applied sciences

Article

A Hybrid Adversarial Attack for Different Application Scenarios †

Xiaohu Du *,‡, Jie Yu ‡, Zibo Yi, Shasha Li, Jun Ma, Yusong Tan and Qinbo Wu

College of Computer Science and Technology, National University of Defense Technology, Changsha 410073, China; yj@nudt.edu.cn (J.Y.); yizibo14@nudt.edu.cn (Z.Y.); shashali@nudt.edu.cn (S.L.); majun@nudt.edu.cn (J.M.); yusong.tan@nudt.edu.cn (Y.T.); qinbo.wu@nudt.edu.cn (Q.W.)

* Correspondence: duxiaohu18@nudt.edu.cn; Tel.: +86-155-8098-5235

† This paper is an extended version of our paper published in the 6th International Conference on Artificial Intelligence and Security (ICAIS 2020).

‡ These authors contributed equally to this work.

Received: 28 April 2020; Accepted: 15 May 2020; Published: 21 May 2020

Abstract: Adversarial attack against natural language has been a hot topic in the field of artificial intelligence security in recent years. It is mainly to study the methods and implementation of generating adversarial examples. The purpose is to better deal with the vulnerability and security of deep learning systems. According to whether the attacker understands the deep learning model structure, the adversarial attack is divided into black-box attack and white-box attack. In this paper, we propose a hybrid adversarial attack for different application scenarios. Firstly, we propose a novel black-box attack method of generating adversarial examples to trick the word-level sentiment classifier, which is based on differential evolution (DE) algorithm to generate semantically and syntactically similar adversarial examples. Compared with existing genetic algorithm based adversarial attacks, our algorithm can achieve a higher attack success rate while maintaining a lower word replacement rate. At the 10% word substitution threshold, we have increased the attack success rate from 58.5% to 63%. Secondly, when we understand the model architecture and parameters, etc., we propose a white-box attack with gradient-based perturbation against the same sentiment classifier. In this attack, we use a Euclidean distance and cosine distance combined metric to find the most semantically and syntactically similar substitution, and we introduce the coefficient of variation (CV) factor to control the dispersion of the modified words in the adversarial examples. More dispersed modifications can increase human imperceptibility and text readability. Compared with the existing global attack, our attack can increase the attack success rate and make modification positions in generated examples more dispersed. We've increased the global search success rate from 75.8% to 85.8%. Finally, we can deal with different application scenarios by using these two attack methods, that is, whether we understand the internal structure and parameters of the model, we can all generate good adversarial examples.

Keywords: adversarial attack; adversarial example; sentiment classification; deep learning

1. Introduction

In the past few decades, machine learning and deep learning techniques have achieved great success in some applications. However, so far, there are some technologies that have proven to be vulnerable. Some modified inputs can be easily distinguished by humans, but the neural network model will be classified incorrectly [1]. Adversarial attacks on neural networks have attracted a lot of attention. The main target of these attacks is a computer vision model for image classification [2,3]. Since the input features of these models are continuous, we can apply artificially indistinguishable perturbations. Unlike image data, text input consists of individual words represented by word embeddings, but we cannot directly perturb them and find another word because we need to consider whether the context and the grammar are smooth. We call the perturbed words an adversarial example.

The adversarial example is a small modification to the input, which can change the judgment of the classifier but is not easily detected by humans. These adversarial examples expose some of the vulnerabilities of classifiers and can be used to evaluate and improve machine learning models. Liang et al. [4] pointed out that, if we want to produce a very effective adversarial example, we need to solve two problems. The first is to maintain the original meaning of the adversarial example sentences. The second problem is making it difficult for humans to detect that the adversarial examples are modified. Adversarial attacks can be divided into two cases: white-box attacks and black-box attacks. The former can obtain the parameters and structure of the model, while the latter lacks this information. Black-box attacks are more challenging because they often require a large number of queries on the model. White-box attacks have some gradient-based methods for specific models. Alzantot et al. [5] proposed a black-box based genetic algorithm (GA) to generate semantic and grammatically similar adversarial examples. These examples are used to obfuscate the sentiment classification and textual entailment models with good training results; their attack success rates have reached 97% and 70%, respectively. Based on the above work, Wang et al. [6] proposed an improved genetic algorithm (IGA). Compared with GA, IGA can achieve higher attack success rates while maintaining the transferability of the adversarial examples. The use of differential evolution (DE) for generating adversarial images has be proved to be effective [7]. Both GA and DE are population based optimization algorithms. DE has mechanisms in the population selection phase that maintains the diversity. For that, we propose a novel black-box attack for generating adversarial text examples based on differential evolution.

When we understand the model parameters and structure, we can use white-box attack. Most white-box attacks are based on gradient methods. Greedy attack has proven to be a very effective attack method [8]. However, it also has some problems. Tsai et al. [9] pointed out that greedy attack may not guarantee optimal results. Another limitation is that substitute words often appear in the close range of the sentence, especially in front. This greatly reduces the readability of the sentence and even destroys the semantic meaning of the original text. They proposed a "global search" algorithm that computes a perturbation to obtain candidate words, and then replaces the words of the more perturbed locations. The larger the perturbation, the more sensitive the classifier is to changes in the word. Global search results produce better examples than greedy attack and have a higher attack success rate. When generating candidate words, the global search only uses the Euclidean distance to find the nearest word. We consider combining Euclidean distance and cosine distance to find similar words, and find that some of the original attack failed examples can be re-attacked successfully when we use the cosine distance. In the process of generating the adversarial examples, the existing methods modify the word without considering the dispersion of the modified position in the whole sentence. In particular, the word replacement position of a greedy attack is often close in the sentence, which greatly reduces the readability. Our method is essentially to use different metrics to measure embedding distance to find similar words in generating candidate words and introduce the CV in the Generate Adversary Function in [9]. We compare results of different CV weight on the success rate of the attack, and finally get the best CV weight for generating adversarial examples.

In summary, our contributions in this work are as follows:

(1) We propose a novel black-box attack for generating adversarial text examples based on differential evolution.
(2) We use a Euclidean distance and cosine distance combined metric to find the similar words when generating perturbations and candidate words; the results show that the combining metric can increase the attack success rate.
(3) We propose a white-box attack to generate adversarial examples in which the modified words have high semantic relevance and their positions are more dispersed.

(4) Finally, we prove that the global search attack with a coefficient of variation is more similar to the original text and more imperceptible for humans, which is verified by human evaluation.

The rest of this paper is organized as follows: In Section 2, we describe the related works done by the previous scholars, and some of them are our comparison objects. In Section 3, we describe our methods in detail. Section 4 introduces the relevant contents of the experiment, including the target model, data set, experimental parameters, experimental results, and so on. Finally, we analyze and summarize the experiment. Section 5 introduces the final conclusions and future work.

2. Related Work

The first adversarial attack originates in 2014, when Szegedy et al. [1] find that deep neural networks used for image classification could be tricked by a tiny pixel perturbations added to the input image. They found that image classifiers have a high rate of misclassification, but humans have not detected such changes in images. In 2017, Jia et al. [10] are the first to consider generating adversarial examples on text-based deep neural networks. Since then, people have begun to pay attention to generating adversarial examples for text.

2.1. Black-Box Attack on Text

Robin et al. [10] propose a black-box attack in 2017; this is the first job for the reading comprehension system. The attack proposed by the author is to add distracting but meaningless sentences to the end of the paragraph. These distracting sentences do not change the semantics of paragraphs and answers, but they can trip up the neural network model. The distracting sentence can be a carefully generated real sentence or an arbitrary sequence of words using 20 random common words. Finally, the attack is considered successful when the neural network is queried iteratively until the output changes. Later, Yicheng et al. [11] improve the work by changing the position of noise words and expanding the false answer set used to generate the noise words. They provide new adversarial examples that can help train more robust neural models. Yonatan et al. [12] propose in 2018 that interfered with the input data of neural machine translation applications in two ways: synthesis, which changes the order of characters, such as swapping, and randomization in the middle (that is, except for the first and last characters, Randomly change the order of characters); and completely random (that is, randomly change the order of all characters) and keyboard typing error types. They also collect typographical and typographical errors as adversarial examples. These take advantage of typos in the data set. Gao et al. [13] present a novel algorithm, DeepWordBug. They use a new scoring strategy to identify key tokens. If these tokens are modified, it will cause the classifier to misclassify. They use simple character substitution for the highest-ranking tokens to minimize the edit distance of the perturbation, and change the original classification. Their method has achieved good results in tasks such as text classification, sentiment analysis, and spam detection, and reduces the prediction accuracy of the current state-of-the-art deep learning models. In the same year, Alzantot et al. [5] used genetic algorithms (GA) to minimize the number of word replacements in the original text, but, at the same time, they can make the model wrong. They use crossover and mutation operations in genetic algorithms to generate disturbances. Their attack targets are sentiment classification and textually implied DNN models. In 2019, Ren et al. [14] propose a greedy algorithm called probability weighted word saliency (PWWS) for text ad-versarial attack. They think that, compared with images, the main difficulty of generating text adversarial examples is that the text space is discrete, and it is difficult to make small perturbations along the gradient direction. The challenge is that the generated results need to ensure vocabulary correctness, grammatical correctness, and semantic similarity. Therefore, on the basis of the synonym replacement strategy, they introduce a new vocabulary replacement method that is determined by the POS saliency and classification probability. Wang et al. [6] propose an improve genetic algorithm (IGA) based on the above genetic algorithm. Compared with existing genetic-based adversarial attacks, IGA changed from the random initialization of the GA to the first population. Synonyms randomly replace each word to initialize the first population, making the population more diverse. It also allows replacing previously replaced words to avoid local minima. In the cross section, in order to better simulate breeding and biological crossing, the text of the two parents is randomly cut, and the two fragments are merged into a new text, instead of randomly selecting a word from each position of the two parents. In 2020, Jin et al. [15] propose a black-box attack method called TEXTFOOLER. Different from the above method, they define a new method for ranking the importance of words; TEXTFOOLER first obtains the prediction score $F_Y(X)$ for the Y label corresponding to the original text X through the model, and the score $F_Y(X \backslash w_i)$ of X after deleting the word w_i. Then, it calculates the importance score of w_i according to the relationship between $F_Y(X)$, $F_Y(X \backslash w_i)$, Y, and $\widehat{Y}$. Finally, it replaces the words according to the importance score. This method achieves good results in text classification and text implication tasks.

To sum up, the above black-box attack method consists of many similarities. Because the black-box does not know the internal structure and specific parameters of the model, scholars attempt to find the optimal replacement in the word space by different methods. Some of them replace the words first and then calculate the prediction score after the replacement, such as multiple iterations of GA to find the optimal replacement. At the same time, they also try to find out the important score of the replacement words and then replace them in order to minimize the word replacement ratio, such as DeepWordBug and TEXTFOOLER. However, they all have the same goal, which is to find an optimal adversarial examples.

2.2. White-Box Attack on Text

In terms of white-box attacks, the related works are not as much as black-box attacks, which are mainly based on gradient methods. Ebrahimi et al. [16] propose a gradient-based white-box attack method to generate adversarial examples. This attack aims at a character level RNN classifier and greatly increases the error rates on text classification and machine translation.Cheng et al. [17] propose a white-box method to generate adversarial examples against NMT and improve the robustness of NMT. Greedy attack proves to be a very effective method of attack [8,9,18], and Yang et al. [8] even show that greedy attack achieves state-of-the-art attack rates across various kinds of models. However, it also has some problems. Tsai et al. [9] propose that it may not guarantee producing the optimal results and sometimes spends too much time because the algorithm needs to search the candidate words for every iteration, at the same time, due to the nature of being greedy, the algorithm can replace sub-optimal words that do not contribute much to the final goal in an earlier position. Another limitation is that replaced words tend to be in a close area of the sentence, especially in the front. This greatly reduces the readability of the sentence and even destroys the semantic meaning of the original text. They propose a "Global Search" algorithm, which obtains candidate words by calculating a perturbation, and then replace the words in the position where the perturbation is larger. The larger the perturbation, the more sensitive the classifier is to the change of the word. Results of global search prove to generate better examples than the greedy attack and higher attack success rate, which is the baseline used in our experiments. In addition, Lei et al. [19] propose a greedy method can be very time consuming when the space of attacks is large and give the optimization scheme.

The related work of white-box attacks have proved to be very effective in text-based attacks, but there are still some problems to be solved, such as a good algorithm in the replacement percentage, attack success rate, word similarity, grammatical correctness, and semantic similarity can still have better results. Based on the strategy of gradient generation perturbation, we propose a new white-box attack method for sentiment classification task and make some progress in the above aspects.

2.3. Different Measures of Textual Similarity

An important problem is that the generated adversarial examples must not only fool the target model, but also keep the perturbation undetected. A good adversarial example should convey the same semantics as the original text, so we need some metrics to quantify the similarity. There are three metrics that used on vectors and documents [20,21].

Euclidean Distance. In text, Euclidean Distance measures the linear distance between two vectors in Euclidean space.

Cosine Similarity. Cosine similarity represents the semantic similarity of two words by calculating the cosine of the angle between two vectors. Cosine distance is more concerned with the direction difference between two vectors. The smaller the angle between two vectors (the larger the cosine value), the greater the similarity.

Word Movers Distance (WMD). WMD [22] mainly reflects the distance between documents, so it is not suitable for finding similar words. Its semantic representation can be based on the embedding vector obtained by word2vec. Of course, it can also be based on other word embedding methods. This algorithm constructs the document distance into a combination of the semantic distances of the words in the two documents. For example, the Euclidean distance is obtained from the word vectors corresponding to any two words in the two documents and then weight and sum. The WMD distance between two documents A and B is:

$$WMD(A,B) = \sum_{i,j=1}^{n} T_{ij} \cdot D(\overrightarrow{i}, \overrightarrow{j}) \tag{1}$$

where $D(\overrightarrow{i}, \overrightarrow{j})$ is the Euclidean distance of the word vectors corresponding to the two words i and j. The Bag Of Words model is used to get the word frequency of the word in the document (as the weight of a word in the document), and the problem then becomes how to "carry" all the word units of document A to the corresponding word units of document B with the minimum cost, and finally get the weight matrix T. The WMD algorithm is a special case of the EMD [23] algorithm, and some improved algorithms based on WMD, such as WCD and RWMD.

3. Methods

3.1. Black-Box Attack on Text

In the black-box attack, we adopt the method based on a differential evolution (DE) algorithm to generate an adversarial example, a differential evolution algorithm with a genetic algorithm (GA) has some similarities. It has the choice stage to keep a population diversity mechanism; therefore, in practice, it is expected to be more effective than other types of group evolution algorithms to find a higher quality solution. In a black-box setting, the attacker does not know the model architecture and parameters. They can only use the input provided to query the target model to get prediction results and corresponding confidence scores. The whole algorithm of black-box attack is described in Algorithm 1, which basically follows the idea of differential evolution algorithms. The overall structure of the differential evolution algorithm is similar to the genetic algorithm. It also has mutation, crossover, and selection operations, but it is different from the genetic algorithm. The main difference from the basic genetic algorithm is the mutation operation. For example, in the genetic algorithm, two children are generated by the intersection of two parent individuals, and, in the differential evolution algorithm, the difference vector of two or several individuals is used to perturb. Then, we will get the new individuals. In traditional genetic algorithms, offspring individuals replace their parent individuals with a certain probability, and newly generated individuals in differential evolution only replace individuals in a population when they are better than individuals in the population. In traditional genetic algorithms, offspring individuals replace their parent individuals with a certain probability, and newly generated individuals in differential evolution only replace individuals in a population when they are better than individuals in the population.

Algorithm 1: Black-box Attack on Text

```
Input: Original text X, target model f, maximum iterations G, initial generation size S
Output: Adversarial text $X_{adv}$
1: $y \leftarrow f(X)$
2: for $i = 1 \to S$ population do
3:   $\mathcal{P}_i^0 \leftarrow Mutate(X, w_i)$
4: for $g = 1 \to G$ generations do
5:   for $i = 1 \to S$ population do
6:     $F_i^{g-1} = f(\mathcal{P}_i^{g-1})$
7:   $X_{adv} = argmin(F_i^{g-1})$
8:   if $f(X_{adv}) \neq y$ then return $X_{adv}$
9:   $\mathcal{P}_1^g \leftarrow X_{adv}$
10:  for $i = 2 \to S\ population$ do
11:    $\mathcal{P}_i^g \leftarrow Differential(\mathcal{P}_i^g, \mathcal{P}_j^g, \mathcal{P}_k^g) \quad 3 \leq i \neq j \neq k \leq S$
12:  $X_{adv} = argmin(f(\mathcal{P}_i^g))$
13:  if $f(X_{adv}) \neq y$ then return $X_{adv}$
14:  else
15:    Randomly Sample $parent_1$, $parent_2$ from $\mathcal{P}^{g-1}$
16:    $child = Crossover(parent_1, parent_2)$
17:    if $f(child) \neq y$ then return $child\ as\ X_{adv}$
18:    $\mathcal{P}_i^g \leftarrow Mutate(child, w)$   Randomly word $w$ in child
19:    if $f(\mathcal{P}_i^g) \neq y$ then return $child\ as\ X_{adv}$
```

The algorithm consists of the following steps:

Step 1: Mutation Given a text of n words $X = \{w_1, w_2, ..., w_n\}$, we look for the most similar word to replace one of them, then we get the mutated text. We use the counter-fitting method as word embedding from [24]. We use Euclidean distance to calculate the N nearest neighbors of the word to replace, whose cosine similarity with the word to be replaced are smaller than δ, and this word embedding ensures that the neighbors we find are synonyms. Then, we use the Google language model [25] to select the most context-appropriate K words, and replace a word in the original text with these words respectively. Finally, we choose the word with the lowest confidence score of the model to replace, and get the replaced text. We can get an initial population $\mathcal{P}^0$ of S individuals by repeating this process of mutation S times. The prediction score of each individual can be obtained by querying the victim model function f. In each generation of evolution, we select the individual with the lowest prediction score of the target model and make it the first individual $\mathcal{P}_1^{g+1}$ in the next generation. If the prediction tag of one of the members of the population is not the original tag, the attack is completed. However, it is rare that only one word can change the model's prediction, and most of the examples go to the next step.

If there is no sample in the initial population that can change the prediction of the target model, we do a different operation. For all individuals of the g-th generation, $\mathcal{P}^g = \{\mathcal{P}_1^g, \mathcal{P}_2^g, ..., \mathcal{P}_n^g\}$. For each individual, the algorithm continues to generate mutants according to the following formula:

$$\mathcal{P}_i^{g'} = \mathcal{P}_i^g + F \cdot (\mathcal{P}_j^g - \mathcal{P}_k^g) \qquad i = 1, 2, ..., n \tag{2}$$

where $\mathcal{P}_j^g$ and $\mathcal{P}_k^g$ are two randomly selected individuals in the population, and $i \neq j \neq k$, and F is the mutation factor, which is generally 0 to 2, so as to obtain the mutant $\mathcal{P}_i^{g'}$. If the prediction tag of one of the individuals after differential operation is not the original tag, the attack is completed. Otherwise, the algorithm proceeds to the next step.

Step 2: Crossover In this step, the new individual is obtained from the mutant individual $\mathcal{P}_i^{g'} = \{\mathcal{P}_{i1}^{g'}, \mathcal{P}_{i2}^{g'}, ..., \mathcal{P}_{in}^{g'}\}$ and the parent individual $\mathcal{P}_i^g = \{\mathcal{P}_{i1}^g, \mathcal{P}_{i2}^g, ..., \mathcal{P}_{in}^g\}$ through a crossover operation.

$$child = \begin{cases} \mathcal{P}_i^{g'} & \text{if rand[0 1]} \leq \text{CR} \\ \mathcal{P}_i^{g} & \text{if rand[0 1]} > \text{CR} \end{cases} \tag{3}$$

where rand [0 1] is a random number between [0 1], and CR is a constant between [0 1], which is called a crossover factor. The greater the value of CR, the greater the probability of crossover. If the prediction tag of one of the individuals after crossover is not the original tag, the attack is completed. Otherwise, the mutation subroutine is applied to the resulting children.

Compared with the genetic algorithm, the black-box attack based on the differential evolution algorithm has a higher attack success rate and a smaller word replacement rate. At the same time, it is also superior to GA in terms of running time because GA uses Perturb every time a new word is generated. Perturb is a multi-step process of finding similar word replacements, which will take a lot of time. The more words a text needs to replace, the more time it takes, such as long sentences. Our method only takes more time to generate the first generation of individuals. In the subsequent evolution process, the difference operation is used to perform subtraction and multiplication based on the existing words, which takes very little time.

3.2. Greedy Search Attack

Greedy search attack is our comparative experiment. We follow the greedy search algorithm in [9,18], which starts from the first word and then selects the word that has the greatest influence on the success of the attack according to different labels in the k nearest of each word. To find the word with the closest Euclidean distance or cosine distance in word space E, if the modification of the word causes the classifier logit output to be larger than the original and the label is pos or the logit output is smaller than the original and the label is neg, the word is replaced. Otherwise, the word is not modified. Finally, the attack is successful until the label of the classifier output is different from the original, and the total number of replacement words is lower than our threshold.

3.3. White-Box Attack on Text

The white-box attack algorithm is described in Algorithm 2. The algorithm is divided into two parts including generating candidate words and modifying the candidate words which have the most influence on the classifier. We use the whole text as the research object. X represents word embedding of the original input text. $y \ and \ \hat{y}$ represent the original and adversarial label. For a text of n words, $X = \{w_1, w_2, ..., w_n\}$, and a valid adversarial example X_{adv} should conform to the following requirements:

$$\hat{y} \neq y, \ and \ Sim(w_i, w_i^{'}) \leq threshold \tag{4}$$

where Sim is the distance between w_i and $w_i^{'}$ in the word space. It should be less than a threshold. In the first loop, we obtain a perturbed embedding $X^{'}$ by the gradient-based method, and then traverse every word in $X^{'}$, in order to find the word with the closest Euclidean distance or cosine distance in word space E, and then generate a list of candidate words. In this candidate words list, every word in the original text has been modified. The subsequent second loop will choose to modify these words successively according to priority. We attack the classifier with this list. If the label is not the same as the original input, the loop ends. Otherwise, the algorithm continues to calculate the gradient to get a new $X^{'}$. After obtaining a list of valid candidate words, we will make a modification order selection.

Algorithm 2: White-box Attack on Text

```
**Input:** Original text $\mathrm{X} = \{\mathrm{w}_1, \mathrm{w}_2, ..., \mathrm{w}_\mathrm{n}\}$, target model f, perturbation $\delta$
**Output:** Adversarial text $\mathrm{X}_{\mathrm{adv}}$
1:  $y \leftarrow f(X)$
2:  Initialization : $X_{adv} \leftarrow X,\ \delta \leftarrow 0$, success $= False$, candidates $\leftarrow \varnothing$, finCandidates $\leftarrow \varnothing$
3:  Filter out the stop words in X.
4:  **while** *not success* **do**
5:    $X^{'} \leftarrow X + \delta$        $\triangleleft$ *back propagation*
6:    **for** $w_i$ *in* $X^{'}$ **do**
7:      candidates $\leftarrow$ extracting the top N synonyms using Euclidean and Cosine
        distance for $w_i$.
8:      candidates $\leftarrow$ POScheck(candidates)
9:      **for** $k_i$ *in candidates* **do**
10:       $X^{'} \leftarrow$ Replace $w_i$ with $k_i$ *in* $X_{adv}$
11:       $P_i \leftarrow f(X^{'})$
12:     $w^* \leftarrow argmax(P_i)$
13:     $finCandidates \leftarrow finCandidates\ append\ w^*$
14:   $\widehat{y} \leftarrow f(\text{finCandidates})$
15:   **if** $\widehat{y} \neq y$ **then** *break*
16: $grad \leftarrow ||\delta||_2$
17: $selected \leftarrow 0, list \leftarrow \varnothing, \Omega \leftarrow \varnothing$
18: **while** *not success* **do**
19:   $selectedX = selected.index(0)$
20:   **for** $x$ *in* $total(grad)$ *and* $selected[x] \neq 1$ **do**
21:   $CV_1 = CV(list\ append\ x),\ CV_2 = CV(list\ append\ selectedX)$
22:   **if** $grad[x] + \lambda \cdot CV_1 > grad[selectedX] + \lambda \cdot CV_2$ **then**
23:     $selectedX \leftarrow x$
24:   $\Omega \leftarrow \Omega\ append\ x$
25:   $selected[selectedX] = 1, list \leftarrow list\ append\ selectedX$
26:   **for** $w_i$ *in* $\Omega$ **do**
27:     **if** $distance < threshold\ and\ \frac{i}{len(x)} < r$ **then** $X_{adv} \leftarrow$ Replace $w_i$
28:     **if** $\widehat{y} \neq y$ **then**
29:       $success = True$
30:       **return** $X_{adv}$
31:     **else**
32:       **return** None
```

In the second loop, first, we will take the gradient of the first loop. We set a full 0 tensor with the same dimensions as the original text, with *selected* for recording the position of the selected word, *list* for the sequence of selected words, and *positions* for the last modified order. Then, start the loop selection. First, we select the index number *selectedX* whose value is 0 from the *selected*, the number of words in the text is the traversal range, and the variable is represented by x, when *selected* $[x] \neq 1$, calculating the coefficient of variation CV_1 of x added to the *list* and the CV_2 of the *selectedX* added to the *list*, if the weighted sum of gradient and CV_1 is greater than the sum of gradient and CV_2, we then replace *selectedX* with x, add x to *positions*, update *selected* and *list*, let *selected* $[selectedX]$ = 1, and add *selectedX* to *list*. Finally, a modification sequence is generated. Then, we modify them in order until the output of the classifier is different from the original text. In the process, we can set the word distance threshold, word replacement rate, and other parameters, and compare different results under different parameters. The whole attack process of white-box attack is shown in Figure 1.

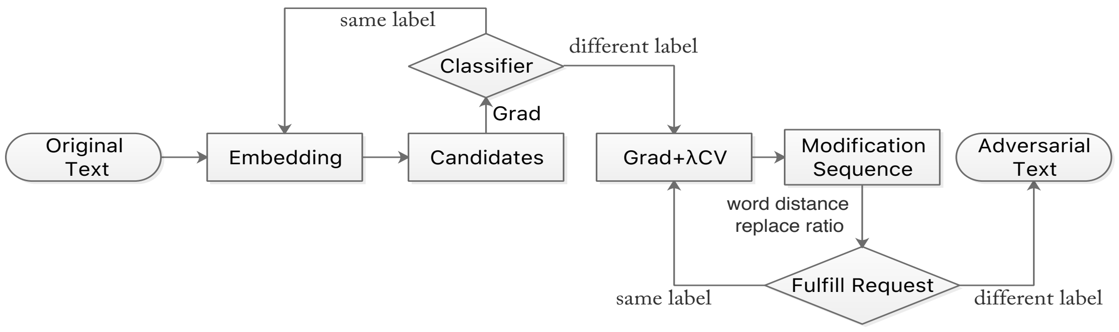

Figure 1. The whole attack process of white-box attack.

Simply put, we combine the gradient of the word we want to modify and coefficient of variation of the text position of all modified words after the word is modified, and use these two data to determine the order of modification.

We can control the degree of dispersion of the word's position by the weight value λ of the CV. When λ is too large, the modified words will be more dispersed. The order of modification is closer to the global search algorithm when λ is smaller; when $\lambda = 0$, the algorithm becomes the global search.

4. Experiments

4.1. Dataset and Target Model

The dataset contains 25,000 IMDB film reviews specifically for sentiment analysis. The mood of the comment is binary. Labels are pos(positive) and neg(negative). We randomly segment 25,000 data into 20,000 examples as training sets and 5000 examples as test sets.

We use the CNN model and some of its hyperparameters [26] as our target model. In this model, there are filter windows of 3, 4, and 5 with 100 feature maps each, a dropout rate of 0.5, and a max pooling layer. The batch size has been modified to 64. We use 20,000 examples to train the model and test the model with 5000 examples. The results of this model are that the accuracy of training sets is 1 and the accuracy of testing sets is 0.89.

4.2. Evaluation

4.2.1. Black-Box Attack Based on Differential Evolution

We randomly sample 200 correctly classified examples from the IMDB test set to evaluate our algorithm. We choose examples of correct classification to avoid the impact of model accuracy. Our purpose is to make the model prediction error, that is, the original prediction is positive. The model will be judged as negative after the adversarial attack, so that our attack is considered successful.

We limit the attacker to maximum G = 20 iterations. We also fix the maximum percentage of each text change to 10% and 20%. We believe that the quality of the text will decrease if the modification ratio exceeds 20%. Although the success rate of black-box attack based on a genetic algorithm has reached 97%, its word replacement rate has reached 25%. If an attack fails within the iteration limit or exceeds a specified threshold, it is considered a failure. Table 1 shows the attack success rate under different F-values and percentage of modified words. From our results, we can see that we can achieve a high success rate by making small changes to the text. In addition, our algorithm is significantly better than the genetic algorithm in word modification percentage and attack success rate.

It is worth noting that, under the 10% replacement rate threshold, GA and DE have only about a 60% attack success rate, which is far lower than the white-box attack algorithm mentioned later. It shows that there is still a lot of room for improvement in a black-box attack. Once the threshold of the word replacement rate is increased to 20%, we see that both GA and DE exceed 90% in the attack success rate, and our method even approaches 100%. This shows that the word replacement rate is an important factor affecting the success rate of attack. Algorithms that do not strictly limit the word replacement rate are not good algorithms.

Table 1. Attack success rate with different word modifiers and F values.

	GA	**F = 0.1**	**F = 0.3**	**F = 0.5**	**F = 1.2**	**F = 1.5**
10% modified	58.5%	63%	62.5%	56.5%	55%	53%
20% modified	91%	96%	94.5%	94.5%	96%	97.5%

The mutation factor F can be used to control the degree of scaling of the difference vector between two random individuals. F value has a great influence on the success rate of the final attack. Experiments show that the smaller the F value, the higher the attack success rate at low replacement rates. When the F value is larger, the attack success rate is higher at a relatively higher replacement rate. Moreover, when F is 0.3, the attack success rate of our algorithm is higher than that of the genetic algorithm in terms of 10% and 20% word substitution rate. The example output generated by a black-box attack is shown in Table 2. From the results, the examples we generated can maintain the original sentence form to some extent.

Table 2. Original text and adversarial example of black-box attack.

Original Text Prediction = **Positive**.
absolutely fantastic whatever i say wouldn't do this underrated movie the justice it deserves watch it now fantastic.
Adversarial Text Prediction = **Negative**.
absolutely fantastic whatever i say wouldn't do this underestimated movie the justice it deserve watch it now fantastic.
Original Text Prediction = **Negative**.
poorly directed short film shot on hi def or betacam it appears it screams student film video all the way the premise is limited in scope and the short actually feels a lot longer than it runs some interesting acting moments and some decent production value but not enough to lift this film from the hole it has fallen into.
Adversarial Text Prediction = **Positive**.
poorly directed gunshot film shot on hi def or betacam it appears it shrieks student film video all the way the premises is limited in scope and the short actually feels a lot longest than it runs some interesting acting moments and some decent production value but not enough to lift this film from the hole it has fallen in.

4.2.2. White-Box Attack with the Coefficient of Variation

Our experimental goal is to generate adversarial examples to confuse the classifier. As long as the prediction result of the adversarial example is different from the original comment result, the attack is considered successful. At the same time, we need to exclude the influence of classifier accuracy on the experimental results, so we select 500 correctly classified examples. Our experiments have the same word distance threshold and word replacement rate, the word Euclidean distance is set at 50 and the word replacement rate is set at 0.1, and k is set at 35 in the greedy attack. Finally, we add cosine distance experiments for greedy attack and global attack.

We calculate the attack success rate corresponding to several different cosine distance thresholds. The experimental results are shown in Figure 2. We can see that the smaller the cosine distance is, the higher the success rate the attack will have because the word angle is larger and their similarity is smaller. Greedy attack and global attack without cosine distance had original success rates of 65.8% and 75.8%. In the global attack, there is increased cosine distance, even if the cosine distance is set to 0.9848; the final success rate will increase to 77.4%, while the greedy attack has no new successful examples. There is no significant change in the number of increases between the cosine of 0.9848 and 0.6428. An angle of more than 0.6428 will have a significant increase in the number of successful attacks. In addition, we do an experiment to find candidate words only by cosine distance as a contrast. We find that, in the case of high cosine distance threshold, the attack success rate is very low, but finally when the cosine distance threshold is 0, it can also achieve a high attack success rate, which shows, that in terms of word similarity, the effect of cosine distance is worse than that of European distance. We can use the cosine distance as a supplement of European distance to improve the overall attack success rate. The evaluation of adversarial examples shows that greedy attack still shows that the replacement locations are sometimes close, or even in the front of the text, for which readability is not good, while the replacement position of global attack is relatively random. The following human evaluation also shows this conclusion. The example output generated by black-box attack is shown in Table 3.

Table 3. Original text and three adversarial examples of white-box attack.

Original Text, Prediction = **Negative**.
The Pallbearer is a disappointment and at times extremely boring with a love story that just doesn't work partly with the casting of Gwyneth Paltrow (Julie). Gwyneth Paltrow walks through the entire film with a confused look on her face and its hard to tell what David Schwimmer even sees in her. However The Pallbearer at times is funny particularly the church scene and the group scenes with his friends are a laugh but that's basically it. Watch the Pallbearer for those scenes only and fast forward the rest. Trust me you aren't missing much.
Greedy Attack Text, Prediction = **Positive**.
on despite has given tempered well outside well surprisingly boring with a love story that just doesn't work partly with the casting of Gwyneth Paltrow (Julie). Gwyneth Paltrow walks through the entire film with a confused look on her face and its hard to tell what David Schwimmer even sees in her. However The Pallbearer at times is funny particularly the church scene and the group scenes with his friends are a laugh but that's basically it. Watch the Pallbearer for those scenes only and fast forward the rest. Trust me you aren't missing much.
Global Attack Text, Prediction = **Positive**.
The Pallbearer is a artist and at times extremely boring with affection love story that just doesn't work partly with the american of Paltrow Paltrow (Julie). Paltrow Paltrow walks through the entire film with a confused look on her face and its hard to tell what David Schwimmer even sees in her. However The Pallbearer at times is funny particularly the church scene and the group scenes with his friends are a laugh but that's basically it. Watch the Pallbearer for those scenes only and fast forward the rest. Trust me you aren't missing much.
Global Attack Text with CV, Prediction = **Positive**.
The Pallbearer is affection artist and at times extremely boring with a love story that just doesn't work partly with the casting of Gwyneth Paltrow (Julie). Paltrow Paltrow walks through the entire film with a confused look on her face and its hard to tell what David Schwimmer even sees in her. However The Pallbearer at times is funny particularly the church scene and the group scenes with his friends are affection laugh but that's basically it. Watch the Pallbearer for those scenes only and fast forward the rest. Trust me you aren't missing much.

Because greedy attack may make the word modification position close and result in poor readability, global search improves this situation to some extent, but it does not have the ability to effectively control the dispersion of word modification positions because the modified word is selected from big to small according to the gradient magnitude. We propose a method to control the dispersion of the modified position by adding the coefficient of variation, and control the dispersion degree of the modified position by the weight of CV. We introduced the CV factor into the global search with a combining metric. Finally, we use global attack with the combined metric and set the cosine distance threshold to 0.9848 to maintain the word's high similarity, and compare the attack success rate under different CV weight. We select 500 comments used by greedy attacks and calculate the attack success rate under different CV weight(λ).

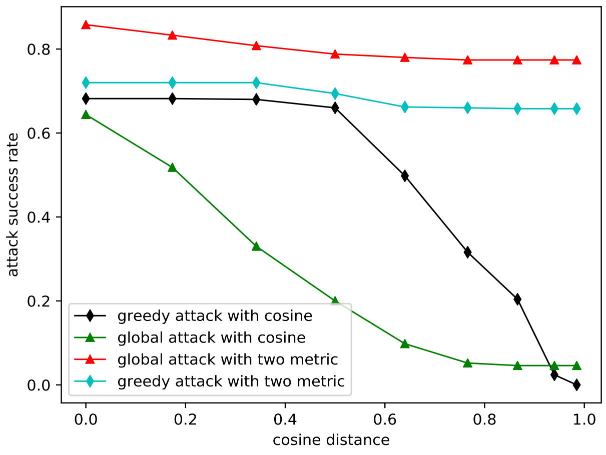

Figure 2. The total success rate of greedy attack and global attack added cosine distance.

In order to prove the validity of adding the CV factor when determining the modification order, we select 50 comments that are correctly classified by the classifier and the algorithm attack is successful, and calculate the CV value of the adversarial examples under different λ. When the CV weight value is 0, our method becomes the global search attack [9].

Experiments shown in Table 4, when λ is larger, the degree of dispersion of the modified positions of the adversarial example is larger (CV value is larger), and the final attack success rate has a slight decrease overall, which indicates that the word with a larger gradient magnitude has more influence on the classifier. We analyze the adversarial examples and find that some examples of the original attack failure will be re-attacked successfully after joining CV, and some examples of the original successful attack will fail the attack after joining CV. Taken as a whole, the attack success rate basically decreases as λ increases, but this change is very small. The experiments also show that even if we set the λ to 30 after we add the cosine distance, the attack success rate still reaches 76.2%, which still exceeds the original global attack success rate of 75.8%. Thus, our methods can improve the attack success rate and make the modification positions in an adversarial example more dispersed.

Table 4. The attack success rate and CV value under different λ.

λ	0	1	5	10	20	30
CV value	0.487	0.589	0.70	0.73	0.827	0.886
attack rate	77.4%	77.6%	76.6%	76.8%	77%	76.2%

4.2.3. Human Evaluation

In order to illustrate that adversarial text with more dispersion words modification could lead to being more similar to the original text and more imperceptible for humans, we select five volunteers to score the similarity between the adversarial examples and the original text from 1 to 5. We randomly chose 10 comments with positive and negative examples and their adversarial examples generated by greedy search, global search, and the global search with CV. This is used to prove the similarity between the original text and adversarial examples. On the other hand, we select 15 comments including the original text, greedy text, global text, and global text with CV, and then ask volunteers to identify which of them are modified adversarial examples. This is used to prove the imperceptibility for humans. After counting all the scoring results, in terms of similarity, we calculate the average score, the global text with CV similarity score is 3.642, the global text score is 3.612, and the greedy text score is 2.91. Because the greedy text often modifies the front words by a large amount, it can obviously find the difference from the original text. In terms of imperceptibility, the global text with CV has a detectable ratio of 0.37, the global text has a detectable ratio of 0.4, and the greedy text has a detectable ratio of

0.46. We find that the scores of global attack and global attack with CV are close, and they are better than greedy attacks.

5. Conclusions and Future Work

In this paper, we propose a hybrid adversarial attack for different scenarios. Specifically, we adopt different attack measures for whether we understand the deep neural network internal structure and specific parameters. If we do not understand the above information, it is suitable for us to use a black-box attack, so we propose a new black-box attack method based on a differential evolution algorithm, and generate adversarial examples with low word substitution ratio and high attack success rate. Since black-box attack does not need to know the model parameters and structure, it also has better universality to attack different natural language processing tasks. At the same time, in the case of a white-box attack, for the problem of greedy search, we propose the factors of increasing CV in the modification position of words to prevent the position of words from being too close. Our algorithm proves that the placement of word modifiers in the adversarial example can be more dispersed with the addition of CV factors. It makes up for the poor readability of the greedy attack example. Our approach maintains a high attack success rate and makes the locations of changes in the adversarial example more diffuse through the CV factor. In pursuit of a high attack success rate, we improve the quality of the adversarial example. We compare the two types of adversarial attacks above against different scenarios with existing attack methods. Our method has a certain degree of improvement in word replacement rate and attack success rate.

In our experiments, although we try to find the most similar words, some of them didn't look the same in real life, according to the example of adversarial actually generated. For example, the substitution of some synonyms does not conform to the context of movie reviews. Therefore, word embedding trained for a specific data set will greatly improve this situation. Word embedding that better reflects word similarity can also enhance our work. At the same time, the adversarial example can reveal the vulnerability of the NLP model, and we can use it to improve the robustness of the model. Although we find what is most similar to a word, in a specific context, the original words should conform the most to the original context. In defense against attack, we can use the idea of adversarial attack to restore the text. Before entering the classifier, we first preprocess the text, which will improve the security and robustness of the model. This will be future work.

Author Contributions: Writing—original draft preparation, X.D.; methodology, X.D. and J.Y.; writing—review and editing, Z.Y., J.Y. and J.M.; supervision, J.Y., Y.T. and Q.W.; funding acquisition, J.Y. and S.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the National Key Research and Development Program of China (No. 2018YFB0204301).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Szegedy, C.; Zaremba, W.; Sutskever, I.; Estrach, J.B.; Erhan, D.; Goodfellow, I.; Fergus, R. Intriguing properties of neural networks. In Proceedings of the 2nd International Conference on Learning Representations (ICLR 2014), Banff, AB, Canada, 14–16 April 2014.
2. Kurakin, A.; Goodfellow, I.J.; Bengio, S. Adversarial Examples in the Physical World. In *Artificial Intelligence Safety and Security*; Chapman and Hall/CRC: London, UK, 2018; pp. 99–112.
3. Dong, Y.; Liao, F.; Pang, T.; Su, H.; Zhu, J.; Hu, X.; Li, J. Boosting adversarial attacks with momentum. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 9185–9193.
4. Liang, B.; Li, H.; Su, M.; Bian, P.; Li, X.; Shi, W. Deep text classification can be fooled. In Proceedings of the 27th International Joint Conference on Artificial Intelligence, Stockholm, Sweden, 13–19 July 2018; pp. 4208–4215.

5. Alzantot, M.; Sharma, Y.S.; Elgohary, A.; Ho, B.J.; Srivastava, M.; Chang, K.W. Generating Natural Language Adversarial Examples. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, 31 October–4 November 2018.
6. Wang, X.; Jin, H.; He, K. Natural language adversarial attacks and defenses in word level. *arXiv* **2019**, arXiv:1909.06723.
7. Su, J.; Vargas, D.V.; Sakurai, K. One pixel attack for fooling deep neural networks. *IEEE Trans. Evol. Comput.* **2019**, *23*, 828–841. [CrossRef]
8. Yang, P.; Chen, J.; Hsieh, C.J.; Wang, J.L.; Jordan, M.I. Greedy attack and gumbel attack: Generating adversarial examples for discrete data. *arXiv* **2018**, arXiv:1805.12316.
9. Tsai, Y.T.; Yang, M.C.; Chen, H.Y. Adversarial Attack on Sentiment Classification. In Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP, Florence, Italy, 28 July–2 Augusy 2019; pp. 233–240.
10. Jia, R.; Liang, P. Adversarial Examples for Evaluating Reading Comprehension Systems. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Copenhagen, Denmark, 7–11 September 2017; pp. 2021–2031.
11. Wang, Y.; Bansal, M. Robust Machine Comprehension Models via Adversarial Training. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers), New Orleans, LA, USA, 1–6 June 2018; pp. 575–581.
12. Belinkov, Y.; Bisk, Y. Synthetic and natural noise both break neural machine translation. *arXiv* **2017**, arXiv:1711.02173.
13. Gao, J.; Lanchantin, J.; Soffa, M.L.; Qi, Y. *Black-Box Generation of Adversarial Text Sequences to Evade Deep Learning Classifiers*; IEEE: Piscataway, NJ, USA, 2018; pp. 50–56.
14. Ren, S.; Deng, Y.; He, K.; Che, W. Generating Natural Language Adversarial Examples through Probability Weighted Word Saliency. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Florence, Italy, 28 July–2 Augusy 2019; Association for Computational Linguistics: Stroudsburg, PA, USA, 2019; pp. 1085–1097. [CrossRef]
15. Jin, D.; Jin, Z.; Zhou, J.T.; Szolovits, P. Is BERT Really Robust? A Strong Baseline for Natural Language Attack on Text Classification and Entailment. *arXiv* **2020**, arXiv:2002.06261.
16. Ebrahimi, J.; Rao, A.; Lowd, D.; Dou, D. HotFlip: White-Box Adversarial Examples for Text Classification. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), Melbourne, Australia, 15–20 July 2018; pp. 31–36.
17. Cheng, Y.; Jiang, L.; Macherey, W. Robust Neural Machine Translation with Doubly Adversarial Inputs. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Florence, Italy, 28 July–2 August 2019; pp. 4324–4333.
18. Kuleshov, V.; Thakoor, S.; Lau, T.; Ermon, S. Adversarial Examples for Natural Language Classification Problems. 2018. Available online: https://openreview.net/forum?id=r1QZ3zbAZ (accessed on 20 May 2020)
19. Lei, Q.; Wu, L.; Chen, P.Y.; Dimakis, A.; Dhillon, I.; Witbrock, M. Discrete Adversarial Attacks and Submodular Optimization with Applications to Text Classification. *arXiv* **2018**, arXiv:1812.00151.
20. Zhang, W.E.; Sheng, Q.Z.; Alhazmi, A.; Li, C. Adversarial attacks on deep learning models in natural language processing: A survey. *arXiv* **2019**, arXiv:1901.06796.
21. Wang, W.; Tang, B.; Wang, R.; Wang, L.; Ye, A. A survey on Adversarial Attacks and Defenses in Text. *arXiv* **2019**, arXiv:1902.07285.
22. Kusner, M.; Sun, Y.; Kolkin, N.; Weinberger, K. From word embeddings to document distances. In Proceedings of the International Conference on Machine Learning, Lille, France, 6–11 July 2015; pp. 957–966.
23. Rubner, Y.; Tomasi, C.; Guibas, L.J. *A Metric for Distributions with Applications to Image Databases*; IEEE: Piscataway, NJ, USA, 1998.
24. Mrkšic, N.; OSéaghdha, D.; Thomson, B.; Gašic, M.; Rojas-Barahona, L.; Su, P.H.; Vandyke, D.; Wen, T.H.; Young, S. Counter-fitting Word Vectors to Linguistic Constraints. In Proceedings of the NAACL-HLT, Atlanta, GA, USA, 12–17 June 2016; pp. 142–148.

25. Chelba, C.; Mikolov, T.; Schuster, M.; Ge, Q.; Brants, T.; Koehn, P.; Robinson, T. One billion word benchmark for measuring progress in statistical language modeling. *arXiv* **2013**, arXiv:1312.3005.
26. Kim, Y. Convolutional Neural Networks for Sentence Classification. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014; pp. 1746–1751.