mu$ and $\tilde{\Delta} = z - \tilde{\mu}$. Moreover, the error is amplified when computing the Mahalanobis distance due to the ill-conditioned $\tilde{\Sigma}$ with very small eigenvalues, which is observed empirically in all datasets and models (Figure 3) possibly due to the redundant features. The (relative) condition number of the Mahalanobis distance function $g(\Delta)$ - relative change in the output given a relative change in the inputs - is bounded by the inverse of the smallest eigenvalue of $\tilde{\Sigma}^{-1}$.

$$\begin{aligned}\kappa_g(\Delta) &= \frac{||\frac{\partial g}{\partial \Delta}||}{||g(\Delta)||/||\Delta||} \\ &= \frac{||\Delta||}{||g(\Delta)||}||2\Sigma^{-1}\Delta|| \\ &\leq \frac{||\Delta||}{||g(\Delta)||}2||\Sigma^{-1}||||\Delta||\end{aligned} \tag{2}$$

where the first equality follows from the definition of condition number and differentiability of g and C_Δ. The last equality follows from the Caucy-Schwarz Inequality. The matrix norm induced by the L2 norm is given by the largest singular value (largest eigenvalue for a positive definite square matrix). Given the eigenspectrum of Σ as $\lambda_{\text{max}} \geq \cdots \geq \lambda_{\text{min}}$, the eigenspectrum of Σ^{-1} is given by the reciprocal of that of Σ. Thus, $||\Sigma^{-1}||$ is equal to inverse of the minimum eigenvalue of Σ and the last equality can be further decomposed into

$$\begin{aligned}\kappa_g(\Delta) &\leq \frac{||\Delta||}{||g(\Delta)||}2||\Sigma^{-1}||||\Delta|| \\ &\leq C_\Delta \frac{1}{\lambda_{\text{min}}}\end{aligned} \tag{3}$$

where C_Δ is a constant for a given Δ. This means that when the smallest eigenvalue is in the scale of 10^{-12}, even a estimation error of scale 10^{-3} on μ may be amplified by at most by a scale of 10^9. This leads to a serious problem in density estimation of z.

A.3 More details on MCD

We explain some of the properties of the determinant of the covariance matrix. First, the determinant is directly related to the differential entropy of the Gaussian distribution. For a D-dimensional multivariate Gaussian variable X and its probability density function f, the differential entropy is given by

$$\begin{aligned}H(X) &= -\int_{\mathcal{X}} f(x)\log f(x)dx \\ &= \frac{1}{2}\log((2\pi e)^D \det(\Sigma) \\ &\propto \det(\Sigma)\end{aligned} \tag{4}$$

In addition, the determinant is also proportional to the volume of the ellipsoid for some k, $\{z \in \mathbb{R}^D : (z-\mu)^T\Sigma^{-1}(z-\mu) = k^2\}$. We refer the

Attacks	Citations	Word Transformation Method	Word Importance Ranking Method	Constraints
TF	204	Counter-fitted GLOVE (Mrkšić et al., 2016)	Delete Word	USE(0.84) WordEmbedding Distance(0.5)
PWWS	187	WordNet (Princeton, 2010)	Weighted Saliency	-
BAE	71	Bert Masked LM	Delete Word	USE(0.94)
TF-adj	18	Counter-fitted GLOVE	Delete Word	USE(0.98) WordEmbedding Distance(0.9)

Table A.1: Summary of attack methods and their defining characteristics.

Attacks	Post-Attack Accuracy	Attack Success Rate	Average Num. Queries
IMDB (91.9%)			
TF	0.6%	99%	558
PWWS	3%	97%	1681
BAE	34%	64%	455
TF-adj	84.2%	11%	298
AG-News (94.2%)			
TF	18%	81%	334
PWWS	41%	57%	362
BAE	82%	14%	122
TF-adj	91%	5%	56
SST-2 (92.43%)			
TF	4%	96%	91
PWWS	12%	87%	143
BAE	37%	61%	60
TF-adj	89%	5%	25

Table A.2: Summary of attack results for BERT on three datasets. Original accuracy of each dataset is written in parenthesis next to the dataset.

readers to Section 7.5 of Anderson (1962) for the proof. This explain why the MCD estimate forms a much narrower probability contour than MLE as shown in Fig. 4.

A.4 Implementation Details

To meet the memory constraint of computing the kernel matrix, we sample a subset of $\mathcal{X}_{\text{train}}$ (8,000 samples) for all experiments. All models are pre-trained models provided by TextAttack and both kPCA and MCD are implemented using scikit-learn (Pedregosa et al., 2011). We use the radial basis function as our kernel.

For subsampling generated attacks described in Section 2.2, we set the maximum number of adversarial samples for each dataset. For IMDB and AG-News, the maximum is set to 2000 and for SST-2 this is set to 1000. Then, the number of target samples (i.e. $||\mathcal{S}||$ or $||\mathcal{S}_1||$) is initialized to the maximum number divided by adversarial success ratio and task accuracy. Target sample is decremented until ratio between clean and adversarial samples can roughly be 5:5. Algorithm of RDE and MLE is provided in Algorithm 1.

Algorithm 1: RDE and MLE

Input: $\mathcal{X}_{\text{train}}, \mathcal{Y}_{\text{train}}, \mathcal{D} = \{\mathcal{D}_{\text{adv}}, \mathcal{D}_{\text{clean}}\}$
Input: Feature Extractor $\mathcal{H}$
Output: Likelihood L

$\mathcal{Z}_{\text{train}} = \mathcal{H}(\mathcal{X}_{\text{train}})$
if *MLE* **then**
  for `c in Class` **do**
    $\tilde{\mu}_c = \frac{1}{N_c}\sum_{i \in Y_c}^{N_c} z_i$
    $\tilde{\Sigma}_c = \frac{1}{N_c}\sum_{i \in Y_c}^{N_c} (z_i - \tilde{\mu})(z_i - \tilde{\mu})^T$
else if *RDE* **then**
  $\overline{Z}$ = kPCA($\mathcal{Z}$)
  for *c in Class* **do**
    $\tilde{\mu}_c, \tilde{\Sigma}_c = \text{MCD}(\overline{Z_c})$
L = []
for x `in` $\mathcal{D}$ **do**
  $z = \mathcal{H}(x)$
  $\hat{y} = argmax_k\ \mathcal{F}(x)_k$
  if *RDE* **then**
    z = kPCA(z)
  L.`append`($\mathcal{N}(z|\tilde{\mu}_{\hat{y}}, \tilde{\Sigma}_{\hat{y}})$)

A.5 Benchmark Usage Example

```
from AttackLoader import Attackloader

#Set seed, scenario, model type,
#attack type, dataset, etc.
loader = AttackLoader(...)

#Split test and validation set
loader.split_csv_to_testval()

# Subsample from testset according to
    chosen scenario
sampled, _ =
    loader.get_attack_from_csv(..)

"""
Apply detection method
"""
```

A.6 ROC Curve Examples

Below (Fig. A.2) we provide Receiver Operating Characteristic (ROC) curves of ROBERTA on two attacks. For all plots, samples from the first seed are used.

A.7 Comparison with PCA

In all our experiments, we used a radial basis function for kPCA. This allows finding non-linear patterns in the feature space. When the linear kernel is used, kPCA is equivalent to ordinary PCA. We demonstrate that exploiting non-linearity preserve much more meaningful information by comparing the detection performance in the IMDB dataset (Table A.3).

A.8 Experiment results on Scenario 2

Here we provide experimental results on Scenario 2. Although pairs of samples are included in the dataset, the general trend is similar to that of Scenario 1. For attack methods with low success rate, the adversarial to clean sample ratio is low, which affects the F1-score.

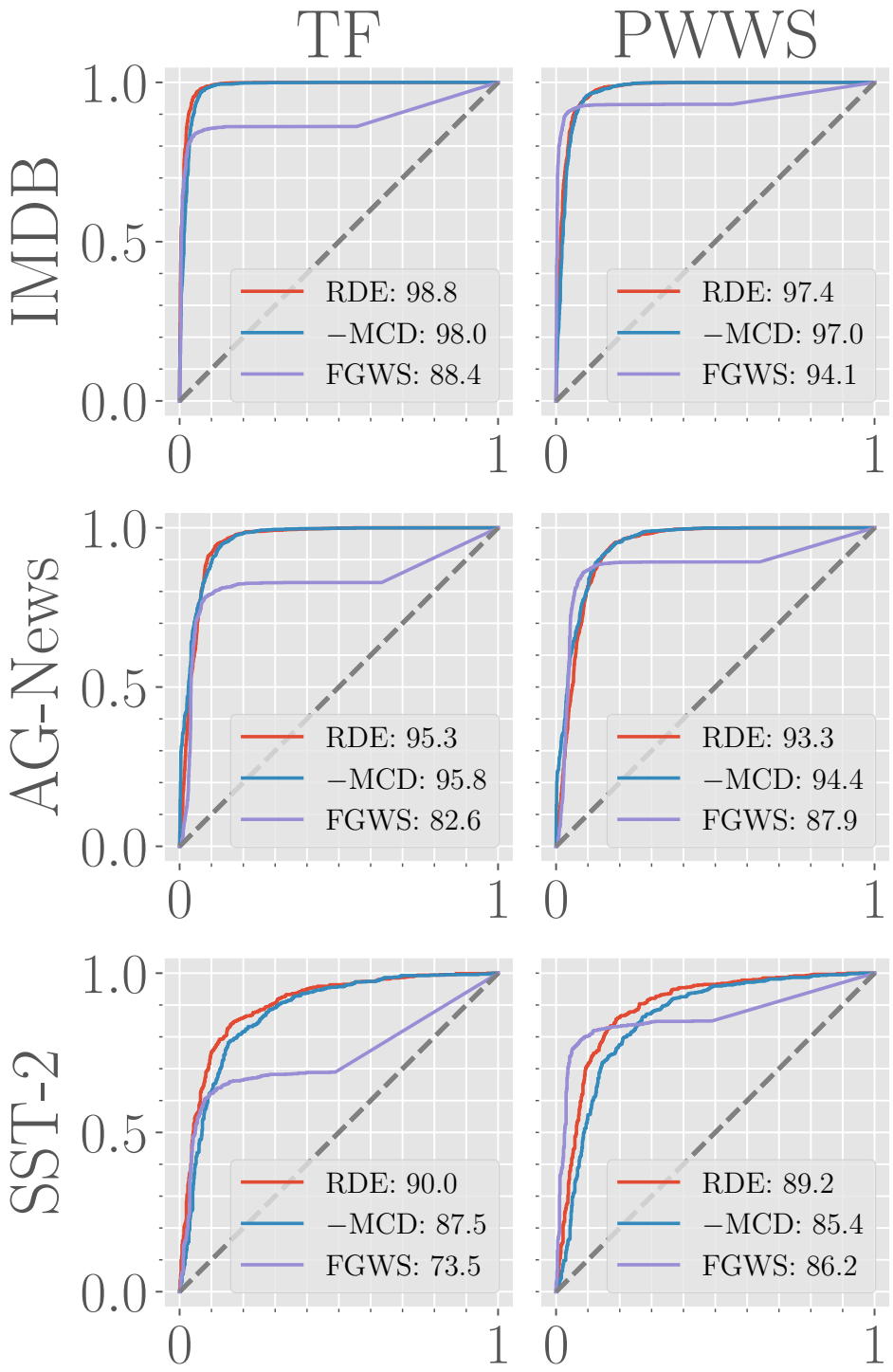

Figure A.2: ROC curves for RoBERTa on two attacks TF and PWWS. Row indicates the dataset, while the column indicates the attack methods. For all plots, the x-axis and y-axis represents FPR and TPR, respectively. The legend indicates the AUC of each methods. RDE(-MCD) is written as -MCD due to space constraint.

Models	Methods	Attacks											
		TF			PWWS			BAE			TF-adj		
		TPR	F1	AUC	TPR	F1	AUC	TPR	F1	AUC	TPR	F1	AUC
BERT	PCA	89.5±1.1	89.7±0.6	95.2±0.1	77.9±1.3	82.9±0.8	93.1±0.2	83.5±1.3	86.3±0.7	94.2±0.2	92.8±1.2	91.7±0.7	96.2±0.2
	kPCA	96.3±0.3	93.4±0.2	96.8±0.1	86.9±0.9	88.3±0.5	94.6±0.2	92.5±0.5	91.4±0.3	95.8±0.2	98.2±0.2	94.6±0.2	97.6±0.2
RoBERTa	PCA	94.7±0.6	92.5±0.3	96.3±0.1	89.7±0.9	89.9±0.5	95.4±0.1	88.2±1.3	89.0±0.7	95.0±0.3	92.6±1.8	91.6±0.9	96.7±0.5
	kPCA	98.5±0.1	94.5±0.1	97.9±0.1	95.0±0.3	92.7±0.2	96.7±0.1	95.4±0.4	93.0±0.2	97.0±0.2	98.6±0.4	94.8±0.2	98.1±0.4

Table A.3: Results of using linear kernel for KPCA, which is equivalent to the ordinary PCA on IMDB. For fair comparison, we compare with RDE(-MCD) where MCD estimation is not used. All results lead to higher performance when kPCA is used.

Models	Methods	Attacks											
		TF			PWWS			BAE			TF-adj		
		TPR	F1	AUC	TPR	F1	AUC	TPR	F1	AUC	TPR	F1	AUC
		IMDB											
BERT	PPL	49.3±0.2	61.5±0.2	77.4±0.2	38.9±0.3	51.9±0.3	71.9±0.2	28.1±0.3	38.8±0.3	67.6±0.1	24.3±0.1	25.0±0.2	66.7±0.1
	FGWS	82.6±0.1	85.4±0.1	85.1±0.1	86.6±0.1	87.7±0.1	89.3±0.1	60.6±0.2	68.5±0.2	69.3±0.2	71.3±0.2	60.1±0.2	76.6±0.1
	MLE	86.4±1.2	87.6±0.7	94.6±0.1	76.1±1.7	81.3±1.1	92.6±0.2	82.0±0.6	82.6±0.3	93.7±0.1	87.0±0.2	67.6±0.2	95.0±0.0
	RDE(-MCD)	96.0±0.4	92.8±0.2	96.6±0.1	86.2±0.8	87.5±0.5	94.4±0.1	92.1±0.3	88.3±0.2	95.6±0.1	98.5±0.0	73.4±0.0	97.5±0.0
	RDE	96.8±0.2	93.2±0.1	97.7±0.1	87.4±0.5	88.1±0.3	95.1±0.2	93.3±0.3	89.0±0.1	96.8±0.1	98.8±0.0	73.7±0.1	98.6±0.0
RoBERTa	PPL	49.5±0.4	61.8±0.3	78.9±0.3	45.1±0.2	57.9±0.2	76.5±0.2	26.9±0.1	37.9±0.1	67.6±0.1	26.6±0.4	21.3±0.3	68.1±0.3
	FGWS	83.5±0.2	86.2±0.1	86.6±0.2	91.6±0.1	90.7±0.0	93.1±0.1	60.7±0.1	69.1±0.1	69.3±0.2	66.0±0.4	46.9±0.2	72.9±0.4
	MLE	80.7±0.8	84.4±0.5	94.0±0.0	77.3±1.0	82.3±0.6	93.2±0.1	75.9±1.0	79.5±0.6	93.0±0.1	84.1±0.7	54.7±0.2	94.3±0.0
	RDE(-MCD)	98.1±0.1	94.1±0.0	97.9±0.0	94.7±0.3	92.3±0.2	96.7±0.0	95.0±0.1	90.5±0.0	96.8±0.0	98.9±0.1	61.7±0.1	97.8±0.0
	RDE	98.5±0.1	94.3±0.0	98.6±0.0	94.9±0.4	92.4±0.2	97.2±0.0	94.8±0.1	90.4±0.0	97.5±0.0	99.1±0.1	62.1±0.1	98.9±0.0
		AG-News											
BERT	PPL	76.3±0.3	80.7±0.2	91.3±0.1	72.4±0.4	75.8±0.3	90.2±0.1	30.6±0.5	29.9±0.4	72.8±0.2	35.0±0.5	19.8±0.2	74.3±0.4
	FGWS	82.4±0.6	84.4±0.4	84.2±0.7	90.9±0.2	86.8±0.1	90.0±0.1	62.9±0.3	53.0±0.3	70.5±0.1	66.0±0.6	36.3±0.6	72.4±0.4
	MLE	78.2±0.4	81.8±0.3	93.5±0.0	71.4±0.3	75.3±0.2	92.3±0.1	69.9±0.3	57.4±0.1	92.2±0.0	64.0±0.5	33.4±0.4	91.1±0.1
	RDE(-MCD)	96.3±0.3	92.1±0.1	97.2±0.0	90.5±0.3	86.6±0.1	95.7±0.1	91.4±0.3	68.8±0.1	95.5±0.0	92.2±0.5	44.4±0.1	95.6±0.0
	RDE	96.0±0.3	91.9±0.1	97.0±0.0	89.8±0.4	86.2±0.2	95.4±0.0	94.0±0.2	69.9±0.1	96.2±0.0	96.6±0.1	47.4±0.3	96.6±0.0
RoBERTa	PPL	77.3±0.3	81.5±0.2	91.8±0.1	73.0±0.3	77.2±0.2	90.0±0.1	36.2±0.5	36.3±0.5	74.9±0.3	36.2±0.3	21.5±0.2	75.8±0.2
	FGWS	79.6±0.4	83.0±0.2	82.6±0.3	86.6±0.3	85.5±0.2	88.0±0.2	52.7±0.4	48.9±0.3	64.6±0.5	60.7±0.7	34.4±0.3	70.2±0.3
	MLE	81.4±0.2	84.1±0.1	94.0±0.1	78.7±0.0	80.8±0.0	93.1±0.0	68.4±0.2	59.1±0.2	91.7±0.1	62.6±0.2	34.3±0.2	90.0±0.0
	RDE(-MCD)	89.9±0.3	89.0±0.2	96.1±0.0	85.8±0.5	85.0±0.3	95.1±0.1	81.4±0.2	66.6±0.2	94.4±0.1	80.4±0.1	42.0±0.2	93.7±0.0
	RDE	92.7±0.2	90.5±0.1	95.6±0.0	86.4±0.3	85.4±0.2	94.2±0.1	92.6±0.2	72.3±0.2	95.7±0.0	93.6±0.1	47.6±0.4	95.7±0.0
		SST-2											
BERT	PPL	33.4±0.5	46.2±0.6	73.1±0.1	30.4±0.5	42.6±0.5	73.1±0.0	22.2±0.1	31.7±0.1	65.7±0.0			
	FGWS	61.4±0.6	71.2±0.5	73.5±0.4	79.4±0.2	82.9±0.1	86.2±0.1	33.0±0.3	43.8±0.3	61.2±0.2			
	MLE	33.3±0.2	46.1±0.2	80.5±0.1	23.3±0.5	34.4±0.6	78.4±0.1	34.1±0.0	45.0±0.0	76.6±0.0			
	RDE(-MCD)	60.5±0.6	70.6±0.4	86.4±0.2	45.9±0.3	58.0±0.3	83.8±0.1	44.1±0.0	54.5±0.0	79.9±0.0			
	RDE	66.3±0.3	74.8±0.2	87.6±0.2	53.0±0.1	64.2±0.1	85.8±0.1	47.6±0.1	57.7±0.1	80.3±0.0			
RoBERTa	PPL	35.1±0.2	48.1±0.2	74.5±0.0	33.3±0.4	45.8±0.5	74.0±0.1	21.2±0.1	30.6±0.2	64.7±0.1			
	FGWS	61.4±0.4	71.2±0.3	73.7±0.2	80.1±0.3	83.4±0.2	86.2±0.1	36.7±0.4	47.7±0.4	60.5±0.1			
	MLE	41.8±0.3	54.7±0.2	84.2±0.1	31.6±0.4	44.0±0.4	81.5±0.2	37.0±0.0	48.0±0.0	77.7±0.0			
	RDE(-MCD)	62.5±0.6	72.1±0.5	87.5±0.3	51.9±0.6	63.3±0.5	84.7±0.1	45.6±0.3	56.1±0.2	79.2±0.1			
	RDE	73.3±0.6	79.6±0.4	90.4±0.2	65.7±0.3	73.9±0.2	88.5±0.1	50.9±0.1	60.6±0.1	80.2±0.1			

Table A.4: Adversarial detection results for BERT and RoBERTa on Scenario 2 on three datasets (IMDB, AG-News, SST-2). For all three metrics, higher means better.