Con-Detect: Detecting Adversarially Perturbed Natural Language Inputs to Deep Classifiers Through Holistic Analysis

This paper was downloaded from TechRxiv (<https://www.techrxiv.org>).

LICENSE

CC BY 4.0

SUBMISSION DATE / POSTED DATE

03-03-2022 / 08-03-2022

CITATION

Ali, Hassan; Khan, Muhammad Suleman; AlGhadhban, Amer; Alazmi, Meshari; Alzamil, Ahmad; AlUtaibi, Khaled; et al. (2022): Con-Detect: Detecting Adversarially Perturbed Natural Language Inputs to Deep Classifiers Through Holistic Analysis. TechRxiv. Preprint. <https://doi.org/10.36227/techrxiv.19295534.v1>

DOI

[10.36227/techrxiv.19295534.v1](https://doi.org/10.36227/techrxiv.19295534.v1)

Con-Detect: Detecting Adversarially Perturbed Natural Language Inputs to Deep Classifiers Through Holistic Analysis

Hassan Ali^{* ‡}, Muhammad Suleman Khan^{*}, Amer AlGhadhban[†], *Member, IEEE* Meshari Alazmi[§], Ahmad Alzamil[†], Khaled Al-utaibi[¶], *Senior Member, IEEE* and Junaid Qadir^{*}, *Senior Member, IEEE*

Abstract—Deep Learning (DL) algorithms have shown wonders in many Natural Language Processing (NLP) tasks such as language-to-language translation, spam filtering, fake-news detection, and comprehension understanding. However, research has shown that the adversarial vulnerabilities of deep learning networks manifest themselves when DL is used for NLP tasks. Most mitigation techniques proposed to date are supervised—relying on adversarial retraining to improve the robustness—which is impractical. This work introduces a novel, unsupervised detection methodology for detecting adversarial inputs to NLP classifiers. In summary, we note that minimally perturbing an input to change a model’s output—a major strength of adversarial attacks—is a weakness that leaves unique statistical marks reflected in the cumulative contribution scores of the input. Particularly, we show that the *cumulative contribution score*, called C_F -score, of adversarial inputs is generally greater than that of the clean inputs. We thus propose *Con-Detect*—a Contribution based Detection method—for detecting adversarial attacks against NLP classifiers. *Con-Detect* can be deployed with any classifier without having to retrain it. We experiment with multiple attackers—Text-bugger, Text-fooler, PWWS—on several architectures—MLP, CNN, LSTM, Hybrid CNN-RNN, BERT—trained for different classification tasks—IMDB sentiment classification, fake-news classification, AG news topic classification—under different threat models—*Con-Detect*-blind attacks, *Con-Detect*-aware attacks, and *Con-Detect*-adaptive attacks—and show that *Con-Detect* can reduce the attack success rate (ASR) of different attacks from 100% to as low as 0% for the best cases and $\approx 70\%$ for the worst case. Even in the worst case, we note a 100% increase in the required number of queries and a 50% increase in the number of words perturbed, suggesting that *Con-Detect* is hard to evade.

Keywords—machine learning security, adversarial detection, adversarial machine learning, secure natural language processing, adversarial signatures.

I. INTRODUCTION

Machine Learning (ML) techniques, particularly Deep Neural Networks (DNNs), have scored applications in autonomous driving [1], [2], medical image analysis [3], healthcare, and IoT devices due to their ability to generalize to complex tasks requiring intelligent decision making (e.g., classification,

generation and regression). One such example is Natural Language Processing (NLP), which exists in myriad forms such as spam filtering, emotion recognition, language-to-language translation, sentence generation and answering questions to name a few. However, DNNs are over-dependent on large training data, which makes them vulnerable to attacks at both the training [4] and the post-training stages [1], [5].

Recent works show that slight perturbations to an input can cause DNNs to misclassify these inputs [6]–[8]. Such a perturbation is known as the adversarial attack, and DNNs are fooled by these attacks for a range of applications in image processing [1], speech recognition [9] and NLP domains [10]. This work particularly focuses on the adversarial attacks on NLP classifiers. Morris et al. [11] note that adversarial attacks on NLP models generally identify and perturb the most contributing words in an input. Authors thus unify 16 different state-of-the-art attacks under a framework, the *TextAttack* library, to benchmark the robustness of NLP models.

Researchers are exploring two major strategies to counter the adversarial threat—detecting adversarial inputs, and defending the models by correctly classifying adversarial inputs. Although defending the model is generally preferable, *detecting adversarial inputs in itself is quite challenging* [5]. To illustrate this, Carlini and Wagner [5] were able to bypass ten different adversarial detection mechanisms. Most adversarial defenses proposed to date require adversarial retraining of a model to improve its robustness. However, *adversarial retraining assumes the attack algorithm beforehand and causes large computational and time overheads*. Pruthi et al. [12] defend NLP classifiers by correcting misspelled words in the input. However, *such defense methods fail against more natural BERT-based attacks* [13], [14]. We identify a dire need for exploring new mitigation techniques—both the detection and the defense methods—to counter adversarial attacks.

In this work, we explore a novel scheme for detecting adversarial inputs based on their contribution scores. Our methodology is motivated by our previous work in which we analyze the robustness of NLP classifiers, specifically the fake-news detectors, under several configurations and settings [10]. Particularly, in our previous work [10], we note that the contributions of words for adversarial inputs have a different distribution than that for the clean inputs. In this work, we study these signatures more comprehensively based on three commonly used contribution metrics—word removal (see Fig. 1), word substitution, and word gradients—and show that adversarial inputs exhibit significantly higher contribution

^{*}IHSAN Lab, Information Technology University, Lahore, Pakistan.

[†]Department of Electrical Engineering, College of Engineering, University of Ha’il, Saudi Arabia. [§]Department of Information and Computer Science, College of Computer Science and Engineering, University of Ha’il, Saudi Arabia. [¶]Department of Computer Engineering, College of Computer Science and Engineering, University of Ha’il, Saudi Arabia.

Emails: {hassan.ali, msds19011}@itu.edu.pk, {a.alghadhbhan, ms.alazmi, aa.alzamil, alutaibi}@uoh.edu.sa, junaid.qadir@itu.edu.pk,

[‡]Hassan Ali is the corresponding author.

Fig. 1: Our methodology for computing the cumulative contribution score, $C_{\mathcal{F}}(X)$, for an input, X , using the word-removal method. For each word in an input, we first measure its contribution by measuring the decrease in a classifier’s output before and after removing that word from the input. Absolute contributions are then added to get the cumulative contribution score, $C_{\mathcal{F}}(X)$ for X .

Fig. 2: Comparing histograms of $C_{\mathcal{F}}(X)$ -removal scores for clean and adversarial inputs. (Settings: Dataset: Kaggle dataset; Threat model: black-box threat model; Attacks: Text-bugger, Text-fooler, PWWS). Adversarial inputs show considerably higher $C_{\mathcal{F}}(X)$ removal scores as compared to the clean inputs for different classifier architectures.

scores as compared to the clean inputs (See Fig. 2). We identify that this behavior is a natural consequence of an attacker’s stealthiness (which works by changing the decision of a model by perturbing a small fraction of input) and show that this property of adversarial examples, usually considered a strength, is also an inherent weakness, particularly for NLP classification tasks. Building upon the above analysis, we propose, *Con-Detect*, an adversarial detection mechanism that computes the contribution score, $C_{\mathcal{F}}$, of an input and detects adversarial inputs at runtime based on a pre-defined alarm threshold. Unlike adversarial training, *Con-Detect* is efficient and can be integrated with any NLP classifier without a need for retraining it, which makes *Con-Detect* an optimal choice in practical scenarios where retraining large models is generally infeasible because of high computational overheads. Through extensive empirical analysis, we show the efficacy of *Con-Detect* against the strongest state-of-the-art attacks—Text-bugger, Text-fooler, PWWS—for models of varying architectures—MLP, CNN, LSTM, Hybrid-CNN-RNN, BERT—trained for a variety of binary- and multi-classification scenarios.

Our contributions are summarized below.

- To the best of our knowledge, we are the first to identify that the adversarial inputs have higher contribution scores as compared to the clean inputs.
- We are the first to associate this behavior to the attack’s *stealth*, which refers to the attack algorithm’s attempt at

significantly impacting a classifier’s output while minimally perturbing the input in a bid to evade detection. Since this stealthy behavior is a common phenomenon inherent in several attack algorithms, our findings can be generalized to a number of attack methods.

- We propose the *Con-Detect* (Contribution based Detection) method for detecting adversarial inputs at runtime. *Con-Detect* can be incorporated with any classifier without having to retrain it.

II. BACKGROUND

A. Threat Models

Current literature on text-based adversarial attacks/defenses assume two different threat models [11] detailed as follows:

- 1) **White-Box:** White-box threat model assumes a superior attacker with complete knowledge of the model architecture and its parameters.
- 2) **Black-Box:** Black-box threat model assumes a weak attacker with no knowledge of the model architecture or its parameters.

Each of these threat models has unique advantages over the other. The white-box threat model favors an attacker over a defender and thus is generally preferred for evaluating adversarial defenses [15], [16]. However, white-box attacks fail against defenses that hide/suppress the gradients of a model. This problem is well known as the *gradient-obfuscation* and

is exhibited by many adversarial defenses and detectors [17]. For such cases, Athalye et al. [17] recommend using black-box attacks to evaluate the robustness of a model. Due to the use of embedding layers, deep NLP classifiers naturally exhibit the gradient-obfuscation. An embedding layer simply maps an input token (word) to a high-dimensional numeric space using a dictionary and thus has no gradients. This is also one of the reasons for the widespread use of black-box threat model for evaluating the robustness of NLP models [10], [11], [18]–[20].

B. Adversarial Attacks on NLP Classifiers

We specifically focus on four recent attacks motivated by their use in similar works [20], recommendations [10], strength, efficiency, relevance, and recency [11]. Consider an input, $X = \{x_1, x_2, \dots, x_i, \dots, x_n\}$ containing n words, and a classifier, $\mathcal{F}$. An attacker aims to perturb X to get, $X_p = (X + \Delta X)$ such that, $\arg\max \mathcal{F}(X) \neq \arg\max \mathcal{F}(X_p)$, where ΔX is the perturbation. To avoid detection, the attacker minimizes ΔX —perturbing just a few words—and retain the grammatical and semantic structure of the sentence by identifying and perturbing topmost contributing words in X until X_p is misclassified.

Text-bugger [18] uses the Jacobian matrix to identify important words in an input sequence and perturbs them using random character substitutions and synonym replacements. Text-fooler [21] and BAE [14] attack identify important words by measuring the change in the output when a word is removed from the input (*word-removal* method). Perturbation is achieved by synonym replacement using a pre-defined embedding by Text-fooler and by the BERT-Masked Language Model by the BAE attack. PWWS [22] measures the change in the output when a word is replaced by an “unknown” word (*word-substitution* method), and perturbs them using synonym replacement using “Word2Net”.

C. Adversarial Defenses Against Adversarial NLP Attacks

Limited work has been done on mitigating adversarial attacks. Most defense techniques rely on adversarial training [18], [21]–[27] which presumes the attack methods and requires retraining the classifier, which limits its practicality. Other techniques which analyze lexical features such as spelling and grammar are rendered ineffective against more natural BERT-attacks [14], [21].

Recently, Wang et al. [28] propose *TextFirewall*, which uses impact scores of individual words to detect adversarial inputs. We notice two similarities with their work; the basic concept of impact score used by *TextFirewall* is slightly similar to our $C_{\mathcal{F}}$ -scores. *TextFirewall* detects adversarial inputs at runtime without retraining the classifier. However, we believe our work to be significantly different in the following ways.

- Unlike *Con-Detect*, *Textfirewall* is only effective for sentiment classification where a word may either be positive or negative [28].
- *TextFirewall* is only effective for the black-box scenario where an attacker does not know about the detection mechanism (This setting is equivalent to our *Threat Model I*). Many works have argued against the sufficiency

Fig. 3: *Con-Detect* methodology; given an input sequence, X , we compute $C_{\mathcal{F}}(X)$ by adding individual word contributions, $C_{\mathcal{F}}(x_i)$ for all $x_i \in X$. If $C_{\mathcal{F}}(X)$ is greater than the alarm threshold, t , the output of sigmoid, $P_A(X) > 0.5$, indicating that X is adversarial.

and practicality of such a threat model when evaluating adversarial detectors and defenses [5], [15], [17].

- *TextFirewall* is only effective against weak attackers who can only perturb up to 2.1% of words in an input sequence (which is equal to 5 words).
- *TextFirewall* ignores neutral words, such as “but”, “because”, and “when”. Although ignorable in sentiment classification, such words play an important role in other tasks like fake-news detection and topic classification.
- The impact score of Wang et al. [28] measures only the local impacts of words without considering a complete input sequence.
- *TextFirewall* has only been tested for the CNN and LSTM architectures [28].

III. CON-DETECT: CONTRIBUTION BASED DETECTION MECHANISM FOR ADVERSARIAL ATTACKS

In this section, we first introduce *Con-Detect*, a novel contribution scores-based detection scheme for detecting adversarial attacks at runtime and later specifically highlight our method for computing cumulative contribution scores. Finally, we illustrate the working of our *Con-Detect* methodology based on calculating cumulative contribution scores by showing it in action in a case study using a Hybrid-CNN-RNN fake-news classifier [29].

A. Con-Detect Methodology

We hypothesize that the adversarial inputs have higher cumulative contribution scores as compared to the clean inputs. We later formalize and validate this hypothesis in Sections III-B and III-C. Here, we propose *Con-Detect*, a novel detection method that can capture adversarial samples at runtime based on their contribution scores.

- 1) Given an input sequence of n words, $X = \{x_1, \dots, x_i, \dots, x_n\}$ classified into class C by a classifier $\mathcal{F}$, let $C_{\mathcal{F}}(x_i|C)$ be the contribution of the word x_i on the output of $\mathcal{F}$. The cumulative contribution of X , called $C_{\mathcal{F}}(X)$, is defined as,

$$C_{\mathcal{F}}(X) = \sum_{x_i \in X} |C_{\mathcal{F}}(x_i|C)| \quad (1)$$

- 2) $C_{\mathcal{F}}(X)$ computed above is then passed through a filter with a pre-defined alarm threshold, t . The filter will mark the input as adversarial if the score is higher than the alarm threshold and vice-versa. Following some prior works [16], [30], we use a sigmoid function, σ , to implement the filter.

$$P_A(X) = \sigma(C_{\mathcal{F}}(X) - t) \quad (2)$$

Fig. 4: Our methodology for computing the contribution scores, $C_{\mathcal{F}}(x_i)$, for a word, x_i , in the input sequence, X , where $\mathcal{F}$ is the classifier.

where $\sigma(x) = \frac{1}{1+e^{-x}}$ while the alarm threshold, t , is a hyper-parameter that needs to be tuned.

We implement *Con-Detect* as a custom layer to be incorporated with the base classifier, as shown in Fig. 3. Unlike other state-of-the-art methods, *Con-Detect* does not require retraining the classifier and does not presume the algorithm of an attack during training. Additionally, *Con-Detect* is scalable in that it can be incorporated with different classifiers of varying architectures. These factors make *Con-Detect* an attractive choice for practical scenarios.

B. Method of Cumulative Contribution Score

Adversarial attacks aim to change a model’s decision while minimally perturbing an input which compels an attacker to choose highly contributing perturbations. We note that this property can be a major weakness of adversarial attacks, specifically for NLP classification. To explain this, consider an input sequence, X , originally classified by $\mathcal{F}$ into the class, C_{org} . An attacker perturbs X into X_p such that X_p gets misclassified into the class, C_{adv} . We divide X into two distinct sets, X_A and X_B such that $(X_A \cup X_B = X) \wedge (X_A \cap X_B = \emptyset)$, where X_A contains words that remain unperturbed while X_B comprises words that the attacker replaces with X_{Δ} while attacking. We make the following key observations:

- 1) In our previous work [10], we note that most words in a clean input positively contribute to a classifier’s output. Additionally, X_B has a higher expected contribution to the output than X_A , which is why an attacker chooses to perturb X_B . Thus, for a clean input,

$$\sum_{\forall x \in X_B} C_{\mathcal{F}}(x|C_{org}) \geq \sum_{\forall x \in X_A} C_{\mathcal{F}}(x|C_{org}) \geq 0 \quad (3)$$

where $C_{\mathcal{F}}(x|C_{org})$ denotes the contribution of the word x to the output class, C_{org} .

- 2) As X_A suggests $\mathcal{F}$ to output the class C_{org} (Eq. 3), X_A expectedly has negative contributions to the class C_{adv} for an adversarial input.

$$\sum_{\forall x \in X_A} C_{\mathcal{F}}(x|C_{adv}) \leq 0 \quad (4)$$

- 3) Finally, for a successful attack, X_{Δ} , expectedly has a higher contribution than X_B to overcome the negative effects of X_A (4). Mathematically,

$$\sum_{\forall x \in X_{\Delta}} C_{\mathcal{F}}(x|C_{adv}) \geq \sum_{\forall x \in X_B} C_{\mathcal{F}}(x|C_{org}) \geq 0 \quad (5)$$

Stated simply, *the expected sum of absolute contributions of words in an input is larger for an adversarial input than*

Algorithm 1 Computing contribution scores for an input, X

Input:

$X \leftarrow \{x_1, x_2, \dots, x_{i-1}, x_i, x_{i+1}, \dots, x_n\}$
 $\mathcal{F} \leftarrow$ NLP Classifier
 $ScoreFn \leftarrow$ Score function to be used

Output:

$C_{\mathcal{F}} \leftarrow$ Cumulative contribution score

```

1: function WORD SUBSTITUTION SCORE( $X, \mathcal{F}, i$ )
2:    $X_{i \rightarrow u} \leftarrow \{x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n\}$ 
3:   return  $\mathcal{F}(X) - \mathcal{F}(X_{i \rightarrow u})$ 
4: end function

5: function WORD REMOVAL SCORE( $X, \mathcal{F}, i$ )
6:    $X_{-i} \leftarrow \{x_1, \dots, x_{i-1}, < unk >, x_{i+1}, \dots, x_n\}$ 
7:   return  $\mathcal{F}(X) - \mathcal{F}(X_{-i})$ 
8: end function

9: function WORD GRADIENT SCORE( $X, i$ )
10:  return  $\frac{\partial \mathcal{F}(X)}{\partial x_i}$ 
11: end function

```

```

12: define  $i \leftarrow 0, C_{\mathcal{F}} \leftarrow 0$ 
13: repeat
14:    $C_{\mathcal{F}} \leftarrow C_{\mathcal{F}} + |ScoreFn(X, \mathcal{F}, i)|$ 
15:    $i \leftarrow i + 1$ 
16: until  $i \leq n$ 
17: return  $C_{\mathcal{F}}$ 

```

that for a clean input, given that the attack is stealthy. This signature is a natural consequence of an attacker’s inconspicuousness. However, as mentioned previously, estimating these contributions is challenging due to the gradient obfuscation in NLP models [17]. Fortunately, we note that word substitution, word removal, and saliency scores are commonly used by the attackers and several explainability methods [31], [32] to estimate a word’s contribution to a classifier’s output. Detailed methodologies are given below.

1) *Word Substitution*: For each word, x_i , in an input, X , we replace the word with an “unknown” token and the absolute change in the probability of the original class indicates the contribution, $C_{\mathcal{F}}(x_i|C)$. Individual word contributions are then added, thereby getting the cumulative contribution score, called $C_{\mathcal{F}}$ -substitution score, for X . A detailed methodology is given in Algorithm 1 and illustrated in Fig. 4(a).

2) *Word Removal*: For each x_i in X , we remove x_i from X , and the absolute change in the probability of the original class measures the contribution, $C_{\mathcal{F}}(x_i|C)$. As before, we add individual word contributions, thereby getting the cumulative contribution score, called $C_{\mathcal{F}}$ -removal score. Details are given in Algorithm 1 and illustrated in Fig. 4(b).

3) *Word Gradients*: For each x_i in X , we compute the gradient of the output of a model with respect to the embedding layer and use it as the contribution score, $C_{\mathcal{F}}(x_i|C)$. Individual

word contributions of different words are separately added to get $C_{\mathcal{F}}$ -gradient score for X . See Algorithm 1 for details.

C. An Illustrative Case Study of Con-Detect's Cumulative Contribution Score Methodology

In the previous section, we hypothesize that the adversarial inputs have higher $C_{\mathcal{F}}$ scores than the clean inputs. To validate this hypothesis, we attack the state-of-the-art Hybrid-CNN-RNN [29] fake-news classifier trained on Kaggle fake-news dataset. Our attacks methods include three state-of-the-art black-box attacks [10], [20]—Text-fooler [21], Text-bugger [18], and PWWS [22] attacks—from the *Text-Attack* library. Our experimental setup for this analysis is shown in Fig. 7.

Fig. 7: Illustration of computation of $C_{\mathcal{F}}(X)$ scores for clean and adversarial inputs against the Hybrid-CNN-RNN fake-news classifier. Blue arrows represent the clean flow and the red arrows represent the adversarial flow.

Word Substitution. Fig. 5 shows the histograms of $C_{\mathcal{F}}$ -substitution scores for clean and adversarial inputs generated by the three attacks. Adversarial inputs are clearly distinguishable from the clean inputs due to higher $C_{\mathcal{F}}$ -substitution scores, which is consistent with our intuitions.

Fig. 5: Comparing distributions of $C_{\mathcal{F}}$ -substitution scores clean (blue) and adversarial inputs. (Settings: Contribution metric is $C_{\mathcal{F}}$ -substitution score, Model is Hybrid-CNN-RNN classifier, Threat model is the Black-box). Adversarial inputs have higher $C_{\mathcal{F}}$ -substitution scores as compared to the clean inputs.

Fig. 6: Comparing distributions of $C_{\mathcal{F}}$ -removal scores clean (blue) and adversarial inputs. (Settings: Contribution metric is $C_{\mathcal{F}}$ -removal score, Model is Hybrid-CNN-RNN classifier, Threat Model is the Black-box). Adversarial inputs have higher $C_{\mathcal{F}}$ -removal scores as compared to the clean inputs.

Being stealthy—perturbing a fraction of input—is considered a strength of adversarial attacks [11]. However, this forces an attacker to choose highly impacting perturbations that raises $C_{\mathcal{F}}$ scores. Interestingly, $C_{\mathcal{F}}$ scores are similar for different attacks despite major differences in the algorithms, suggesting that *Con-Detect* is attack-agnostic. Our experiments establish that *the iconic stealthiness of the adversarial attacks imprints detectable signatures on the inputs of NLP classifiers*.

Word Removal. Fig. 6 shows histograms of $C_{\mathcal{F}}$ -removal scores for the clean and adversarial inputs. For the same reasons discussed previously, adversarial inputs show higher $C_{\mathcal{F}}$ scores, making them distinguishable from the clean inputs. We note that $C_{\mathcal{F}}$ -removal score cannot effectively detect adversarial inputs for the IMDB dataset. This is illustrated by the closeness of clean and adversarial scores in Fig. 6(b).

Word Gradients. A similar analysis is performed as with the previous two methods. However, we note that $C_{\mathcal{F}}$ -gradient score is not effective for detecting adversarial inputs. Detailed investigations are left for future work. In the subsequent experiments, we use the word-substitution and the word-removal methods for evaluating *Con-Detect*.

IV. EXPERIMENTAL SETUP

Our experimental setup is given in Fig. 8. A clean input is perturbed using the *Text-Attack* library and provided to *Con-Detect*-classifier which gives two outputs, $\mathcal{F}(X)$ and $P_A(X)$.

A. Threat Models

Due to the gradient obfuscation problem of NLP classifiers discussed in Section II-A, our experiments assume the black-box threat model. We further consider three different scenarios for the black-box threat model as shown in Fig. 9.

Fig. 8: Experimental setup for our evaluation of *Con-Detect*. The blue lines represent the normal flow, while the red lines represent the adversarial flow.

Fig. 9: The three considered threat models: *Con-Detect*-blind setting assumes that the attacker is unaware of *Con-Detect*; *Con-Detect*-aware threat model depicts a *Con-Detect*-aware scenario where an attacker knows that *Con-Detect* is deployed; and *Con-Detect*-adaptive setting where an attacker adaptively modifies the attack algorithm to fool *Con-Detect*.

- ***Con-Detect*-blind threat model** assumes that the attacker is unaware of *Con-Detect*. The attacker only tries to fool the classifier in this case.
- ***Con-Detect*-aware threat model** assumes that the attacker knows about *Con-Detect* and aims to fool both the classifier and the detector simultaneously.
- ***Con-Detect*-adaptive threat model** assumes that the attacker is aware of *Con-Detect* and intelligently modifies the attack algorithm to fool *Con-Detect*.

Methodology of *Con-Detect*-adaptive attack. Following the convention of adversarial researchers, we assume an adaptive attacker who modifies his attack algorithm to specifically fool *Con-Detect*. The proposed modification can be easily extended to all of the black-box attacks used in this paper. Noting that *Con-Detect* makes it harder for a conventional attacker to estimate the importance of a word due to the mismatch between the importance scores calculated for *Con-Detect* and the classifier, our adaptive attacker unifies the outputs of the classifier and *Con-Detect* in a wrapper function such that changing the wrapped output ensures that both the classifier and the detector are fooled.

To explain the wrapper function, we assume a binary classification task, e.g., fake-news detection where an input sequence may either be real or fake. Consider an input, X , originally marked as *unperturbed* and *fake* by *Con-Detect* and the classifier, respectively. The wrapped output is then defined as,

$$P_{wrap}(X) = (1 - P_A(X)) \begin{bmatrix} P(\text{real}|X) \\ P(\text{fake}|X) \end{bmatrix} + P_A(X) \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad (6)$$

where $P_A(X)$ denote the probability of X being adversarial, and $P(\text{real}|X)$ and $P(\text{fake}|X)$ denote the probability of X being real and fake, respectively. To change the output decision of *Con-Detect*-classifier from *fake* to *real*, the following

Fig. 10: Illustrating the condition defined in Eq. 7 by an adaptive attacker for the fake-news classification task. The area highlighted in red shows that the condition is false, while the blue area represents the condition is true. (Settings: Input, X , is originally fake and unperturbed.). When Eq. 7 is satisfied, $P_A(X) < 0.5$ (undetected) and $P(\text{real}|X) > 0.5$ (successful adversarial attack).

condition must be met,

$$(1 - P_A(X)) \cdot P(\text{real}|X) > 0.5 \quad (7)$$

$$\Rightarrow P_A(X) < 1 - \frac{0.5}{P(\text{real}|X)}$$

Fig. 10 illustrates Eq. 7. Apparently, meeting the condition in Eq. 7 makes $P(\text{real}|X) > 0.5$ and $P_A(X) < 0.5$ indicating that both the classifier and the detector have been fooled. Although Eq. 6 assumes binary classification, we note that our approach can simply be extended to a multi-classification scenario using the following equation

$$P_{wrap}(X) = (1 - P_A(X)) \cdot \mathcal{F}(X) + P_A(X) \cdot O_H(\arg\max \mathcal{F}(X), M) \quad (8)$$

where O_H is the one-hot encoded representation of $\arg\max \mathcal{F}(X)$ for M number of classes.

B. Datasets

We evaluate our methodology on three different datasets, the Kaggle fake-news dataset, the IMDB dataset, and the AG news dataset. All the three datasets are open-source, comprehensive and a common choice in similar recent works [10], [20], [29]. **Kaggle fake-news dataset.** The dataset, accessible at the Kaggle website¹, is a binary classification dataset having 20800 samples in the training set and 5200 samples in the test set, making a total of 26000 samples. Each sample has different fields—*id*, *title*, *author*, *text* and *label*. The label may either be 1 or 0 denoting the fake or real content, respectively. We use the *text* field—a string of multiple sentences of an article—as the input and the one-hot encoded *label* as the output of our classifier.

IMDB dataset. The IMDB dataset is a binary classification dataset comprising 50000 movie reviews—25000 training and 25000 test samples. Each sample has two fields; the *content* field is the input and one-hot encoded *label* is the output of our classifier.

AG News topic classification dataset. A subset of the AG news dataset downloaded from the Kaggle website² having four classes, each class comprising 30000 training and 1900 test samples. Each sample has three fields—*title*, *description*,

¹<https://www.kaggle.com/c/fake-news/data>

²<https://www.kaggle.com/amananandrai/ag-news-classification-dataset>

Fig. 11: Comparing the performance of our classifiers before and after attacking for all datasets. (Settings: Classifiers are MLP, CNN, LSTM, Hybrid, BERT; Attacks are Text-bugger, Text-fooler, and PWWS; Threat model is black-box). Accuracy of classifiers is fairly good for the clean inputs. However, for the adversarial inputs the accuracy severely degrades (dropping to zero in many cases).

and *class*. Following previous conventions [20], we concatenate the *title* and the *description* fields to use as input to our classifier. The one-hot encoded *class* field is used as the output.

C. Model Architectures and Frameworks

We use different model architectures for classification in our experiments. Specifically, in addition to the conventionally used Long-Short Term Memory-based (LSTM) architecture, we experiment with the simple multi-layer perceptrons (MLP), convolutional neural networks (CNNs), and more recent Hybrid-CNN-RNN [29] and BERT architectures. CNNs have already shown their worth as NLP classifiers [33], [34]. Additionally, we note a general trend towards using hybrid CNN-RNN techniques for NLP classification [29], [35], [36]. We choose the architecture proposed by Nasir et al. [29] due to its recency and generalizability [10], [29]. We fine-tune BERT³ for our tasks using a pre-trained tokenizer⁴. For all other cases, we train classifiers from scratch using a Keras tokenizer and initialize the embedding layer with the GloVe word embedding [29].

While evaluating our approach against *Con-Detect*-adaptive attacks, we only use Hybrid-CNN-RNN classifiers trained on different datasets. Hybrid-CNN-RNN is recent [29], computationally more efficient, and is significantly faster to attack as compared to BERT-based classifiers. This considerably reduces the time required for generating the results.

Our implementation of models and detection methodology uses the state-of-the-art open-source libraries, Tensorflow and Keras. For attacking, we use the attack implementations provided in the state-of-the-art “Text-Attack” library [11]. Particularly, we use three different attacks—Text-bugger [18], Text-fooler [21], and PWWS [22]—in our evaluations, following recent practices [10], [20]. In our previous work, we noted that although BERT-based adversarial attacks can generate more natural adversarial perturbations, it achieves low attack success rates as compared to other attack methods, i.e., Text-fooler, Text-bugger, and PWWS. We also show later in this paper that *Con-Detect* is also effective against BERT adversarial attacks.

³Smaller BERT model: https://tfhub.dev/tensorflow/small_bert/bert_en_uncased_L-4_H-512_A-8/1

⁴Text preprocessing for BERT: https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3

V. RESULTS

In this result, we present a thorough performance evaluation by first describing the accuracy of the classifiers (§V-A) and then consider the effect of the alarm threshold on the performance of *Con-Detect* (§V-B). We consider different threat models and provide results for the *Con-Detect-blind* threat model (§V-C); the *Con-Detect-aware* threat model (§V-D); and the *Con-Detect-adaptive* threat model (§V-E). Finally, an analysis of the computational overhead of *Con-Detect* is provided in Section V-F.

To evaluate our methods, we use different metrics used in many recent works for the robustness evaluation either individually [20] or in combination [1], [10].

- *Attack Success Rate (ASR)*. ASR is the ratio of adversarial inputs incorrectly classified by a classifier to the total number of adversarial inputs. A lower ASR suggests a more robust classifier. (Note that ASR is the inverse of accuracy— $ASR \% = 100 - Accuracy \%$).
- *Number of queries*. The number of times an attacker asks a classifier to make predictions. The higher the required number of queries, the more robust the classifier.
- *Percent words perturbed*. Percentage of words that an attacker perturbs from an input sequence. For a model to be robust, the percentage should be high.

A. Accuracy of the classifiers

All our classifiers give an accuracy of above 90% for the Kaggle and AG news datasets, and around 80% for the IMDB dataset. Fig. 11 reports the comparison of *vanilla*-classifiers’ accuracy before and after the attack assuming the *black-box* threat model. We observe that for most cases, the accuracy drops to under 20% when a classifier is under attack, suggesting that most classifiers are fragile. However, we find that CNNs are relatively more robust, as already noted in a previous work [10].

B. Effect of the Alarm Threshold on *Con-Detect*

We analyze the effect of changing t on the efficacy of *Con-Detect* measured using the F1-score. The base model is the Hybrid-CNN-RNN fake-news classifier trained on the Kaggle fake-news dataset and the threat model assumed is *Con-Detect-blind* threat model. Results of our experiments are provided in Fig. 12 and Fig. 13. Each of these figures reports two different results.

Fig. 12: Analyzing the effect of the alarm threshold, t , on the F1-score of *Con-Detect* for different datasets. (Settings: Score: $C_{\mathcal{F}}$ -substitution score; Classifier: Hybrid-CNN-RNN; Threat model: black-box). $C_{\mathcal{F}}$ -substitution score effectively detects adversarial examples. IMDB dataset is more vulnerable to attacks.

Fig. 13: Analyzing the effect of changing the alarm threshold, t , on the F1-score of *Con-Detect* for different datasets. (Settings: Score: $C_{\mathcal{F}}$ -removal score; Classifier: Hybrid-CNN-RNN; Threat model: black-box). $C_{\mathcal{F}}$ -removal score is effective for the Kaggle fake-news and AG news datasets. For the IMDB dataset, $C_{\mathcal{F}}$ -removal score is only partially effective.

- 1) The black line records the F1-scores of *Con-Detect* as the alarm threshold, t , increases from zero to the maximum.
- 2) The green dashed line drawn vertically marks the best scenario determined by the largest F1-score. The dashed line is also accompanied by a set of values—the best F1-score, the optimal alarm threshold, false-negative⁵ and false-positive⁶ rates—detailing the best scenario. For example, in Fig. 12(a), the best results are achieved when $t=0.378$. At this value of t , the F1-score is 94.8%, and the false-positive rate is 7.5%.

For each of the two estimation methods, we observe that the F1-score first increases rapidly reaching a maximum value, and then decreases as the alarm threshold increases. For a too small t , a high false-positive rate causes a low F1-score—i.e., most clean samples are incorrectly marked as adversarially perturbed by our detector. When t is too large, the detector cannot detect adversarial inputs, which increases the false-negative rate, thus F1-score decreases. Interestingly, the rapid

increase in the F1-score for small values of t is because $C_{\mathcal{F}}$ -scores of clean inputs are concentrated near zero (Fig. 5 and 6).

Now coming to the question of *how to automatically choose the alarm threshold*? In Fig. 12 and 13, we note that the false-positive rate can serve as a good indicator for the optimal t . Particularly, the figures show that for most cases, the best value of t lies between the false-positive rates of 7% and 20%. Although the precise value of allowable false-positives in practical applications would vary from case to case, as a general rule, we propose to use the false-positive rate of 15% as an indicator of the best t .

C. Evaluation under *Con-Detect*-blind Threat Model

We first evaluate the efficacy of *Con-Detect* under the *Threat model 1* (Con-Detect-blind Attacker). Fig. 14(a), (b) and (c) reports combined ASRs achieved by the three attacks—Text-bugger, Text-fooler, and PWWS—against different classifiers for the Kaggle, AG News and IMDB datasets respectively. We define the combined ASR as an average of individual ASRs of all three attacks.

Fig. 14: Comparing the Attack Success Rates (ASRs) of three state-of-the-art attacks against the vanilla classifiers and *Con-Detect*-classifiers for a range of deep learning architectures and datasets. (Settings: Architectures are MLP, CNN, LSTM, Hybrid-CNN-RNN, and BERT; Datasets are Kaggle, AG News, and IMDB; Threat model is black-box; Attacker is unaware of *Con-Detect*). *Con-Detect* can effectively detect blind adversarial attacks, which causes low ASRs.

⁵The ratio of adversarial inputs marked as unperturbed by *Con-Detect*.

⁶The ratio of clean inputs that are mistakenly marked as adversarially perturbed by *Con-Detect*.

Fig. 15: Comparison of three state-of-the-art attacks against the vanilla classifiers and *Con-Detect*-classifiers for a range of deep learning architectures and datasets. (Settings: Architectures are MLP, CNN, LSTM, Hybrid-CNN-RNN, and BERT; Datasets are Kaggle, AG News, and IMDB; Threat model is black-box; Attacker is aware of *Con-Detect*.) *Con-Detect* can effectively detect adversarial attacks even in the *Con-Detect*-aware scenario.

Fig. 16: C_F -substitution scores for LSTM classifier trained Kaggle fake-news dataset. C_F -scores for the clean and the adversarial inputs highly overlap which increases the false-negative rate in Fig. 14(a).

We note that, for most cases, *Con-Detect* reduces the ASRs of all three attacks from 100% to 0%. For the **Kaggle dataset**, the word-removal method more effectively detects adversarial inputs as compared to the word-substitution method, specifically for LSTM-classifier. For example, the combined ASR reduces from 100% to $\approx 60\%$ for the word-substitution method and $\approx 0\%$ for the word-removal method. We attribute this to a large overlap of C_F -substitution scores of clean and adversarial inputs as shown in Fig. 16 for the LSTM-classifier. Fig. 14 shows that *Con-Detect* is not effective for BERT-classifiers. The reason is that BERT is fairly hard to attack even without a defense deployed. To fool BERT, an attacker has to perturb significantly more words as compared to other classifiers which reduces the negative contributions of X_A in Eq. 4 and degrades the stealth of attack rendering the attack impractical (§VI-B).

Con-Detect is equally effective against all three adversarial attacks for the **AG news dataset**, irrespective of the contribution method used for detection. This shows that *Con-Detect* not only works for binary classifiers but is also effective for multi-class classifiers. Although *Con-Detect* significantly

reduces the efficacy of attacks, we note that the decrease is not as promising for the **IMDB dataset** as for the Kaggle and AG news datasets, notably for the word-removal method (Fig. 14(c)). We attribute this to the closeness of C_F -removal scores of clean and adversarial inputs for the IMDB dataset, as illustrated in Fig. 6 under Section III-C. For the IMDB dataset, we note that word substitution-based detection gives better results as compared to the word removal-based approach.

D. Evaluation under *Con-Detect*-aware Threat Model

We evaluate the efficacy of *Con-Detect* under the *Threat model 2* (*Con-Detect*-aware Attacker). This setting assumes an attacker who is aware of the detection mechanism and perturbs X to change $\arg\max \mathcal{F}(X)$ while $P_A(X) < 0.5$. The classifiers used in this evaluation are based on MLP, CNN, LSTM, Hybrid-RNN-CNN, and BERT architectures, and trained for different datasets—IMDB reviews, Kaggle fake-news, AG news. Results are shown in Fig. 15.

ASR decreases. Fig. 15(a) provides combined ASRs of attacks against classifiers for different datasets. As previously, the combined ASR is an average of individual ASRs of all three attacks—Text-bugger, Text-fooler, and PWWS—used in the analysis. As expected, *Con-Detect*-aware attackers achieve higher ASRs than the *Con-Detect*-blind-attackers. However, *Con-Detect* significantly degrades the attack performance as compared to vanilla (no detection) classifiers even for the *Con-Detect*-aware attackers as shown in the figure. Unlike previously, we note that *Con-Detect* works best with BERT-classifiers by reducing ASRs to 0% for all attacks and datasets. This is because *Con-Detect* makes it harder for an attacker to attack an already hard-to-attack BERT-classifier. Other reasons can be the gradient-obfuscation caused by a deeper

Fig. 17: Comparison of BERT Adversarial Example (BAE) attack against the vanilla classifiers and *Con-Detect*-classifiers for different datasets. (Settings: Architecture is Hybrid-CNN-RNN; Datasets are Kaggle, AG News, and IMDB; Threat model is black-box; Attacker is aware of *Con-Detect*). *Con-Detect* can effectively detect BAE attacks even in the *Con-Detect*-aware scenarios.

BERT architecture (§ VI-C) and a large mismatch between the contribution scores for the classifier and *Con-Detect*. We observe that the word-removal method works better than the word-substitution method for the Kaggle dataset, while for the AG News and the IMDB datasets, word-substitution gives better performance.

The average number of queries increases. Fig. 15(b) reports the combined average of the number of queries made by the three attacks. We note that successfully attacking classifiers trained on the AG News dataset requires considerably fewer number of queries due to the multiple classes making it easier for an attacker to find an optimal point in the perturbation space. Additionally, the average query count increases considerably when attacking *Con-Detect*-classifier as compared to the vanilla classifiers suggesting that *Con-Detect*-classifiers are harder to attack. For example, considering the Kaggle dataset, an attacker will expectedly query a vanilla MLP classifier 645 times to achieve an ASR of 100%, irrespective of the attack methodology. This is in contrast to *Con-Detect*-MLP classifier, where the attacker is forced to make 1870 ($\approx 3 \times 645$) queries

and still fails to launch a successful attack. In practice, where each query typically incurs a cost, *Con-Detect* significantly increases the cost of attack.

The average percentage of perturbed words increases. Fig. 15(c) reports the average percentage of words perturbed by the three attacks. The percentage is considerably higher for *Con-Detect*-classifiers as compared to the vanilla classifiers. For example, the percentage increases from 16% for a vanilla Hybrid-CNN-RNN to 30% for *Con-Detect*-Hybrid-CNN-RNN classifier for the Kaggle dataset. Such a large increase in the perturbed words suggests that *Con-Detect* is fairly robust and significantly degrades the stealth of an attack.

Evaluation against BAE attack: Previously, we have empirically shown the efficacy of *Con-Detect* against three state-of-the-art adversarial attacks. Here we show that *Con-Detect* is also effective against more recent BERT-based attacks. Specifically, we attack vanilla and *Con-Detect*-classifiers using **BERT Adversarial Example (BAE)** attack while assuming the *Con-Detect*-aware threat model (See Fig. 9). We use Hybrid-CNN-RNN based classifiers trained on Kaggle, AG News, and IMDB datasets for this analysis. Results in Fig. 18

Fig. 18: Comparison of adaptive and non-adaptive attackers using three state-of-the-art attack methods against the vanilla classifiers and *Con-Detect*-classifiers for Hybrid-CNN-RNN architecture on different datasets. (Settings: Architecture is Hybrid-CNN-RNN; Datasets are Kaggle, AG News, and IMDB; Threat model is black-box; Attacker is aware of *Con-Detect*. Adaptive attacker can adaptively manipulate the attack algorithms.). Adaptive attackers can considerably degrade *Con-Detect* performance; however, the percentage of words perturbed also generally increases.

Fig. 19: Combined averages of time (in seconds) required by *Con-Detect*-classifiers to process N number of inputs as compared to the vanilla classifiers. (Settings: Classifiers used for averaging are MLP, CNN, LSTM and Hybrid-CNN-RNN. Dataset is the Kaggle dataset). *Con-Detect*-classifiers cause large time overheads as compared to vanilla classifiers.

report the combined averages of ASRs, number of queries, and percent words perturbed by adaptive attackers.

As previously, we observe that when attacking *Con-Detect*, the ASR of BAE attack significantly decreases, while the average number of queries made by an attacker and the average percentage of words perturbed are increased. Our experiments establish that *Con-Detect*-classifiers are harder to evade even for BERT-based attacks.

E. Evaluation under *Con-Detect*-adaptive Threat Model

Using the methodology proposed in Section IV-A, we adaptively modify Text-bugger, Text-fooler, and PWWS attacks. Results in Fig. 18(a), (b), and (c) respectively report the combined averages of ASRs, number of queries, and percent words perturbed by *Con-Detect*-adaptive attackers.

We note that the ASR of an adaptive attacker is on average 30% greater than that of a non-adaptive attacker. This superiority of adaptive attacks is due to the clever unification of the classifier and detector outputs proposed in Eq. 8. Fig. 18(a) shows that the proposed adaptive attack generalizes to the multi-classification scenario as illustrated by increased ASRs for the AG News dataset. As previously, the reduced performance on the IMDB dataset is due to a high overlap between the C_F -scores for the clean and the adversarial inputs as noted in Fig. 5(b) and 6(b). However, we note that the ASR of an adaptive attacker against *Con-Detect*-classifiers is still smaller than that against vanilla classifiers irrespective of the dataset used.

In Fig. 18(b), we note a slight reduction in the average queries made by an adaptive attacker suggesting that the cost of attack is slightly reduced. However, Fig. 18(c) shows that this supremacy is achieved at the cost of a larger percentage of words perturbed by the attacker. Specifically, an adaptive attacker has to perturb 4 to 6% more words as compared to a non-adaptive attacker. Our findings show that *Con-Detect* is significantly harder to defeat even for an adaptive attacker.

F. Computational overhead of *Con-Detect*

We compare the expected time required by *Con-Detect*-classifiers to process N number of inputs with that for the vanilla classifiers in Fig. 19. It is seen that the computational overhead of *Con-Detect* increases as the number of input samples increases. This is particularly attributed to *Con-Detect*

computing the C_F -score for each word in an input sequence and performing subsequent operations.

Although such large computational overheads can limit the practicality of *Con-Detect*, we observe that the time required for attacking *Con-Detect*-classifiers is also several folds greater than that for the vanilla classifiers. This is mainly attributed to a larger number of queries required by an attacker when attacking *Con-Detect*-classifiers.

VI. DISCUSSION

A. Cost of attacking *Con-Detect*

In practical scenarios, such as online ML-services, there is a huge cost associated with each query/request that a user makes [1]. One such example is Google APIs which only allow a limited free quota per day per user. Our results show a large increase in the number of queries required by an attacker when attacking *Con-Detect*-classifiers as compared to the vanilla classifiers. We attribute this to *Con-Detect*, making it harder for an attacker to estimate the importance of a word due to the mismatch between the importance scores calculated for *Con-Detect* and the classifier. Our experiments suggest that *Con-Detect*, in addition to reducing the Attack Success Rate, significantly increases the cost and time of launching an attack.

B. Stealth of adversarial attacks on *Con-Detect*

Con-Detect skillfully detects adversarial perturbations that change a classifier’s output by perturbing a fraction of the input words. A *Con-Detect*-aware attacker thus has to choose the low impacting perturbations, consequently, perturbing a larger percentage of words to achieve the desired results, which significantly degrades the stealth of the adversarial attack.

Fig. 20 compares the adversarial examples generated by the Text-fooler attack against *Con-Detect*-Hybrid-CNN-RNN classifier for the ***Con-Detect*-blind**, ***Con-Detect*-aware** and ***Con-Detect*-adaptive** threat models. We note that the adversarial examples generated under *Con-Detect*-blind threat model against *Con-Detect*-classifiers are exactly similar to those generated against the vanilla classifiers; this is because in both cases, an attacker’s goal is to only fool the classifier. It is evident from Fig. 20 that the adversarial examples generated against *Con-Detect*-classifiers are highly perturbed, raising suspicion upon manual inspection.

We note many similarities between the adversarial perturbations made by *adaptive* and *aware* attackers when attacking similar *Con-Detect*-methods (substitution or removal). For example, in Fig. 20, the phrase “France chose an idealistic, traditional candidate” from the clean input was changed to “France opting an fantasy, accustomed candidate” and “France selecting an fantasy, traditional candidate” by the *aware* and *adaptive* attackers respectively, when attacking word-removal based *Con-Detect*-classifier. The same original phrase was changed to “France option an notion, usual candidate” and “France option an notion, traditional candidate” by the *aware* and *adaptive* attackers respectively, when attacking word-substitution based *Con-Detect*-classifier. We attribute this to similar methods causing similar word contribution scores.

No Attack	PARIS — France chose an idealistic, traditional candidate in Sunday’s primary to represent the Socialist and parties in the presidential election this spring. The candidate, Benoît Hamon, 49, who ran on the slogan that he would “make France’s heart beat,” bested Manuel Valls, the former prime minister, whose campaign has promoted more policies and who has a strong background. Mr. Hamon appeared to have won by a wide margin, with incomplete returns showing him with an estimated 58 percent of the vote to Mr. Valls’s 41 percent. “Tonight the left holds its head up high again it is looking to the future,” Mr. Hamon said. (words changed = 0)	$P_A(X) = 0$ Classifier: fake
Con-Detect-blind attack	WAGER — Francia elect an idealistic, traditional candidate in Sunday’s primary to represent the Socialist and parties in the presidential election this spring. The candidate, Benoît Hamon, 49, who ran on the motto that he would “deliver France’s heart beat,” bested Schoolbooks Valls, the former elemental presidency, whose campaigning has promoted more policies and who has a strong source. Sirs. Hamon appeared to have won by a wide margin, with incomplete returns showing him with an estimated 58 percent of the vote to Sirs. Valls’s 41 percent. “Thursday the gauche conserved its jefe up greater again it is searching to the future,” Mr. Hamon said. (words changed = 18)	$P_A(X) = 1$ Classifier: real
Substitution Con-Detect-aware attack	PARIS — French opting an fantasy, accustomed candidate in Monday’s primitive to accounted the Communists and parties in the presidential voters this printemps. The candidate, Benoît Hamon, 49, who tumbled on the motto that he would “deliver France’s heart bludgeoned,” bested Manuela Valls, the former original preside, whose countryside owns favored more policies and who had a strong backdrop. Sirs. Hamon spoken to owns won by a bigger margin, with unfulfilled dividend showings him with an valued 58 percent of the votes to Sirs. Valls’s 41 percent. “Thursday the gauche belonged its jefe up grand again it is searching to the future,” Mr. Hamon said. (words changed = 37)	$P_A(X) = 0.78$ Classifier: real
Removal Con-Detect-aware attack	WAGER — France option an notion, usual candidate in Sunday’s primitive to contained the Democrats and parties in the presidential votes this fountains. The candidate, Benoît Hamon, 49, who ran on the motto that he would “deliver France’s heart overcome,” overmatched Schoolbooks Valls, the former elemental leader, whose campaign acres favoured more policies and who has a serious roots. Sirs. Hamon looked to have won by a overwhelming margin, with incomplete repaid showed him with an calculating 58 percent of the vote to Nobleman. Valls’s 41 percent. “Weekend the going occupies its jefe up greater again it is discovering to the future,” Mr. Hamon said. (words changed = 33)	$P_A(X) = 0.8$ Classifier: real
Substitution Con-Detect-adaptive attack	WAGER — France selecting an fantasy, traditional candidate in Friday’s basic to comprise the Socialists and servings in the presidential votes this xuan. The candidate, Benoît Hamon, 49, who flee on the motto that he would “deliver France’s cardiology overcome,” bested Schoolbooks Valls, the former elemental presidency, whose campaigning hectare advancements more politics and who possesses a influential source. Sirs. Hamon surfaced to has won by a widespread margin, with inaccurate earnings demonstration him with an valued 58 percent of the ballots to Sirs. Valls’s 41 percent. “Thursday the gauche conserved its jefe up greater again it is searching to the future,” Mr. Hamon said. (words changed = 41)	$P_A(X) = 0.49$ Classifier: real
Removal Con-Detect-adaptive attack	WAGER — France option an notion, traditional candidate in Weekend’s primitive to is the Socialism and chapters in the republican voters this fountains. Both candidate, Benoît Hamon, 49, who evaded on the motto that he would “deliver France’s heart defeats,” overpowers Schoolbooks Valls, the former elemental presidency, whose campaigning hectares favour more directives and who have a heavy source. Sirs. Hamon emerged to have won by a whopping margin, with unfinished repaid demo him with an seen 58 p of the votes to Sirs. Valls’s 41 percent. “Thursday the gauche conserved its jefe up greater again it is discovering to the future,” Mr. Hamon said. (words changed = 43)	$P_A(X) = 0.49$ Classifier: real

Fig. 20: Comparing the adversarial examples of Text-fooler attack for *Con-Detect-blind*, *Con-Detect-aware* and *Con-Detect-adaptive* threat models. Words changed by the attacker are highlighted in red. Total number of perturbed words is reported in blue. At the right, we report the *Con-Detect* output, $P_A(X)$ and the classifier’s decision. (Settings: Dataset is the Kaggle dataset, Classifier is Hybrid-CNN-RNN). An attacker has to perturb significantly more words, thus compromising the stealth of the attack, to fool *Con-Detect*-classifiers.

Fig. 21: Impact of changing z on the output of a steep sigmoid function, σ_z . When z is large, changing the input does not change the output, except at the transition points. Thus, σ_z approximates a discrete behavior for large z .

C. Limitations of current attacks

As noted in Sections I and II, most state-of-the-art attacks estimate the contribution of a word by perturbing an input (e.g., using word removal or word substitution) and measuring the change in the output. We note that this approach is similar to the zeroth-order optimization commonly used for attacking visual classifiers under the black-box threat model [37].

However, such attacks fail against defenses that deploy very deep architectures (as noted with BERT in Section V-D) or obfuscate/hide the gradients [1]. To illustrate this for NLP tasks, we suppress the gradients of the MLP fake-news classifier by processing its output probabilities using a steep sigmoid

function, σ_z , as defined below

$$\mathcal{F}_{obf}(X) = \sigma_z(\mathcal{F}(X)) = \frac{1}{1 + e^{-z(\mathcal{F}(X) - 0.5)}} \quad (9)$$

where we choose $z = 1000$ for this experiment.

Fig. 21 shows how different values of z affect the output. For $z = 1000$, a change in the the probability of a class (whether *real* or *fake*) is not reflected at the output, thus hiding the gradients. Consequently, the attacker is unable to estimate the contribution of a word because the output does not change.

We attack the *obfuscated*-MLP fake-news classifier using the Text-bugger, Text-fooler, and PWS attacks. Although all three attacks initially got a 100% ASR against MLP classifier in Fig. 14(a), none of the attacks could achieve an ASR of more than 10% in this experiment. In the past, researchers have confused obfuscation with the robustness [17]. Although we find plenty of image-based attacks to overcome such limitations, to the best of our knowledge, there is no attack in the current NLP literature that addresses this issue. We, therefore, identify a need for an iterative hard-label black-box attack against NLP classifiers to evaluate defenses more fairly.

D. Future Directions

Improving the computational efficiency of *Con-Detect*. We motivate future researchers to extend our idea to make *Con-Detect* computationally more efficient. One way to achieve this is to divide an input sequence into several chunks, where

each chunk encloses multiple subsequent words. Instead of computing the C_F -score for each word, we can compute C_F -scores of each chunk enclosing multiple words, which should considerably reduce the time overhead. However, such a modification to *Con-Detect* methodology may increase the error rate, i.e., false-positives and false-negatives. A detailed investigation is left for future work.

Extension to the image and speech classification tasks. Similar to the NLP attacks, image-based attacks aim to change a model's output by minimally perturbing its input image. We hypothesize that such constraints should imprint detectable adversarial features on an input. In the future, we plan to extend *Con-Detect* methodology to the audio and visual classification tasks where inputs are more continuous. However, we note that our assumptions in Section III-B are particular to the NLP classification tasks, and thus our work cannot be trivially extended to other domains without careful adaptation.

VII. CONCLUSIONS

In this work, we propose *Con-Detect*, a novel methodology to detect adversarial inputs to natural language processing (NLP) classifiers at the runtime. We show that these adversarially perturbed inputs can be detected by using our proposed technique of cumulative contribution scores denoted C_F -scores since these scores are higher for adversarial inputs when compared to clean inputs. We identify the reason underpinning the difference between adversarial and clean NLP input as the stealthiness of adversarial attacks, due to which adversarial attacks only aim to minimally perturb an input to change a classifier's decision for the sake of evading detection. These minimal perturbations, however, are also highly contributing perturbations when viewed through the lens of our proposed technique of cumulative contribution scores. We leverage the fact that the technique of minimalist perturbations, considered a strength of adversarial attacks, can also act as its Achilles' heel and help us in detecting these adversarial attacks. Further, we propose an alarm threshold mechanism to detect adversarial inputs based on their C_F -scores.

In addition, we also propose a novel modification to the state-of-the-art attacks, the so-called adaptive attacks, to specifically target *Con-Detect*. We perform black-box attacks assuming *Con-Detect*-blind, *Con-Detect*-aware, and *Con-Detect*-adaptive attackers and show that *Con-Detect* can detect adversarial attacks with high precision and accuracy. Although an adaptive attacker may considerably degrade *Con-Detect* performance, the attacker only achieves this at the cost of increased queries and the percentage of words perturbed, which suggests that *Con-Detect* is robust.

ACKNOWLEDGMENT

This research has been funded by Deputy for Research & Innovation, Ministry of Education through Initiative of Institutional Funding at University of Ha'il-Saudi Arabia through project number IFP-2004.

REFERENCES

- [1] F. Khalid, H. Ali, M. A. Hanif, S. Rehman, R. Ahmed, and M. Shafique, "FaDec: A Fast Decision-based Attack for adversarial machine learning," in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–8.
- [2] A. Qayyum, M. Usama, J. Qadir, and A. Al-Fuqaha, "Securing connected & autonomous vehicles: Challenges posed by adversarial machine learning and the way forward," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 998–1026, 2020.
- [3] N. Petrick *et al.*, "SPIE-AAPM-NCI BreastPathQ challenge: an image analysis challenge for quantitative tumor cellularity assessment in breast cancer histology images following neoadjuvant treatment," *Journal of Medical Imaging*, vol. 8, no. 3, p. 034501, 2021.
- [4] H. Ali, S. Nepal, S. S. Kanhere, and S. Jha, "HaS-Nets: A heal and select mechanism to defend dnns against backdoor attacks for data collection scenarios," *arXiv preprint arXiv:2012.07474*, 2020.
- [5] N. Carlini and D. Wagner, "Adversarial examples are not easily detected: Bypassing ten detection methods," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, 2017, pp. 3–14.
- [6] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.
- [7] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.
- [8] A. C. Serban, E. Poll, and J. Visser, "Adversarial Examples-a complete characterisation of the phenomenon," *arXiv preprint arXiv:1810.01185*, 2018.
- [9] S. Latif, R. Rana, and J. Qadir, "Adversarial machine learning and speech emotion recognition: Utilizing generative adversarial networks for robustness," *arXiv preprint arXiv:1811.11402*, 2018.
- [10] H. Ali, M. S. Khan, A. AlGhadhban, M. Alazmi, A. Alzamil, K. Al-utaibi, and J. Qadir, "Analyzing the robustness of fake-news detectors under black-box adversarial attacks," *IEEE Access*, 2021.
- [11] J. Morris, E. Lifland, J. Y. Yoo, J. Grigsby, D. Jin, and Y. Qi, "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*, 2020, pp. 119–126.
- [12] D. Pruthi, B. Dhingra, and Z. C. Lipton, "Combating adversarial misspellings with robust word recognition," *arXiv preprint arXiv:1905.11268*, 2019.
- [13] L. Li, R. Ma, Q. Guo, X. Xue, and X. Qiu, "Bert-Attack: Adversarial attack against bert using bert," *arXiv preprint arXiv:2004.09984*, 2020.
- [14] S. Garg and G. Ramakrishnan, "BAE: Bert-based adversarial examples for text classification," *arXiv preprint arXiv:2004.01970*, 2020.
- [15] F. Tramer, N. Carlini, W. Brendel, and A. Madry, "On adaptive attacks to adversarial example defenses," *arXiv preprint arXiv:2002.08347*, 2020.
- [16] F. Khalid, H. Ali, H. Tariq, M. A. Hanif, S. Rehman, R. Ahmed, and M. Shafique, "QuSecNets: Quantization-based defense mechanism for securing deep neural network against adversarial attacks," in *2019 IEEE 25th International Symposium on On-Line Testing and Robust System Design (IOLTS)*. IEEE, 2019, pp. 182–187.
- [17] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," in *International Conference on Machine Learning*. PMLR, 2018, pp. 274–283.
- [18] J. Li, S. Ji, T. Du, B. Li, and T. Wang, "TextBugger: Generating adversarial text against real-world applications," *arXiv preprint arXiv:1812.05271*, 2018.
- [19] B. Liang, H. Li, M. Su, P. Bian, X. Li, and W. Shi, "Deep text classification can be fooled," *arXiv preprint arXiv:1704.08006*, 2017.
- [20] J. Zeng *et al.*, "Certified Robustness to Text Adversarial Attacks by Randomized [MASK]," *arXiv preprint arXiv:2105.03743*, 2021.
- [21] D. Jin, Z. Jin, J. T. Zhou, and P. Szolovits, "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*, vol. 34, no. 05, 2020, pp. 8018–8025.
- [22] S. Ren, Y. Deng, K. He, and W. Che, "Generating natural language adversarial examples through Probability Weighted Word Saliency," in *Proceedings of the 57th annual meeting of the association for computational linguistics*, 2019, pp. 1085–1097.
- [23] J. Ebrahimi, A. Rao, D. Lowd, and D. Dou, "HotFlip: White-box adversarial examples for text classification," *arXiv preprint arXiv:1712.06751*, 2017.
- [24] G. Bekoulis, J. Deleu, T. Demeester, and C. Develder, "Adversarial training for multi-context joint entity and relation extraction," *arXiv preprint arXiv:1808.06876*, 2018.
- [25] Y. Belinkov and Y. Bisk, "Synthetic and natural noise both break neural machine translation," *arXiv preprint arXiv:1711.02173*, 2017.
- [26] M. Sato, J. Suzuki, H. Shindo, and Y. Matsumoto, "Interpretable adversarial perturbation in input embedding space for text," *arXiv preprint arXiv:1805.02917*, 2018.

- [27] Y. Zhou, J.-Y. Jiang, K.-W. Chang, and W. Wang, "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)*, 2019, pp. 4906–4915.
- [28] W. Wang, R. Wang, J. Ke, and L. Wang, "TextFirewall: Omni-Defending against adversarial texts in sentiment classification," *IEEE Access*, vol. 9, pp. 27 467–27 475, 2021.
- [29] J. A. Nasir, O. S. Khan, and I. Varlamis, "Fake news detection: A hybrid CNN-RNN based deep learning approach," *International Journal of Information Management Data Insights*, vol. 1, no. 1, p. 100007, 2021.
- [30] H. Ali, F. Khalid, H. A. Tariq, M. A. Hanif, R. Ahmed, and S. Rehman, "SSCNets: Robustifying DNNs using secure selective convolutional filters," *IEEE Design & Test*, vol. 37, no. 2, pp. 58–65, 2019.
- [31] M. T. Ribeiro, S. Singh, and C. Guestrin, "'why should I trust you?'" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.
- [32] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," in *International Conference on Machine Learning*. PMLR, 2017, pp. 3319–3328.
- [33] M. Hughes, I. Li, S. Kotoulas, and T. Suzumura, "Medical text classification using convolutional neural networks," *Stud Health Technol Inform*, vol. 235, pp. 246–50, 2017.
- [34] S. Wang, M. Huang, and Z. Deng, "Densely connected CNN with multi-scale feature attention for text classification," in *IJCAI*, 2018, pp. 4468–4474.
- [35] C. Li, G. Zhan, and Z. Li, "News text classification based on improved Bi-LSTM-CNN," in *2018 9th International Conference on Information Technology in Medicine and Education (ITME)*. IEEE, 2018, pp. 890–893.
- [36] J. Zhang, Y. Li, J. Tian, and T. Li, "LSTM-CNN hybrid model for text classification," in *2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*. IEEE, 2018, pp. 1675–1680.
- [37] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh, "Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models," in *Proceedings of the 10th ACM workshop on artificial intelligence and security*, 2017, pp. 15–26.