SCAD: Subspace Clustering based Adversarial Detector

Anonymous Author(s)

ABSTRACT

Adversarial examples pose significant challenges for Natural Language Processing (NLP) model robustness, often causing notable performance degradation. While various detection methods have been proposed with the aim of differentiating clean and adversarial inputs, they often require fine-tuning with ample data, which is problematic for low-resource scenarios. To alleviate this issue, a **S**ubspace **C**lustering based **A**dversarial **D**etector (termed **SCAD**) is proposed in this paper, leveraging a union of subspaces to model the clean data distribution. Specifically, SCAD estimates feature distribution across *semantic subspaces*, assigning unseen examples to the nearest one for effective discrimination. The construction of semantic subspaces does not require many observations and hence ideal for the low-resource setting. The proposed algorithm achieves detection results better than or competitive with previous state-of-the-arts on a combination of three well-known text classification benchmarks and four attacking methods. Further empirical analysis suggests that SCAD effectively mitigates the low-resource setting where clean training data is limit.

CCS CONCEPTS

• **Computing methodologies** → **Anomaly detection**; **Natural language processing**; **Cluster analysis**.

KEYWORDS

Adversarial example detection, Sparse subspace clustering, Low-resource training, Model Robustness

ACM Reference Format:

Anonymous Author(s). 2024. SCAD: Subspace Clustering based Adversarial Detector. In *WSDM '24: The 17th ACM International Conference on Web Search And Data Mining, March 4 to March 8, 2024, Mérida, Mexico.* ACM, New York, NY, USA, 9 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

In recent years, Transformer-based models have been widely utilized in various fields. Despite the remarkable performance achieved, they remain susceptible to adversarial attacks [3, 11, 19]. These attacks manipulate clean/normal samples by introducing subtle perturbations, resulting in misleading outputs from the victim models [3, 11, 19]. In response to this issue, approaches such as adversarial defense and adversarial sample detection, aimed at mitigating the impact of adversarial samples, have been proposed [9, 13, 26, 29]. While the former focuses on achieving a robust model accuracy on both the clean and adversarial inputs [10, 18], the detection methods differentiate between adversarial and normal samples, subsequently discarding the adversarial ones during the inference stage [21, 29]. Our paper is on a novel method for adversarial example detection.

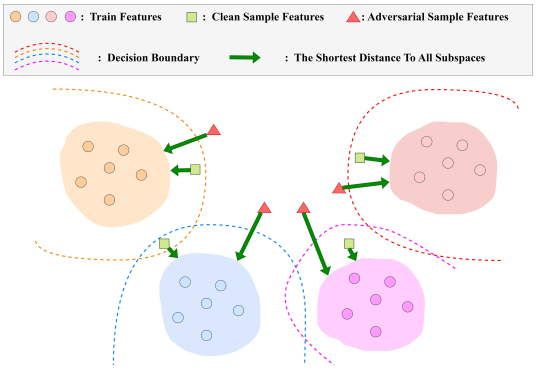


Figure 1: Illustration of the proposed SCAD. The four blocks of different colors indicate four subspaces of all clean data identified by subspace clustering algorithm. A probability density function is estimated for each cluster *separately***. Adversarial features as outliers tend to distribute in low probability regions, indicated by the union of the "outside" areas of the dotted curves as they curve towards the high probability regions which are the centers of the clusters.**

Existing detection methods can be broadly cast into three categories: 1) perturbation-based methods [4, 20–22], 2) inference-based methods [26, 31], and 3) distribution-based methods[13, 29]. Among them, perturbation-based techniques assess shifts in the victim model's response due to slight targeted perturbations, such as word replacement [21, 22]; inference-based methods scrutinize model properties, such as input loss landscape [31] or output uncertainty [26]; distribution-based methods characterize the clean data distribution, capitalizing on the tendency of adversarial samples to locate around decision boundaries.

While significant progress has been made in achieving remarkable performance, current methods remain heavily reliant on the victim model for subsequent detection tasks. This dependency often necessitates a sizable and meticulously labeled training dataset, posing challenges in real-world situations due to the resource-intensive and time-consuming nature of annotations. As a result, the low-resource setting with limited training data, leading to underperforming victim models and subsequent degradation in detection performance.

To the best of our knowledge, our research represents the first attempt to conduct adversarial example detection in low-resource scenarios. The idea is to identify a group of *semantic subspaces* such that the observations within each subspace have high coherence in terms of semantics while exhibiting necessary variation for diversity independent of observations from other subspaces. Withe the presence of severely limited training resources, our method exhibits

WSDM '24, March 4 to March 8, 2024, Mérida, Mexico

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/XXXXXXX.XXXXXXX

exceptional performance and efficacy under such testing circumstances. This is primarily attributed to the efficiency of constructing these semantic subspaces, which inherently demand fewer observations. The advantages of our approach include: 1) independence from the model for training and validation; 2) lower time consumption; 3) near-zero modifications required to adapt to different architectures. Experimental results, validated using a combination of three datasets, four attacks, and varying percetanges of training samples, demonstrate that our method offers superior performance both in regular scenarios and in low-resource scenarios[1].

2 RELATED WORK

2.1 Adversarial Detection

Given an input sample X, an adversarial sample X' is created by introducing a perturbation δ to X, resulting in $X' = X + \delta$. The purpose is to mislead the victim model f so that $f(X) \neq f(X')$. In practical scenarios, the perturbation δ is often imperceptible to human observers, but pose significant challenges to the robustness of the model.

Existing attack methods primarily focus on two aspects: character and word. Character-level attacks implement the perturbation δ to make X and X' indistinguishable visually, such as order swapping or substitution of characters [3]. On the other hand, word-level attacks generate δ by employing strategies such as word deletion, insertion, or substitution, while minimizing semantic dissimilarity between X and X' [5, 11].

Adversarial example detection (AED) is a efficient approach against attacks. Via training a binary classifier, AED distinguishes between normal and adversarial samples within the input data, and subsequently discard the perturbed ones during the inference stage. Existing detection methods often utilize the victim model to train the adversarial example detector, and there exists three categories of AED: perturbation-based [4, 20–22], inference-based [26, 31], and distribution-based methods.

Specifically, perturbation-based methods manipulate input samples by substituting, masking, or adding character/words. They then compare model outputs before and after these modifications to identify adversarial instances. **FGWS** [21] substitutes words using the disparity in word frequency between pre-collected clean and adversarial samples. **CHECKHARD** [22] replaces words with synonyms, while **WDR** [20] masks words based on their significance. Additionally, **UAPAD** [4] initially injects a random noise and refines it during the model fine-tuning.

Inference-based, on the other hand, only operates on the output level of the model. For instance, **ADDMU** [26] conducts detection through both the model and data uncertainty. The assumption is the model is uncertain about predictions with the adversarial examples. Sharpness estimates the input loss landscape from the geometric perspective, and concludes that the adversarial samples have a deep and sharp local minima on the input loss landscape.

Another line of research relies on the feature distribution, and **RDE** [29] and **MLE** [13] are typical examples. Specifically, RDE applies a parametric density estimation model to cluster samples with different labels. Later those clusters are regarded as representations of raw inputs to differentiate samples. In addition, MLE measures the probability density of testing sample on feature spaces using the Mahalanobis distance and achieve promising detection results.

It's evident that many current methods still rely on the victim model for training, potentially restricting their appilcation, particularly in low-resource scenarios. In contrast, our research introduces a novel perspective by leveraging semantic subspaces, which demand fewer observations and are thus well-suited for low-resource settings.

2.2 Semantic Subspace Clustering

As mentioned earlier, our approach is to divide data into several independent semantic groups where the distributions can be separately modelled. The semantic coherence can be expressed by the successful reconstruction of any observation by using others within the same group. This connects perfectly to subspace clustering in machine learning.

Subspace Clustering (SC) has attracted considerable research interest in machine learning. The basic assumption of SC is that the true data generating source is multiple subspaces and the observed data may be affected by additive noise. SC concerns recovering the underlying subspace structure of the data, and the data within one subspace may exhibit independent distribution characteristics to others. Sparse models sparked the explosion of methods for SC represented by sparse subspace clustering [2], many innovative methods were proposed to greatly improve SC outcomes for various data modalities [6, 25, 27, 28] and computational efficiency [7, 8].

The principle underlying sparse subspace clustering (SSC) is that for any observation $\mathbf{x}$ in a subspace of dimension d, only d peers are needed to reconstruct it perfectly in noise-free case. When the number of the total observation $N \gg d$, which is normally true, this implies sparsity in linear regression. To formalize this, let $\mathbf{x}_i \in \mathbb{R}^D$ be the ith observation living in the ambient space of dimension D, $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N] \in \mathbb{R}^{D\times N}$ the data matrix and $\mathbf{X}_{-j}$ the matrix with the jth column removed. Then for $\mathbf{x}_i$, SSC solves the following optimization problem

$$\min_{\mathbf{w}_i} \|\mathbf{w}_i\|_p \text{ s.t. } \mathbf{x}_i = \mathbf{X}_{-i}\mathbf{w}_i,\ \forall i = 1, \ldots, N$$

where $\mathbf{w}_i \in \mathbb{R}^{N-1}$ is the reconstruction coefficients vector and $\|\mathbf{w}_i\|_p$ is a sparsity encouraging ℓ_p norm such as ℓ_1 norm. In reality, noise is inevitable, in which case, an additive noise model is assumed to extract clean data:

$$\min_{\mathbf{w}_i} \|\mathbf{w}_i\|_p + \lambda\|\mathbf{e}_i\|_2^2 \text{ s.t. } \mathbf{x}_i = \mathbf{X}_{-i}\mathbf{w}_i + \mathbf{e}_i,\ \forall i = 1, \ldots, N$$

where $\mathbf{e}_i \in \mathbb{R}^D$ is the noise and $\lambda \geq 0$ is the regularisation to control the trade-off between the goodness-of-fit (represented as $\|\mathbf{e}_i\|$) and sparsity. This can be expressed in matrix form as

$$\min_{\mathbf{W}} \|\mathbf{W}\|_p + \lambda\|\mathbf{E}\|_2^2 \text{ s.t. } \mathbf{X} = \mathbf{X}\mathbf{W} + \mathbf{E}, \text{diag}(W) = 0, \qquad (1)$$

where $\mathbf{W} \in \mathbb{R}^{N\times N}$ and $E \in \mathbb{R}^{D\times N}$ are reconstruction coefficients matrix and noise matrix respectively, and $\text{diag}(W)$ is the diagonal vector of W and the corresponding condition is to prevent an observation from reconstructing itself. Note that in (1), the equality constraint reflects semantics coherence with deviations captured by error $\mathbf{E}$. There are numerous solvers for (1). Many of them were sought from compressive sensing research. W is interpreted as a new

[1] The source code will be made available upon acceptance.

representation of the original data that contains all subspace information. A traditional clustering algorithm such as k-means is applied to W to finalize the subspace membership of individual observations. We highlight that the identification of each subspace is not part of the SSC. One has to apply other methods such as principal component analysis (PCA) [12] for this purpose. Moreover, the number of subspaces k has to be given or estimated. We use the Eigen-gap [24] to compute k (the number of subspace) in this work, although it is still a vibrant research area in subspace clustering.

3 PROPOSED METHOD

We propose to use a union of subspaces to model the distribution of normal (clean data that are not attacked/perturbed), leading to a much preciser model than those in other works such as the one in [29]. The hypothesis is that the clean (normal) documents (their vectorized representations) reside in some *semantic subspaces* following some distributions such as multivariate Gaussian. While the attacked ones, which would be unusual in some sense and hence different to the cleans ones, are outliers that do not belong to any of the semantic subspaces. Fig. 1 pictorially illustrates this idea, the core of SCAD. Each color blob represents a semantic subspace with a probability distribution. The dotted curve of the same color of the semantic subspace shows the "outskirt" of the distribution, beyond which (further away from the center of the semantic subspace) the probability is very low. Each semantic subspace has such a "soft" boundary. Jointly, they form semantic bubbles for all clean documents. The adversarial samples are excluded from any of these bubbles and attract low probabilities. SCAD has the advantage of capable of modeling the total distribution of clean data in fine detail. Furthermore, it serves as a framework as it offers great flexibility, for example, the distributions within individual semantic subspaces can be totally independent. As a proof of concept, we start from simple assumption that the data within a semantic subspace is following Gaussian distribution. However, we point out that the joint distribution of all clean data is complex and it is not necessarily a mixture of Gaussian as the configuration of subspaces is arbitrary and their dimensionality could be very different.

SCAD starts from identifying semantic subspaces using available clean data using efficient algorithms such as [8]. After subspace clustering, the membership of each data point, *i.e*. the segmentation of subspaces, is known. Assume we have S total subspaces and the data for s subspace is written as $\mathbf{X}_s \in \mathbb{R}^{N_s \times D}$, where N_s is the number of observations in subspace s and D is the ambient space dimension, for example, 768 in BERT.

Based on Gaussian assumption, we run PCA for each subspace to find its structure. We outline the procedure here. Assume $\mathbf{Y} \in \mathbb{R}^{N \times D}$ and $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_N]^\top$ as the data matrix in certain subspace. Then

$$\mathbf{m} = \frac{1}{N} \sum_i \mathbf{y}_i$$

and

$$\Sigma = \frac{1}{N-1} \sum_i (\mathbf{y}_i - \mathbf{m})^\top (\mathbf{y}_i - \mathbf{m}).$$

The eigendecomposition of Σ gives all components of $\mathbf{Y}$, and the first d of them corresponding to the largest d eigenvalues are the PC's. Write the eigendecomposition of Σ as

$$\Sigma = \mathbf{U}\Lambda\mathbf{U}^\top$$

where $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_D]$ contains all *orthonormal* components and $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_D)$ is a diagnoal matrix with all eigenvalues assuming all eigenvalues are sorted by *decreasing* order. Then we have

$$d = \arg_k \min \frac{\sum_i^k \lambda_i}{\sum_i^D \lambda_i} \geq \tau$$

where $\tau \in (0, 1]$ is the pre-specified threshold called *variance explained* in PCA. Normally $\tau = 0.95$. So we retain the first d eigenvectors from $\mathbf{U}$ as the subspace coordinates centred at $\mathbf{m}$ derived from all observations in $\mathbf{Y}$. We write $\mathbf{P} = [\mathbf{u}_1, \ldots, \mathbf{u}_d]$. For any subspace s, we write $(\mathbf{P}_s, \mathbf{m}_s, \Sigma_s)$ as its coordinates, centre and covariance triplet. Under our assumption, data from subspace s follow multivariate Gaussian $\mathcal{N}(\mathbf{m}_s, \Sigma_s)$.

SCAD utilizes the following three steps to detect adversarial attack for a given new observation $\mathbf{x}$. The first is subspace assignment, *i.e*. finding the closest subspace l to $\mathbf{x}$ by

$$l = \arg_s \min \|(\mathbf{I} - \mathbf{P}_s\mathbf{P}_s^\top)(\mathbf{x} - \mathbf{m}_s)\|_F^2$$

where $\mathbf{I}$ is the identity matrix with compatible dimensions and $\|\cdot\|$ is the Frobenius norm. Second is to compute the likelihood of $\mathbf{x}$, *i.e*. $\mathcal{L}(\mathbf{x})$, being from subspace l. This is straightforward as

$$\mathcal{L}(\mathbf{x}) = p(\mathbf{x}'|\mathbf{m}_l, \Sigma_l)$$

where $\mathbf{x}' = \mathbf{P}_l\mathbf{P}_l^\top(\mathbf{x} - \mathbf{m}_l)$, $p(\cdot|\mathbf{m}, \Sigma)$ is the density function of Gaussian $\mathcal{N}(\mathbf{m}, \Sigma)$. Finally, we make decision on whether $\mathbf{x}$ is normal or attacked. It is manifested as a threshold γ, satisfying $p(\mathbf{0}|\mathbf{m}_l, \Sigma_l) \gg \gamma > 0$, such that $\mathbf{x}$ is an outlier, *i.e*. attacked, if $L(\mathbf{x}) > \gamma$ and normal otherwise.

PROPOSITION 1. *Given the assumption that the clean data are distributed within S subspaces following multivariate Gaussian, and the $\mathbf{x}$ is assigned to subspace l with dimension d_l, then for any $\gamma \leq e^{-\frac{d_l \log 2\pi}{2}}$ the success probability of SCAD, written as $\mathbb{P}\{SCAD\}$ is at least*

$$1 - S\left(2 - 2\Phi(\sqrt{-\frac{2}{d_l}\log\gamma - \log 2\pi})\right)^{d_l}, \tag{2}$$

where $\Phi(t) = \int_{-\infty}^{t} \frac{1}{2\pi} e^{-\frac{x^2}{2}} dx$, i.e. the cumulative probability function of standard Gaussian.

PROOF. We start by writing the success probability of SCAD

$$\begin{aligned}
\mathbb{P}\{\text{SCAD}\} &= \mathbb{P}\{\mathbf{x} \notin \mathcal{S}_1 \cap \mathbf{x} \notin \mathcal{S}_2 \cap \cdots \cap \mathbf{x} \notin \mathcal{S}_S\} \\
&= 1 - \mathbb{P}\{\cup_i^S \mathbf{x} \in \mathcal{S}_i\} \\
&\geq 1 - S \max_i \mathbb{P}\{\mathbf{x} \in \mathcal{S}_i\} \quad \text{(Union bound)} \\
&= 1 - S\mathbb{P}\{\mathbf{x} \in \mathcal{S}_l\} \quad (\mathbf{x} \in \mathcal{S}_l \text{ as in SCAD})
\end{aligned} \tag{3}$$

Now we need the tail bound on multivariate Gaussian. There are some works about tight tail bound. However, our purpose is to show $\mathbb{P}\{\text{SCAD}\}$ increases as γ decreases, and therefore we take a simple route by sphericalizing a Gaussian. Let $\tilde{\mathbf{x}}$ be sphericalized version of $\mathbf{x}$ such that

$$\mathcal{N}(\mathbf{x}|\mathbf{m}_l, \Sigma_l) = \mathcal{N}(\tilde{\mathbf{x}}|\mathbf{0}, \mathbf{I}) = r$$

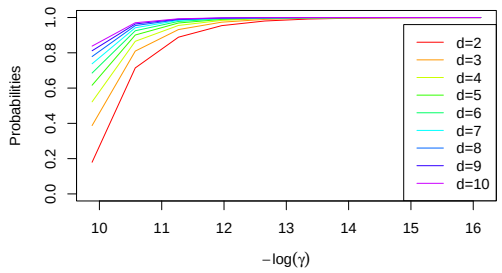


Figure 2: The function of $\mathbb{P}$(SCAD) vs $-\log(\gamma)$. We set $S = 5$ and let d_l vary from 2 to 10. When γ is the very small, $\mathbb{P}$(SCAD) is very close to 1.

for $r > 0$. From above, we derive

$$\mathbb{P}\{\mathbf{x} \in \mathcal{S}_l\} = \mathbb{P}(\|\tilde{\mathbf{x}}\| > r) \tag{4}$$

$$\leq \mathbb{P}(\forall i, |\tilde{x}_i| > r/\sqrt{d_l}) \tag{5}$$

$$= \prod_i \mathbb{P}(|\tilde{x}_i| \leq r/\sqrt{d_l})$$

$$= (1 - \mathbb{P}(-r/\sqrt{d_l} \leq \tilde{x}_i \leq r/\sqrt{d_l}))^{d_l} \tag{6}$$

$$= [2 - 2\Phi(r/\sqrt{d_l})]^{d_l} \tag{7}$$

The inequality (5) comes from expanding the integral region and $\tilde{x}_i$ is the ith element in $\tilde{\mathbf{x}}$. Independence in standardized Gaussian leads to (6) and (7). Next we connect r to the choice of γ in SCAD. From (4) we have

$$(2\pi)^{-d_l/2} \exp\{-\frac{1}{2}\tilde{\mathbf{x}}^\top \tilde{\mathbf{x}}\} = \gamma.$$

It is straightforward to see that

$$r = \sqrt{-2\log(\gamma) - d_l \log 2\pi}.$$

Substituting the above into (7) and (3) gives the probability bound as in the proposition description. □

REMARK 1. *The success probability of SCAD approaches 1 when γ approaches 0 as shown in Fig. 2. In other words, the smaller the value of γ,* i.e. *the threshold for the likelihood, the larger the probability of SCAD being successful. This provides a confidence level of SCAD. This coincides with the intuition illustrated in Fig. 1 so that the further away* $\mathbf{x}$ *is from all semantic subspaces, the more likely that it is attacked. From a desired probability p, we can work backwards to determine the value of γ*

$$\gamma = e^{\left\{-\frac{d_l}{2}\left(\left[\Phi^{-1}(1-e^{\frac{\log(1-p)-\log(S)}{d_l}}/2)\right]^2 + \log(2\pi)\right)\right\}}, \tag{8}$$

where $\Phi^{-1}(\cdot)$ is the quantile function of standard Gaussian, i.e. *the inverse of $\Phi(\cdot)$.*

Finally, we discuss the standardization of Gaussian. Let $\mathbf{A}$ be a linear transformation such that $\mathbf{A}\Sigma\mathbf{A}^\top = \mathbf{I}$. It is clear that

$$\mathbf{A} = \Lambda^{-\frac{1}{2}}\mathbf{U}^\top \tag{9}$$

where Λ collects all eigenvalues of Σ in its diagonal. If $\Lambda > 0$ strictly, i.e. no zero eigenvalues. $\Lambda^{-\frac{1}{2}}$ is element-wise, i.e. $\lambda_i^{-\frac{1}{2}}$. If some eigenvalues are zero, simply ignore then and leave them as zero in $\Lambda^{-\frac{1}{2}}$ because the corresponding eigenvectors will have zero projection. The standardisation transformation for $\mathbf{x}$ is

$$\tilde{\mathbf{x}} = \mathbf{A}(\mathbf{x} - \mathbf{m})$$

and then $\tilde{\mathbf{x}} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Put the result in the context of subspace l, we have

$$\mathbb{R}^{d_l} \ni \tilde{\mathbf{x}} = \Lambda_l^{-\frac{1}{2}}\mathbf{U}_l^\top(\mathbf{x}' - \mathbf{m}_l).$$

Now we can set the decision rule. Assume $d_l = 3$, $S = 5$. If we want

$$\mathbb{P}(\text{SCAD}) \geq 0.95,$$

by using (8), we can easily compute $\gamma = 0.006354996$. From we can obtain r as well, which is simply

$$r = \Phi^{-1}(1 - e^{\frac{\log(1-p)-\log(S)}{d_l}}/2).$$

Continue on our example, this connects to the geometric interpretation that within semantic subspace l we have

$$\|\tilde{\mathbf{x}}\| > r = 1.238735.$$

Therefore, with above values of γ and r, one can threshold using either γ on likelihood or r on projected and standardized version of of $\mathbf{x}$ in subspace l to obtain a detection probability of 0.95.

4 EXPERIMENTS

4.1 Setup

Datasets. Experiments are conducted on three text classification benchmarks (summarized in Table 1), including:

- **Stanford Sentiment Treebank 2 (SST-2)** [23]: this dataset is a well-regarded resource for sentiment analysis, focusing on fine-grained sentiment classification using the Stanford Sentiment Treebank corpus.
- **Yelp Reviews (YELP)**[30]: this dataset consists of reviews sourced from the Yelp platform, widely recognized for its extensive use in local business ratings and customer feedback.
- **Internet Movie Database (IMDB)**[17]: this dataset is specifically designed for document polarity classification, encompassing a collection of movie reviews collected from the IMDB database.

Table 1: Statistics for employed text-classification benchmarks.

Dataset	#Train	#Test	#Avg Length
SST-2	67,300	1,821	16
YELP	560,000	38,000	152
IMDB	25,000	25,000	161

Attacking algorithms. Four adversarial attacking methods are implemented using *TextAttack* [19] to pollute input sequences, that is [2],

[2] Pretrained attacking models can be accessed through https://huggingface.co/textattack. These models are then used directly for generating adversarial samples, following the approach outlined in [26, 29].

- **TextBugger**[14] performs perturbations of space insertion, char deletion/swapping, and synonym substitution;
- **DeepWordBug** [3] deletes, replaces, and inserts characters to inputs;
- **BERT-Attack** [15] substitutes key words using a pre-trained masked model;
- **TextFooler** [11] replaces important words with their synonyms;

Notably, the first two methods primarily focus on character-level attacks, whereas the latter two are mainly employed for word-level attacks. The diversity in adopted different attacking algorithms provides a variety of options to effectively explore the capability of detection models.

Implementation details. For the proposed SCAD, the BERT-base model [1] is employed as the context encoder. The dropout rate across all layers is set as 0.1. The Adam optimizer with a dynamic learning rate is adopted, for which the learning rate is warmed up for 10 thousand steps to a maximum value of $2e^{-5}$ before decaying linearly to a minimum value of $1e^{-6}$ and a gradient clip of (-1, 1). The batch size is set as 32 with the maximum length of 512 tokens for each input sequence. The training process is bounded by a maximum of 5 epochs.

Again, existing detection methods require fine-tuning the victim models before they can differentiate the normal and adversarial inputs. To investigate how the variations of training data sizes impact the performance and flexibility of the detection, we specifically consider the low-resource setting via limiting the scale of training data. That is, let r be the portion of samples randomly selected from the full dataset. We then systematically form five training subsets, commencing from r = 10% to 50%, incrementally increasing in size by 10% intervals. The proposed method is then applied on these data to form subspaces, while the Eigen-gap [24] is applied to compute the number of subspace (SST-2: 6, YELP:5, IMDB:8). At last, to make fair comparison we adopt the same inferring process presented in [26, 29]. That is, m clean samples are randomly selected from the testing dataset, out of which half are subjected to the attacking methods to produce adversarial data. For the employed SST-2, YELP, and IMDB dataset, m is set as 2000, 4000, and 4000, respectively. Detection methods are accordingly employed to differentiate the normal and attacked testing samples.

Evaluation metrics. Following recent work [20, 26, 29], three measurements are employed to assess the effectiveness of adversarial detection. Specifically, **True Positive Rate (TPR)** quantifies the ratio of accurately identified adversarial examples to the total number of samples predicted to be adversarial; **F1** evaluates the balance between precision and recall capabilities of the models; **Area Under the Curve (AUC)** provides an aggregated measure of model performances across all possible classification thresholds. Accordingly, higher values for TPR, F1, and AUC indicates the better detection accuracy.

4.2 Main results

To start, the vanilla model of BERT is fine-tuned with varying percentages of training samples (denoted by r). Then perturbed samples are generated via applying attacking methods on those fine-tuned models, except the case of r=100% where existing perturbed models are sourced directly from the official *TextAttack* [19]. Notably, the adversarial dataset could consist of both successful and unsuccessful attack samples. We focus on those successful samples hereafter, while an ablation study is conducted in a later session to further investigate the impact from unsuccessful ones.

Next, the proposed method is compared with several state-of-the-art approaches, including **MLE**[13], **FGWS**[21], **WDR**[20], **RDE**[29], and **ADDMU**[26]. It is important to note that existing detection models are fine-tuned with the full dataset (*i.e.*, r=100%). Hence, their performance concerning the entire dataset is directly obtained from original papers. However, to evaluate their performance in low-resource settings (*i.e.*, when r ranges from 10% to 50%), we re-run these methods using provided open-source codes while ensuring consistent key configurations for all hyper-parameters as reported.

The averaged results, over five runs, from the proposed method and state-of-the-arts, are presented in Table 2 and Table 3 for character-level and word-level attacks, respectively. Our proposed SCAD substantially outperforms all baseline methods, demonstrating superior performance across a combination of three benchmarks, four attacking methods, and varying percentages of training samples.

Specifically, when provided with the full-size training data (100%), SCAD surpasses other methods. This is exemplified in its performance on the Yelp dataset, where SCAD achieves a remarkable surge exceeding 15% in TPR and an advancement of over 10% in F1 score compared to the strongest baseline (ADDMU). Furthermore, in the low-resource setting (where training samples range from 10% to 50%), SCAD maintains its robust performance, especially in terms of TPR and F1 scores. These results essentially validate the robustness of SCAD, as the marginal change rate is merely 2.5% (r: 100% $\rightarrow$ 10%). This is significantly smaller than that of RDE (5.8%) and ADDMU (4.3%), highlighting the superior stability and consistency of our proposed method. We also implement the significance test for five runs via randomly selecting various seeds and performing the Student's T-test on each dataset. The averaged p-values obtained from SST-2, YELP, and IMDB are $6.19e^{-4}$, $9.18e^{-5}$, and $5.19e^{-5}$, respectively, which verifies the effectiveness of the proposed method.

Empirically, the proposed SCAD method not only demonstrates superiority in full-scale data samples but also showcases remarkable adaptability in low-resource settings. This is a critical advantage, as real-world scenarios often involve limited data availability and resource constraints. The capability to perform well even with a reduced number of training samples makes SCAD a promising approach for adversarial sample detection.

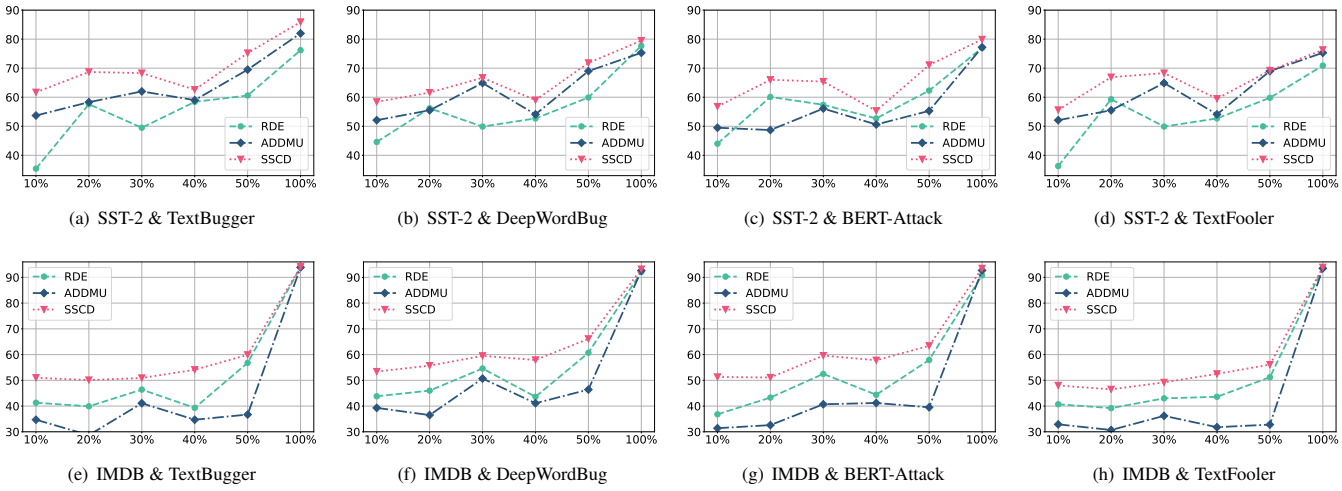
4.3 Ablation study

On the encoder flexibility. To start with, we evaluate the impact from the fundamental encoder towards the proposed method. Specifically, the RoBERTa encoder [16] is implemented, and the most strongest baselines, *i.e.*, RDE and ADDMU, are employed for comparison purposes. To make the fair comparison, again, we directly utilize pre-trained publicly-available attacking models of *TextAttack* [19], on SST-2 and IMDB, to generate the adversarial samples.

The performance comparison among the proposed method and two recent baselines is shown in Fig. 3. The proposed method demonstrates superior performance in terms of F1 score across all datasets,

Table 2: The comparison of various detection models in detecting character-level adversarial attacks (*i.e.*, TextBugger and DeepWordBug). Specifically, r represents the proportion of training samples utilized for fine-tuning the encoder, relative to the full dataset size. FGWS is excluded, as it is designed for detecting word-level attacks. The best performance is highlighted in bold, and the presented results are based on the average of five runs conducted with random seeds. Statistical significance testing at p-value < 0.01 (using T-test) are marked with †.

Attacks		Textbugger									Deepwordbug								
Dataset		SST-2			Yelp			IMDB			SST-2			Yelp			IMDB		
r	Methods	TPR	F1	AUC	TPR	F1	AUC	TPR	F1	AUC	TPR	F1	AUC	TPR	F1	AUC	TPR	F1	AUC
10%	MLE	42.1	54.7	75.6	58.3	67.1	77.5	58.6	64.7	80.6	35.7	52.9	75.8	62.4	71.9	82.5	54.6	65.8	84.9
	WDR	50.1	55.7	73.3	63.8	68.7	80.1	80.2	82.8	84.7	38.8	53.1	76.1	64.7	72.7	83.4	71.3	78.3	87.4
	RDE	63.5	72.4	86.2	67.3	74.7	86.3	84.8	83.7	87.2	44.6	57.3	81.4	69.4	73.3	89.4	78.3	82.5	89.3
	ADDMU	66.3	75.3	88.3	69.2	77.4	88.5	89.0	89.5	95.2	47.2	60.0	84.3	74.3	80.6	91.0	80.1	84.5	93.2
	SCAD	**74.9**†	**81.2**†	**92.2**†	**96.4**†	**94.1**†	**97.6**†	**91.5**†	**91.5**†	**95.9**†	**50.7**†	**63.1**†	**85.1**†	**96.8**†	**93.6**†	**97.2**†	**82.8**†	**85.9**†	**94.2**†
20%	MLE	53.5	59.7	78.3	60.7	68.6	80.8	60.7	68.3	81.5	44.7	57.6	77.7	63.8	72.6	84.1	63.9	72.4	86.5
	WDR	61.7	67.2	80.8	65.3	70.1	81.7	81.6	82.2	85.3	48.1	59.4	80.1	65.4	73.4	85.5	84.2	87.3	89.7
	RDE	64.1	74.8	86.1	68.6	76.1	87.2	86.2	85.3	88.3	53.6	65.6	82.6	70.1	75.6	90.1	87.3	89.2	91.6
	ADDMU	67.9	76.4	89.0	71.7	79.2	89.3	90.8	89.5	95.2	55.4	67.0	84.1	75.7	81.6	91.7	88.9	89.6	95.4
	SCAD	**74.8**†	**81.2**†	**92.4**†	**97.1**†	**94.0**†	**97.7**†	**92.6**†	**92.0**†	**96.2**†	**60.0**†	**70.7**†	**87.9**†	**97.1**†	**93.6**†	**97.1**†	**91.4**†	**90.7**†	**96.8**†
30%	MLE	53.2	60.8	80.5	61.1	70.3	81.9	62.1	69.8	82.6	43.2	56.7	76.7	64.2	73.8	85.4	62.4	71.1	85.8
	WDR	63.1	68.8	81.4	67.2	71.2	82.3	83.7	84.7	86.9	47.2	61.3	79.4	65.8	74.1	86.3	82.4	85.2	89.2
	RDE	68.1	77.3	87.7	70.3	77.6	88.6	87.5	86.9	89.8	52.4	64.6	82.2	70.8	76.3	89.9	85.7	88.7	91.8
	ADDMU	70.4	78.2	90.1	73.4	80.3	90.5	91.4	90.2	95.6	54.6	66.3	83.7	76.9	82.1	91.5	86.3	89.3	95.2
	SCAD	**74.7**†	**81.2**†	**91.7**†	**97.9**†	**93.9**†	**97.9**†	**93.5**†	**92.5**†	**96.8**†	**58.1**†	**69.2**†	**86.3**†	**97.0**†	**93.5**†	**97.3**†	**89.6**†	**90.1**†	**96.4**†
40%	MLE	54.6	62.2	81.7	62.7	71.6	82.6	64.2	70.3	84.5	50.2	62.9	82.4	65.7	74.7	86.1	63.2	71.7	86.2
	WDR	64.6	70.7	83.7	69.1	73.4	83.9	85.4	85.3	87.2	51.1	64.6	82.7	67.1	76.3	88.9	87.7	87.1	90.7
	RDE	70.1	76.3	89.3	71.8	78.3	89.3	91.2	88.1	90.1	55.3	67.8	85.3	72.4	77.9	90.2	90.1	90.8	92.1
	ADDMU	73.4	79.8	91.8	75.2	81.4	91.3	94.7	92.6	95.8	61.4	71.4	86.2	79.4	83.6	91.6	90.5	91.2	95.3
	SCAD	**76.5**†	**82.1**†	**92.1**†	**98.3**†	**94.2**†	**97.8**†	**95.2**†	**93.1**†	**97.2**†	**64.5**†	**74.1**†	**88.1**†	**97.2**†	**93.7**†	**97.8**†	**93.7**†	**92.3**†	**97.3**†
50%	MLE	59.6	68.4	82.6	63.8	72.3	83.6	66.4	72.1	85.2	54.1	65.4	84.1	67.2	76.5	88.6	66.7	74.5	87.4
	WDR	67.7	73.5	84.1	70.9	74.1	84.4	86.3	86.4	88.9	55.7	66.7	85.5	69.5	77.8	89.5	89.1	88.4	91.1
	RDE	72.5	79.1	90.7	74.2	79.5	89.7	92.6	89.5	90.7	57.2	69.3	86.4	73.7	79.5	90.8	93.0	90.5	92.6
	ADDMU	75.3	81.3	92.2	77.5	82.7	91.9	95.2	92.9	96.0	64.7	72.7	87.8	80.1	84.3	92.9	93.2	91.8	96.6
	SCAD	**81.0**†	**84.9**†	**93.5**†	**99.1**†	**95.9**†	**98.9**†	**96.8**†	**94.2**†	**97.9**†	**67.1**†	**75.8**†	**89.3**†	**97.5**†	**94.1**†	**98.0**†	**96.8**†	**93.7**†	**97.4**†
100%	MLE	67.6	72.3	84.3	61.8	70.6	82.7	67.9	73.4	86.6	53.7	64.2	83.3	65.8	74.9	87.4	64.3	72.5	86.8
	WDR	70.7	77.8	88.4	63.2	72.8	87.2	87.7	87.3	89.7	53.8	63.6	84.1	66.4	75.4	88.1	87.7	87.6	90.1
	RDE	72.4	79.6	89.6	66.2	75.2	89.2	93.4	90.1	91.0	55.5	67.0	86.1	70.5	78.1	90.4	90.6	90.8	92.0
	ADDMU	73.3	80.0	90.9	70.8	78.3	91.0	95.7	93.1	97.5	62.1	71.0	87.4	78.2	83.2	92.2	90.8	90.5	95.3
	SCAD	**75.3**†	**81.3**†	**91.5**†	**98.6**†	**94.6**†	**98.0**†	**96.1**†	**93.3**†	**97.6**†	**63.5**†	**73.3**†	**87.9**†	**95.7**†	**93.1**†	**97.1**†	**93.4**†	**91.8**†	**96.0**†



(a) SST-2 & TextBugger (b) SST-2 & DeepWordBug (c) SST-2 & BERT-Attack (d) SST-2 & TextFooler

(e) IMDB & TextBugger (f) IMDB & DeepWordBug (g) IMDB & BERT-Attack (h) IMDB & TextFooler

Figure 3: Impact analysis from the underlying encoder (*i.e.*, RoBERTa) on the adversarial detection.

Table 3: The comparison of various detection models in detecting word-level adversarial attacks (*i.e.*, Bert-Attack and TextFooler). Specifically, r represents the proportion of training samples utilized for fine-tuning the encoder, relative to the full dataset size. The best performance is highlighted in bold, and the presented results are based on the average of five runs conducted with random seeds. Statistical significance testing at p-value < 0.01 (using T-test) are marked with †.

Attacks		Bert-Attack									TextFooler								
Dataset		SST-2			Yelp			IMDB			SST-2			Yelp			IMDB		
r	Methods	TPR	F1	AUC	TPR	F1	AUC	TPR	F1	AUC	TPR	F1	AUC	TPR	F1	AUC	TPR	F1	AUC
10%	MLE	48.8	58.1	73.4	68.1	75.1	87.4	56.8	64.1	74.1	38.7	55.9	75.4	36.8	49.1	60.8	78.1	80.7	70.1
	FGWS	32.6	43.5	66.7	61.9	72.1	72.4	42.6	57.3	61.2	28.2	40.7	73.6	60.9	66.4	73.3	60.7	69.4	75.6
	WDR	42.1	48.9	74.1	70.6	78.2	89.2	60.4	72.1	82.4	30.5	43.3	74.3	63.6	70.4	77.1	76.3	78.4	80.4
	RDE	62.6	71.9	85.2	74.7	82.1	90.2	76.2	80.1	89.4	58.8	68.4	83.8	66.7	74.3	87.8	86.6	87.4	93.7
	ADDMU	64.1	73.8	87.5	78.5	83.5	91.5	78.6	82.8	90.4	60.9	70.2	85.6	72.1	78.4	90.3	89.0	89.5	95.2
	SCAD	**68.0**†	**76.4**†	**88.5**†	**97.3**†	**93.9**†	**97.7**†	**80.1**†	**84.3**†	**91.8**†	**61.3**†	**71.6**†	**86.5**†	**97.3**†	**94.0**†	**98.2**†	**90.8**†	**90.4**†	**95.6**†
20%	MLE	53.2	42.8	75.4	68.6	75.7	87.9	73.4	80.1	88.7	43.9	58.1	77.2	37.2	50.4	62.6	80.3	82.4	72.4
	FGWS	50.8	54.1	74.3	62.7	72.7	73.1	52.1	63.7	67.1	58.4	69.3	75.8	62.6	68.7	75.8	72.4	77.4	78.3
	WDR	57.5	70.9	79.5	68.1	77.0	88.4	65.9	76.3	84.2	60.3	70.4	78.5	64.2	71.2	79.8	83.4	84.6	86.3
	RDE	69.3	77.3	86.7	73.4	81.3	89.5	88.4	87.6	90.8	64.7	74.7	85.1	67.8	75.9	87.9	90.1	89.7	93.8
	ADDMU	71.3	78.0	88.6	76.2	82.1	91.1	90.1	90.0	92.4	67.8	76.3	87.6	73.5	80.4	90.9	92.4	91.3	94.2
	SCAD	**73.4**†	**80.3**†	**89.7**†	**96.8**†	**93.7**†	**97.6**†	**91.3**†	**91.2**†	**94.3**†	**68.3**†	**77.5**†	**88.0**†	**97.5**†	**94.1**†	**98.8**†	**94.6**†	**92.5**†	**95.3**†
30%	MLE	54.6	43.7	76.6	68.1	75.2	87.7	75.1	81.4	90.8	45.3	59.8	79.7	37.7	51.3	63.1	81.2	83.7	73.6
	FGWS	51.1	56.2	72.7	62.6	72.1	72.4	56.7	65.1	68.9	62.1	73.9	77.8	63.1	69.2	77.5	75.4	80.4	80.4
	WDR	58.8	71.1	80.8	67.8	76.7	88.2	67.2	77.1	86.4	63.4	74.4	81.1	65.6	72.5	80.8	84.1	85.7	87.1
	RDE	70.7	78.3	87.3	72.7	80.1	89.6	90.4	89.2	92.1	65.5	74.9	85.3	68.2	76.6	88.2	90.4	90.1	94.8
	ADDMU	73.2	80.5	87.1	75.4	81.7	90.7	92.1	91.1	93.7	70.3	78.1	86.8	74.7	81.2	91.1	92.7	91.0	95.6
	SCAD	**75.8**†	**81.6**†	**89.9**†	**96.6**†	**93.6**†	**97.9**†	**93.7**†	**92.3**†	**95.3**†	**72.5**†	**79.5**†	**88.9**†	**98.6**†	**94.7**†	**98.7**†	**94.0**†	**92.1**†	**96.0**†
40%	MLE	54.8	61.3	77.6	67.4	74.8	87.3	76.4	82.1	91.3	44.3	58.4	78.2	39.1	52.5	65.1	83.8	85.4	74.9
	FGWS	50.5	58.1	71.4	62.2	71.7	72.2	58.4	67.4	70.4	61.7	73.1	76.1	65.4	70.2	78.7	80.3	84.6	82.6
	WDR	57.6	70.9	80.7	67.2	76.1	87.8	67.4	77.6	87.7	65.9	73.5	82.2	67.9	72.8	81.6	85.4	86.8	88.4
	RDE	70.1	75.7	85.7	71.9	78.8	89.1	91.7	89.4	92.7	63.8	73.7	84.0	69.8	77.4	88.7	91.3	90.9	95.2
	ADDMU	72.1	78.6	88.2	74.1	80.2	90.5	93.1	91.3	94.3	68.3	76.8	85.9	75.3	81.7	91.7	92.8	91.4	95.8
	SCAD	**74.8**†	**81.0**†	**90.0**†	**96.1**†	**93.4**†	**97.8**†	**94.4**†	**92.7**†	**95.8**†	**70.6**†	**78.2**†	**88.4**†	**98.4**†	**95.7**†	**98.7**†	**93.5**†	**92.1**†	**96.2**†
50%	MLE	55.1	63.6	78.5	69.4	76.3	89.1	78.2	83.7	91.6	48.2	60.5	81.7	40.3	53.9	65.7	85.1	86.7	75.6
	FGWS	51.4	61.5	72.3	63.9	72.8	73.1	59.6	70.8	71.6	63.2	72.4	77.1	66.7	71.4	79.3	82.1	86.4	85.7
	WDR	57.3	70.3	81.2	68.8	78.7	87.5	68.9	80.4	88.4	65.8	75.2	83.2	68.4	73.8	82.1	87.6	88.7	90.7
	RDE	70.7	76.2	86.5	72.5	79.8	89.6	92.5	90.5	93.3	64.2	74.1	84.2	70.2	78.4	89.5	94.6	92.0	96.8
	ADDMU	73.0	79.8	87.6	75.8	81.6	90.7	94.2	92.3	95.9	70.1	77.9	86.8	76.8	82.3	92.1	95.2	92.4	97.1
	SCAD	**75.6**†	**81.7**†	**89.9**†	**96.7**†	**93.9**†	**98.0**†	**95.7**†	**93.1**†	**97.0**†	**72.1**†	**79.4**†	**88.7**†	**98.3**†	**94.9**†	**98.7**†	**96.4**†	**93.5**†	**97.5**†
100%	MLE	56.8	64.3	79.4	68.7	75.3	88.7	76.9	82.3	91.4	33.3	46.5	79.8	41.3	54.6	66.5	86.3	87.9	76.9
	FGWS	52.8	63.7	73.6	63.5	72.6	72.1	58.9	69.3	70.6	62.9	72.8	76.5	67.1	72.7	80.6	84.6	87.1	87.1
	WDR	59.2	72.3	83.5	67.5	77.4	86.9	67.8	78.2	87.1	63.4	73.4	82.6	69.6	76.3	83.4	89.3	90.2	91.2
	RDE	74.6	80.9	90.3	72.0	79.2	90.2	89.3	88.6	92.7	62.9	72.8	86.5	72.0	79.2	89.6	96.6	93.5	97.7
	ADDMU	77.7	82.8	92.3	74.6	80.8	91.4	91.3	90.8	95.7	67.1	75.8	**88.8**	78.7	83.5	91.6	97.0	93.9	97.7
	SCAD	**80.2**†	**85.7**†	**92.7**†	**97.7**†	**94.1**†	**97.6**†	**92.2**†	**91.2**†	**96.0**†	**68.1**†	**76.6**†	**88.8**†	**98.1**†	**94.3**†	**97.1**†	**97.1**†	**94.2**†	**98.1**†

outperforming both RDE and ADDMU. This result aligns with the pattern observed in Table 2 and 3, where our approach consistently outperforms the competing methods.

Furthermore, this comparison strengthens the evidence for the stability and effectiveness of our approach using different underlying encoders. That is, our method outperforms the current best models, RDE and ADDMU, utilizing either RoBERTa or BERT as the encoder. This consistent advantage across different encoders reinforces the robustness and generalizability of our proposed method. Without explicitly mentioning, the following ablation studies are conducted using the BERT-base encoder.

On the Far-Boundary attacking. The Far-Boundary (FB) attack, introduced in [26], generates adversarial samples that are deliberately pushed far away from the model decision boundaries. These FB examples pose an even greater challenge for detection models compared to regular adversarial samples, as the detection performance is significantly worse than random guessing [26]. To ensure a fair comparison, we reproduce the FB adversarial samples using the Textfooler and Textbugger attacking methods on both the SST-2 and YELP datasets, following the methodology outlined in ADDMU[26]. Additionally, we evaluate our model performance in scenarios where only 30% and the full-scale of training samples are available. The inclusion of FB adversarial samples and low-resource settings allows us to thoroughly assess the effectiveness of the proposed SCAD method under hard detection conditions.

The average results from five runs are reported in Table 4, showcasing the outstanding performance of SCAD in detecting FB adversarial samples across low and full resource scenarios. SCAD consistently outperforms other baseline methods, demonstrating its effectiveness in handling challenging and distant-from-boundary attacks. One possible reason for SCAD's superior performance is the independence of class labels, unlike other baseline methods that rely on categorizing detection samples into various class categories. This allows SCAD to detect FB samples with minimal probability deviation from normal samples, making it more robust in detecting

Table 4: Performance comparison of the proposed method and current SOTAs on Far-Boundary (FB) adversarial examples detection.

SST-2		Textfooler-FB			Textbugger-FB		
r	**Methods**	**TPR**	**F1**	**AUC**	**TPR**	**F1**	**AUC**
30%	RDE	29.2	45.3	73.7	42.1	55.7	71.8
	ADDMU	45.8	58.8	82.5	48.6	61.4	81.5
	SCAD	**58.9**	**69.7**	**84.4**	**65.0**	**74.3**	**89.1**
100%	RDE	31.9	45.0	81.5	29.5	42.5	81.1
	ADDMU	62.0	72.2	88.0	50.5	62.9	86.1
	SCAD	**67.6**	**74.1**	**90.8**	**53.5**	**64.6**	**90.3**
YELP		**Textfooler-FB**			**Textbugger-FB**		
r	**Methods**	**TPR**	**F1**	**AUC**	**TPR**	**F1**	**AUC**
30%	RDE	38.0	51.4	75.6	43.8	57.0	80.8
	ADDMU	62.8	67.2	86.4	64.5	71.9	83.9
	SCAD	**98.1**	**94.6**	**97.9**	**98.7**	**94.7**	**98.4**
100%	RDE	31.5	44.6	82.7	63.9	73.5	88.4
	ADDMU	72.8	79.7	89.7	74.8	81.0	90.8
	SCAD	**97.6**	**94.1**	**96.8**	**98.5**	**94.6**	**97.3**

adversarial samples that may be significantly far away from the model's decision boundaries.

Overall, results from Table 4 further reinforce SCAD's effectiveness and reliability as a powerful detection method against adversarial attacks, particularly when faced with challenging FB adversarial samples in various resource scenarios.

On unsuccessful adversarial examples. While the above scenarios investigate the performance on successful adversarial examples, hereafter we consider those unsuccessful-attack samples. These special samples represent a more challenging scenario for detection methods and closely resemble real-world settings [4]. Hence we evaluate the proposed method capability to detect those harder samples in this section. Specifically, we compare SCAD with current SOTA methods (such as UAPAD), and present results of F1 scores (with r=100%) in Table 5[3].

Table 5: Comparison of detection results on unsuccessful adversarial examples.

F1	SST-2		IMDB	
Methods	Textfooler	Bert-Attack	Textfooler	Bert-Attack
RDE	73.2	81.0	73.2	81.3
UAPAD	**80.9**	85.1	76.8	78.0
SCAD (ours)	76.0	**85.4**	**92.8**	**90.4**

Empirically, the performance of all detection methods exhibits a notable decrease when faced with unsuccessful adversarial examples, as exemplified by RDE using the IMDB dataset. For instance, the F1 scores for Textfooler and Bert-Attack are reduced from 93.5 to 88.6 and 73.2 and 81.3, respectively. Moreover, our proposed method demonstrates a competitive performance, outperforming other approaches in three out of the four scenarios evaluated. This indicates its robustness and potential for practical application in detecting even unsuccessful adversarial samples.

Complexity. At last, we compare the computational efficiency of various detection methods. Specifically, Table 6 shows the GPU time required for training with 1000 and detecting 1500 samples, using the SST-2 and IMDB datasets.

Particularly, FGWS achieves the quickest inference time by simply substituting low-frequency words with high-frequency alternatives during testing. In contrast, ADDMU necessitates iterative forward propagations to compute both model and data uncertainty. Comparable to ours in computational expense, RDE incorporates additional steps like kPCA and MCD for training and estimating the data Gaussian distribution. Overall, SCAD offers a computationally affordable solution, exhibiting efficient time consumption in both training and detection stages.

Table 6: Computational efficiency of various adversarial detectors, for training 1000 samples and inferring 1500 samples, in terms of the GPU calculation time (seconds).

Methods	Training		Testing	
	SST-2	**IMDB**	**SST-2**	**IMDB**
FGWS	52.7	63.1	4.17	5.58
WDR	86.4	99.4	21.4	27.8
RDE	42.7	56.6	76.8	102.3
ADDMU	207.3	251.8	327.8	369.4
SCAD	**40.4**	**53.7**	**12.2**	**13.1**

5 CONCLUSION

We introduced a novel adversarial example detection (AED) approach, the Subspace Clustering based Adversarial Detector (SCAD). While the majority of existing AED methods rely on fine-tuning the victim model, which can constrain their effectiveness, our method takes a different route. The proposed SCAD estimates feature distribution across semantic subspaces and categorizes unseen examples into the nearest one for detection. The main idea is observations within each subspace exhibit strong semantic coherence and essential diversity. Our experimental results, conducted across three datasets using four attack methods, showcased SCAD outperforming existing approaches with superior performance. Notably, SCAD also effectively addresses the challenge of low-resource scenarios where clean training data is scarce.

ETHICAL STATEMENT

This work is generally considered ethically unproblematic, as all datasets and models employed in this study are openly accessible to the public. Nevertheless, we acknowledge the need to address the possibility of subtle biases that might arise from the use of Pre-trained Language Models (PLMs) as encoders for generating the latent representation of tables and queries. These PLMs might inherit biases present in the data they were trained on. However, it is important to note that during our analysis, we did not identify any concerning outcomes related to bias.

[3] As no source codes released by UAPAD, their results are directly sourced from the original paper.

REFERENCES

[1] 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* (2018).

[2] Ehsan Elhamifar and Rene Vidal. 2013. Sparse Subspace Clustering: Algorithm, Theory, and Applications. https://doi.org/10.48550/arXiv.1203.1005 arXiv:1203.1005 [cs, math, stat].

[3] Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. 2018. Black-Box Generation of Adversarial Text Sequences to Evade Deep Learning Classifiers. *2018 IEEE Security and Privacy Workshops (SPW)* (2018), 50–56.

[4] SongYang Gao, Shihan Dou, Qi Zhang, Xuanjing Huang, Jin Ma, and Ying Shan. 2023. On the Universal Adversarial Perturbations for Efficient Data-free Adversarial Detection. In *Findings of the Association for Computational Linguistics: ACL 2023*. Association for Computational Linguistics, Toronto, Canada, 13573–13581. https://aclanthology.org/2023.findings-acl.857

[5] Siddhant Garg and Goutham Ramakrishnan. 2020. BAE: BERT-based Adversarial Examples for Text Classification. *ArXiv* abs/2004.01970 (2020).

[6] Yi Guo, Junbin Gao, and Feng Li. 2014. Spatial subspace clustering for drill hole spectral data. *Journal of Applied Remote Sensing* 8, 1 (April 2014), 083644. https://doi.org/10.1117/1.JRS.8.083644

[7] Yi Guo, Junbin Gao, and Feng Li. 2015. Random spatial subspace clustering. *Knowledge-Based Systems* 74 (Jan. 2015), 106–118. https://doi.org/10.1016/j.knosys.2014.11.006

[8] Yi Guo, Stephen Tierney, and Junbin Gao. 2021. Efficient sparse subspace clustering by nearest neighbour filtering. *Signal Processing* 185 (Aug. 2021), 108082. https://doi.org/10.1016/j.sigpro.2021.108082

[9] Xinrong Hu, Ce Xu, Junlong Ma, Zijian Huang, Jie Yang, Yi Guo, and Johan Barthelemy. 2023. [MASK] Insertion: a robust method for anti-adversarial attacks. In *Findings of the Association for Computational Linguistics: EACL 2023*. 1028–1040.

[10] Xinrong Hu, Ce Xu, Junlong Ma, Zijian Huang, Jie Yang, Yi Guo, and Johan Barthelemy. 2023. [MASK] Insertion: a robust method for anti-adversarial attacks. In *Findings of the Association for Computational Linguistics: EACL 2023*. Association for Computational Linguistics, Dubrovnik, Croatia, 1058–1070. https://aclanthology.org/2023.findings-eacl.78

[11] Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2019. Is BERT Really Robust? A Strong Baseline for Natural Language Attack on Text Classification and Entailment. In *AAAI Conference on Artificial Intelligence*.

[12] M. Jolliffe. 1986. *Principal Component Analysis*. Springer-Verlag, New York.

[13] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. 2018. A Simple Unified Framework for Detecting Out-of-Distribution Samples and Adversarial Attacks. In *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Eds.), Vol. 31. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2018/file/abdeb6f575ac5c6676b747bca8d09cc2-Paper.pdf

[14] Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. 2018. TextBugger: Generating Adversarial Text Against Real-world Applications. *ArXiv* abs/1812.05271 (2018).

[15] Linyang Li, Ruotian Ma, Qipeng Guo, X. Xue, and Xipeng Qiu. 2020. BERT-ATTACK: Adversarial Attack against BERT Using BERT. *ArXiv* abs/2004.09984 (2020).

[16] 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* (2019).

[17] 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*. Association for Computational Linguistics, Portland, Oregon, USA, 142–150. https://aclanthology.org/P11-1015

[18] Zhao Meng, Yihan Dong, Mrinmaya Sachan, and Roger Wattenhofer. 2022. Self-Supervised Contrastive Learning with Adversarial Perturbations for Defending Word Substitution-based Attacks. In *Findings of the Association for Computational Linguistics: NAACL 2022*. Association for Computational Linguistics, Seattle, United States, 87–101. https://doi.org/10.18653/v1/2022.findings-naacl.8

[19] John Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020. TextAttack: A Framework for Adversarial Attacks, Data Augmentation, and Adversarial Training in NLP. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. 119–126.

[20] Edoardo Mosca, Shreyash Agarwal, Javier Rando-Ramirez, and Georg Groh. 2022. " That Is a Suspicious Reaction!": Interpreting Logits Variation to Detect NLP Adversarial Attacks. *arXiv preprint arXiv:2204.04636* (2022).

[21] Maximilian Mozes, Pontus Stenetorp, Bennett Kleinberg, and Lewis D Griffin. 2020. Frequency-guided word substitutions for detecting textual adversarial examples. *arXiv preprint arXiv:2004.05887* (2020).

[22] Hoang-Quoc Nguyen-Son, Huy Quang Ung, Seira Hidano, Kazuhide Fukushima, and Shinsaku Kiyomoto. 2022. CheckHARD: Checking Hard Labels for Adversarial Text Detection, Prediction Correction, and Perturbed Word Suggestion. In *Findings of the Association for Computational Linguistics: EMNLP 2022*. Association for Computational Linguistics, Abu Dhabi, United Arab Emirates, 2903–2913. https://aclanthology.org/2022.findings-emnlp.210

[23] 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*. Association for Computational Linguistics, Seattle, Washington, USA, 1631–1642. https://aclanthology.org/D13-1170

[24] Mahdi Soltanolkotabi, Ehsan Elhamifar, and Emmanuel J. Candès. 2014. Robust subspace clustering. *The Annals of Statistics* 42, 2 (April 2014). https://doi.org/10.1214/13-AOS1199

[25] Stephen Tierney, Junbin Gao, and Yi Guo. 2014. Subspace Clustering for Sequential Data. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, Columbus, OH, USA, 1019–1026. https://doi.org/10.1109/CVPR.2014.134

[26] Fan Yin, Yao Li, Cho-Jui Hsieh, and Kai-Wei Chang. 2022. ADDMU: Detection of Far-Boundary Adversarial Examples with Data and Model Uncertainty Estimation. *arXiv preprint arXiv:2210.12396* (2022).

[27] Ming Yin, Junbin Gao, Zhouchen Lin, Qinfeng Shi, and Yi Guo. 2015. Dual Graph Regularized Latent Low-Rank Representation for Subspace Clustering. *IEEE Transactions on Image Processing* 24, 12 (Dec. 2015), 4918–4933. https://doi.org/10.1109/TIP.2015.2472277

[28] Ming Yin, Junbin Gao, Shengli Xie, and Yi Guo. 2019. Multiview Subspace Clustering via Tensorial t-Product Representation. *IEEE Transactions on Neural Networks and Learning Systems* 30, 3 (March 2019), 851–864. https://doi.org/10.1109/TNNLS.2018.2851444

[29] 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*. Association for Computational Linguistics, Dublin, Ireland, 3656–3672. https://doi.org/10.18653/v1/2022.findings-acl.289

[30] Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems* 28 (2015).

[31] Rui Zheng, Shihan Dou, Yuhao Zhou, Qin Liu, Tao Gui, Qi Zhang, Zhongyu Wei, Xuanjing Huang, and Menghan Zhang. 2023. Detecting Adversarial Samples through Sharpness of Loss Landscape. In *Findings of the Association for Computational Linguistics: ACL 2023*. Association for Computational Linguistics, Toronto, Canada, 11282–11298. https://aclanthology.org/2023.findings-acl.717