Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 9969867, 9 pages
https://doi.org/10.1155/2021/9969867

WILEY | Hindawi

Research Article

Exploring Security Vulnerabilities of Deep Learning Models by Adversarial Attacks

Xiaopeng Fu,[1] **Zhaoquan Gu**,[1] **Weihong Han**,[1] **Yaguan Qian**,[2] **and Bin Wang**[3]

[1]*Cyberspace Institute of Advanced Technology (CIAT), Guangzhou University, Guangzhou 510006, China*
[2]*School of Big Data Science, Zhejiang University of Science and Technology, Hangzhou 310023, China*
[3]*Network and Information Security Laboratory, Hangzhou Hikvision Digital Technology Co, Ltd., Hangzhou 310051, China*

Correspondence should be addressed to Zhaoquan Gu; zqgu@gzhu.edu.cn

Received 4 March 2021; Accepted 16 August 2021; Published 27 September 2021

Academic Editor: Federico Tramarin

Nowadays, deep learning models play an important role in a variety of scenarios, such as image classification, natural language processing, and speech recognition. However, deep learning models are shown to be vulnerable; a small change to the original data may affect the output of the model, which may incur severe consequences such as misrecognition and privacy leakage. The intentionally modified data is referred to as adversarial examples. In this paper, we explore the security vulnerabilities of deep learning models designed for textual analysis. Specifically, we propose a visual similar word replacement (VSWR) algorithm to generate adversarial examples against textual analysis models. By using adversarial examples as the input of deep learning models, we verified that deep learning models are vulnerable to such adversarial attacks. We have conducted experiments on several sentiment analysis deep learning models to evaluate the performance. The results also confirmed that the generated adversarial examples could successfully attack deep learning models. As the number of modified words increases, the model prediction accuracy becomes lower. This kind of adversarial attack implies security vulnerabilities of deep learning models.

1. Introduction

With the fast development of artificial intelligent technologies, deep learning models have been widely adopted in more and more areas [1–3]. In particular, they have been adopted not only in target detection, image classification, and other applications in the field of CV (Computer Vision) [4, 5] but also in more and more NLP (Nature Language Processing) applications, such as sentiment classification, spam classification, and machine translation [6–8].

Compared with traditional machine learning models, deep learning models have the following advantages. First, deep learning models have a strong fitting ability, which can approximate any complex function. The dimensionality of deep learning models can reach an infinite number; hence, the data fitting ability is much more powerful than traditional models. Second, deep neural networks contain many hidden layers that contain many hidden nodes; more hidden nodes are shown to provide stronger performance capabilities than traditional machine learning models. Third, the introduction of a convolutional neural network and recurrent neural network further improves the performance of neural networks, so that they can better deal with specific problems by feature extraction and contexture analysis. Finally, deep learning models can also be combined with probabilistic methods, which enable these models with high inference ability as the random factors could improve the reasoning ability of deep neural networks. Meanwhile, compared with traditional machine learning, deep learning models have better mobility, which makes the models easily adapted in various application scenarios.

Even though deep learning models play an important role in both CV and NLP fields, it does not imply that these models are completely secure and trustful. Since deep learning models lack theoretical analysis, recent studies have shown that deep learning models are very vulnerable to

adversarial attacks, which generate adversarial examples to mislead the model by adding small perturbations to the original input. These security risks may incur severe consequences such as misrecognition in security-sensitive applications and privacy leakage during the deployment and execution of deep learning models.

In this paper, we are to explore the security vulnerabilities of deep learning models by adversarial attacks. This vulnerability property of deep learning models was first discovered in the image processing field. Only a small change of one or several pixels in the original image can cause the deep learning models to output an incorrect label to the modified data. Since this change compared to the original image is very small, human eyes can hardly detect any difference, while deep learning models for image classification would make incorrect prediction, which may lead to serious consequences. For example, a driverless system may cause a serious traffic accident if the system misidentifies a STOP sign on the road.

Not only image recognition tasks but also many NLP tasks face the challenge of adversarial examples. In this paper, we focus on the adversarial attacks in the NLP field. In [9], it proved that adversarial examples could successfully attack Google perspective API, making the models output an incorrect toxicity degree. Chinese text classification models are also threatened by such adversarial examples. Compared with adversarial attacks in image processing, generating adversarial examples in the text field is quite different and much more difficult. The challenges of adversarial attacks in the NLP field include the following aspects:

(1) The text data is discrete [10]. In the image processing field, the image can be regarded as continuous data and the adversarial attacks can be conducted by traditional gradient-based methods. However, text data is discrete, and it is more difficult to adopt traditional gradient-based methods directly to generate adversarial examples for textual analysis

(2) When generating adversarial samples for image data, only one or a few pixels in the original data are modified. This modification is basically indistinguishable to human eyes. However, in the textual analysis field, even if only a character in a word is modified, it will be much easier to be caught by humans and such modification might cause people to misunderstand the meaning of the original text

Therefore, we need to address the above two when exploring security vulnerabilities in textual analysis deep learning. In this paper, we propose the visual similar word replacement (VSWR) algorithm to solve these challenges. To begin with, to solve the problem of data discreteness, the proposed VSWR algorithm directly adds perturbations to the original text, instead of mapping the original text to a vector space. Afterwards, our proposed method could use the gradient-based method to find out the appropriate word to be modified. Second, to solve the second problem, we use words that are visual similar to replace the words in the original text, which would not cause obvious differences to humans and could not be noticed easily by humans.

We summarized the contributions of this paper as follows:

(1) We proposed an algorithm called visual similar word replacement (VSWR) to generate adversarial examples for textual data, and we show the security vulnerability of the deep learning models when faced with such adversarial examples

(2) We use the VSWR algorithm to generate adversarial examples on sentiment analysis datasets, and the adversarial examples are utilized to attack the pretrained deep learning classification models

(3) The experimental results show that the generated adversarial examples can successfully interfere with the classification of the deep learning model. Specifically, only changing 25% of the original text can reduce the classification accuracy of the model from 95% to 60%

The rest of the paper is organized as follows. The next section briefly introduces related research results on textual adversarial examples. Section 3 presents the preliminaries, including the system model and the problem definition, and then proposes the VSWR algorithm. And the experimental results are provided in Section 4; the discussion is also shown here. Finally, we make a brief summary of this paper and shed light on some future directions in Section 5.

2. Related Work

2.1. White-Box Attacks. The attacker fully understands all the information of the model and conducts an adversarial attack on the model on this basis. Therefore, the attacker can find out the relatively weak module of the model to perform targeted adversarial attacks. This attack method can test the robustness of the model against adversarial attacks in the worst case.

Although there are differences between textual data and image data, the idea of generating adversarial examples in the image field can also be used in the textual field. In [11], it puts forward a method to generate text adversarial examples named HotFlip, which represents text data as one-hot vectors, then modified one character of a certain word in the text, so as to achieve the effect of attacking neural networks. In [12], it applies FGSM [13] and JSMA [14] algorithms that use gradient descent to determine the perturbation in the image domain to generate text adversarial examples.

In fact, we have little knowledge about the neural network models we are using, including the parameter value of each layer even its structure. So, gradient methods have many restrictions.

2.2. Black-Box Attacks. Because of the limitation of white-box attacks in practical scenarios, many researchers turn their attention to black-box attacks.

In black-box attacks, an attacker knows nothing about the internal structure of the attacked model, training

parameters, defense methods (if any defense methods are applied), or other information about the attacked model. The attackers can only interact with the model through input and output. Since manufacturers will not disclose information about the models they apply, most of the current contacts are black-box attacks.

In this case, the attacker generates adversarial examples by directly modifying the words in the text data or the letters/characters in the words. In [15], it proposed the AddSent method to attack the reading comprehension system by adding a carefully constructed sentence after the original text. The generated sentence could make the system make incorrect results. However, the way of adding sentences is very imperceptible, and these sentences could be easily discovered by human readers. In [16], it proposed the attack method based on the Metropolis-Hastings algorithm to replace, insert, or delete a word in the text to generate adversarial examples, while it modifies a character in the word in text in [17].

2.3. Limitation. Although much progresses have been made in attacking deep learning models, there is still much space for improvement. For example, the adversarial examples, generated by the sentence-level attack and the word-level attack, can be easily recognized by humans, while the adversarial example generated by the char-level attack can be defended by the spell check module [18]. In this paper, we propose a novel method based on the word replacement strategy of visual similar words to generate textual adversarial examples.

Figure 1 is a simple example of generating an adversarial example by the visual similar word replacement method. As shown in the figure, the original text is recognized as a positive review by the designed deep neural network model. However, we only change the word "sweet" to the word "sweat" which looks similar; the modified text is recognized as a negative review by the deep neural network model. According to the example, only changing one character of a single word in the original data could lead to a contrary label by the pretrained model.

3. Materials and Methods

3.1. Materials. Before giving our method for adversarial example generation, we show briefly the introduction of some definitions that are used in our method. In addition, we also formulate the proposed problems to explore security vulnerabilities of deep learning models.

3.1.1. System Model. We use T to represent an English text and get a word list W by segmenting the original text. An English text T which is made up of n words can be represented as $T = [w_1, w_2, \cdots, w_i, \cdots, w_n]$, where the ith value of T stands for the ith word $w_i \in W$ of this text. We use $Y = [y_1, y_2, \cdots, y_m]$ to represent the label of text T; m means that this dataset has m categories. It is expressed as a one-hot vector. For example, all the portion's values in Y_j of a text T_j with a label k are 0 except y_k. Since there are only two categories in the dataset we use, there are only two portions in Y.

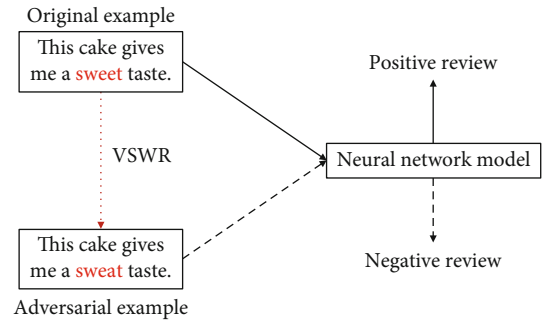

Figure 1: An example of generated texts by the visual similar word replacement algorithm.

A mapping f_θ from a text T to its label Y needs to be learned by a deep learning model which we call M, where θ are the parameters of M; they are optimized by calculating the gap of $f_\theta(T)$ and its label; the smaller of the difference between $f_\theta(T)$ and Y, the more suitable θ is.

3.1.2. Adversarial Examples. Given a well-trained model M, whenever we enter a text T_a into this model, it can give us the label Y_{T_a} of the text. An adversarial example T_a' of T_a is almost the same to T_a except a little bit of artificial perturbations δ; in this paper, δ is a visual similar word of the keyword w_i of T_a; we use $T_a' = (w_1, w_2, \cdots, \delta, \cdots, w_n)$ to represent the adversarial of T_a. When using adversarial examples T_a' as the input of model M, the model will give a different prediction $Y_{T_a'}$ from Y_{T_a}. We summarize this process into the following formulas:

$$T_a' = T_a + \delta, M(T_a) = Y_{T_a}, \quad M\left(T_a'\right) \neq Y_{T_a}. \tag{1}$$

3.1.3. Problem Definition. Since the dataset we used to verify the effect of the algorithm proposed in this paper is a binary dataset, there are only two possibilities for the label of a text T_a, $M(T_a) = 1$ or $M(T_a) = 0$. Assume that a piece of text data T_a whose label $M(T_a) = 0$. The problem we solved in this paper is to generate T_a' by the method proposed in this paper; when we use T_a' as the input of model M, $M(T_a') = 1$. And T_a' must follow the following principles:

(1) The difference between T_a and T_a' must be as small as possible, which means we can only replace a small number of words in the original data to ensure a human's reading

(2) All the visual similar words we choose to replace keywords in original data must be in word list W, and it must be spelled similarly to keywords to ensure the imperceptibility of adversarial examples

3.2. Method. In black-box attacks, the attacker knows nothing about the internal structure and parameters of the model, so it is impossible to calculate the influence of the

gradient change on the model prediction result. The method proposed in this paper is to solve the problem of the inability to pass the gradient, in the case of calculating the words that need to be modified, how to modify the original text to generate adversarial examples, which mainly includes the following steps: word scoring, visual similar word searching, and visual similar word replacement.

3.2.1. Word Scoring. Since each word in a text data has a different contribution to the final label given by pretrained models when models classify text data, for example, in sentiment analysis tasks, words with a particularly strong emotional color such as wonderful will have a greater impact on the results of the classification than other words. Therefore, when generating text adversarial examples in a black-box context, in order to ensure the success rate of the attack, the importance of the words in the original text needs to be ranked first.

According to the scores of these words, we extracted those words with higher scores which means they have the greatest impact on the text label in the original text, as "keywords," and then, adversarial examples of the original text are generated through operations on the words such as destruction or replacement. We use a method that combines context and the position of the word in the entire text to score the word. Through this method, the words in the original text are scored to obtain the words that have the greatest impact on the label.

First of all, we use the training dataset to train a neural network model M. Whenever you input a text data to M, it gives the label of this text and the confidence of each label. Since text data has strong contextual relevance, when scoring a word in the text, it is necessary to consider the context of the word. Assuming a piece of text data T consists of n words, then the text can be expressed as $T = [x_1, x_2, x_3, \cdots, x_n]$. Given a piece of text data which can be presented to $T_{\text{text}} = [x_1, x_2, x_3, \cdots, x_n]$, we give the ith word of this text by the following ways.

(1) Head Score. As we have already trained a model M to classify text data, when we give M a piece of text T_{text}, it will return the confidence of each label, and we present it by $M(T_{\text{text}})$. We define the head score of the ith word to be the score of the text composed of the first $i-1$ words minus the score of the text composed of the first i words. We first choose the first $i-1$ words to form a text T_{head}, using it as the input of model M to get $M(T_{\text{head}})$ of T_{head}. Next, ith is added to T_{head} to form T'_{head}, so that we can get M'_{head} by query model M. So, the head score of the ith word can be presented as follows:

$$S_i^{\text{head}} = M(T_{\text{head}}) - M\left(T'_{\text{head}}\right) = s(x_1, x_2, \cdots, x_{i-1}) - s(x_1, x_2, \cdots, x_i). \tag{2}$$

(2) Tail Score. The same as head score, we define the tail score of the ith word to be the score of the text composed of the words which are after the ith word minus the score of the text which added the ith word to the former text.

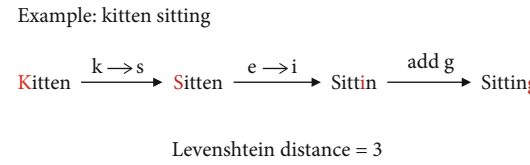

Figure 2: Levenshtein distance.

These two texts are presented as $T_{\text{tail}} = (x_{i+1}, x_{i+2}, \cdots, x_n)$ and $T'_{\text{tail}} = (x_i, x_{i+1}, \cdots, x_n)$; using these two texts, we query model M to get $M(T_{\text{tail}})$ and $M(T'_{\text{tail}})$, so the tail score of the ith word is as follows:

$$S_i^{\text{tail}} = M(T_{\text{tail}}) - M\left(T'_{\text{tail}}\right) = s(x_{i+1}, x_{i+2}, \cdots, x_n) - s(x_i, x_{i+1}, \cdots, x_n). \tag{3}$$

(4) Without Score. Without score is calculated by the text without the ith word $T_{\text{without}} = (x_1, x_2, \cdots, x_{i-1}, x_{i+1}, \cdots, x_n)$ and all of this text $T_{\text{text}} = (x_1, x_2, \cdots, x_n)$. We query these two texts above and then get $M(T_{\text{without}})$ and $M(T_{\text{text}})$. Without score is presented as follows:

$$\begin{aligned} S_i^{\text{without}} &= M(T_{\text{without}}) - M(T_{\text{text}}) \\ &= s(x_1, x_2, \cdots, x_{i-1}, x_{i+1}, \cdots, x_n) \\ &\quad - s(x_1, x_2, \cdots, x_n). \end{aligned} \tag{4}$$

(4) Combined. Since the position of the ith word in the whole text is different, a certain weight needs to be added when combining the above three scores. For the words at the top of the text, we reduce the weight of the head score, and the others are on the contrary. We determine the weight by calculating the proportion of the text before and after the ith word in the entire text. The bigger the i is, the higher the weight of the head score is. Finally, we can get the final score of the ith word through the following formula.

$$S_i^{\text{combined}} = \frac{2i/nS_i^{\text{head}} + 2(n-i)/nS_i^{\text{tail}} + S_i^{\text{without}}}{3}. \tag{5}$$

After the words are scored, the scores of the words need to be sorted. The higher the score of the word, the greater the influence of the word on the final prediction label given by the model when the model classifies the text, and this word is a "keyword" in the original text; modifying the "keywords" can improve the offensiveness of the adversarial samples and increase the attack success rate.

3.2.2. Finding Visual Similar Words. When generating text adversarial examples, the imperceptibility of adversarial examples needs to be considered (that is, the modified text cannot be changed too much from the original text). Therefore, the adversarial example generation algorithm proposed in this paper selects words that are similar in spelling to the keywords in the original text when replacing keywords. But there are many ways to calculate the similarity of two strings, such as Euclidean distance, Levenshtein distance, and cosine

Let English text data X be presented as $X = [\omega_1, \omega_2, \cdots, \omega_i, \cdots \omega_n]$
A neural network model M
A dataset list consist of all the word in this dataset O
for each $\omega_i \in X$ do:

$$s_i^{head} = s(\omega_1, \omega_2, \cdots, \omega_{i-1}) - s(\omega_1, \omega_2, \cdots, \omega_i)$$

$$s_i^{tail} = s(\omega_{i+1}, \omega_{i+2}, \cdots, \omega_n) - s(\omega_i, \omega_{i+1}, \cdots, \omega_n)$$

$$s_i^{without} = s(\omega_1, \omega_2, \cdots, \omega_{i-1}, \omega_{i+1}, \cdots, \omega_n) - s(\omega_1, \omega_2, \cdots, \omega_n)$$

$$s_i^{combine} = (2i/n)s_i^{head} + (2(n-i)/n)s_i^{tail} + s_i^{without}/3$$

sort w_i by $s_i^{combine}$ to get keywords list
for each w in keywords list and o in O do:
  calculating $lev_{(w.o)}(|\omega|, |o|)$
  for each w_i find out o_{w_i} by min $(lev_{w.o}(|\omega|, |o|))$
use o_{w_i} replace ω_i
$X' = [w_1, w_2, \cdots, o_{w_i}, \cdots, w_n]$ is the adversarial example of X

Algorithm 1: The visual similar word replacement algorithm.

similarity. In this paper, we choose Levenshtein distance to measure the similarity between two strings. We also tested other similarity calculation methods and finally chose Levenshtein distance because it is the most direct and fastest method. Figure 2 depicts an example which shows the calculation method of Levenshtein distance, from which we can easily see that Levenshtein distance can be used to easily calculate the similarity of two words, and the smaller the distance is, the closer the two words are.

Levenshtein distance is also known as edit distance, which refers to the minimum number of edit operations required to convert one string to another between two strings. Editing operations include replacing one character with another, inserting a character, and deleting a character.

For two strings a, b with lengths of $|a|$ and $|b|$, it is necessary to calculate the edit distance between $a(1, 2, \cdots, |a| - 1)$ and $b(1, 2, \cdots, |b|)$ and then add 1 (for the case of an increased operation, add the last one character). Or calculate the edit distance between $a(1, 2, \cdots, |a|)$ and $b(1, 2, \cdots, |b| - 1)$, and then add 1 (for the deletion operation, delete the last character). Or calculate the edit distance between $a(1, 2, \cdots, |a| - 1)$ and $b(1, 2, \cdots, |b| - 1)$, and then add 1 (for the modification operation in case, modify one character) and then take the minimum of these three as the minimum edit distance of the previous step, and so on to the first character.

Use $|a|$ and $|b|$ to represent the length of the two strings a and b, respectively; then, the Levenshtein distance between the two strings is $\text{lev}_{a,b}(|a|, |b|)$, where

$$\text{lev}_{a,b}(i,j) = \begin{cases} \max(i,j), & \text{if } \min(i,j) = 0, \\ \min \begin{cases} \text{lev}_{a,b}(i-1,j) + 1, \\ \text{lev}_{a,b}(i,j-1) + 1, \\ \text{lev}_{a,b}(i-1,j-1) + 1_{(a_i \neq b_j)}, \end{cases} & \text{otherwise}, \end{cases} \quad (6)$$

Table 1: Detailed information of Yelp review dataset.

Yelp review dataset	Train	Test
Classes	2	2
Num	400000	30000
Positive : negative	1 : 1	1 : 1
Average words	209	194

in which $1_{(a_i \neq b_j)}$ is an indicator function, when $a_i = b_j$, its value is 0; otherwise, its value is 1. $\text{lev}_{a,b}(i, j)$ represents the Levenshtein distance between the first i characters of a and the first j characters of b (i and j are subscripts starting from 1).

We use Levenshtein distance to measure the similarity between two words. The smaller the Levenshtein distance is, the more similar these two words are. So, we exchange the word chosen by the scoring module with another word whose Levenshtein distance to the keyword is smallest. In this way, the difference between the replaced text and the original text will not be very large, and it is not easy to be noticed by humans and affect human reading.

3.2.3. Generating Adversarial Examples. In the first step, we found those words with high scores, which means they are more important than the other words to the label given by deep learning models; these selected words are called "keywords." Then, we use these keywords to form a keyword list; by calculating the Levenshtein distance between the words in the keyword list and the words in the word list of the dataset, we can find out those words that are similar to the keywords in the keyword list. We only need to find out the word with the shortest Levenshtein distance to keywords. Finally, we only need to use the visual similar words found in the previous step to replace the corresponding keywords in the original text. Then, we can generate the adversarial examples that could fool deep learning models with high imperceptibility. The VSWR algorithm is described in Algorithm 1.

Table 2: Detailed information of Amazon review dataset.

Amazon review dataset	Train	Test
Classes	2	2
Num	400000	30000
Positive : negative	1 : 1	1 : 1
Average words	180	188

Table 3: Accuracy of deep learning models on the Yelp review dataset.

Model	LSTM	BiLSTM
Accuracy	95.6934%	95.647%

Table 4: Accuracy of deep learning models on the Amazon review dataset.

Model	LSTM	BiLSTM
Accuracy	88.483%	88.550%

Table 5: Accuracy changes when the num of replaced words increases (Yelp review dataset).

Num of replaced words	0	10	20	30	40	50
LSTM	0.9569	0.9296	0.8906	0.7812	0.6640	0.6171
BiLSTM	0.9564	0.8984	0.8906	0.7578	0.6562	0.5546

Table 6: Accuracy changes when the num of replaced words increases (Amazon review dataset).

Num of replaced words	0	10	20	30	40	50
LSTM	0.8848	0.7891	0.7500	0.6875	0.6406	0.6484
BiLSTM	0.8855	0.7656	0.7266	0.6953	0.6328	0.5781

4. Results and Discussion

4.1. Result

4.1.1. Dataset. We use the following two datasets: Yelp review dataset and Amazon review dataset to train two deep learning models that are designed for sentiment analysis. Then, we use our proposed algorithm to attack the pre-trained models.

The Yelp review dataset is the comment data of the Yelp website, which is extracted from the Yelp Dataset Challenge 2015. There are only two categories: positive and negative in this dataset. This dataset contains two parts. The first part is the label of the data in this dataset. The second part is the detailed comments of users on the products they bought. The entire dataset contains 560000 pieces of English texts, including 280000 positive samples and 280000 negative samples. However, many texts are only composed of few words. We delete these texts from the dataset and only remain the texts with enough words for attack. The detailed information of the dataset is shown in Table 1. As for the Amazon review dataset, it consists of reviews from Amazon, and these reviews are also divided into two categories: positive and negative. Each piece of data consists the comment text and the label. In the experiment, we also deleted redundant data from the Amazon review dataset; the detailed information is shown in Table 2.

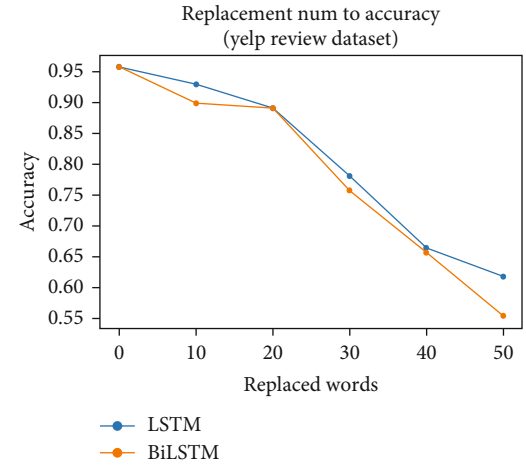

Figure 3: Accuracy changes with the number of replaced words (Yelp review dataset).

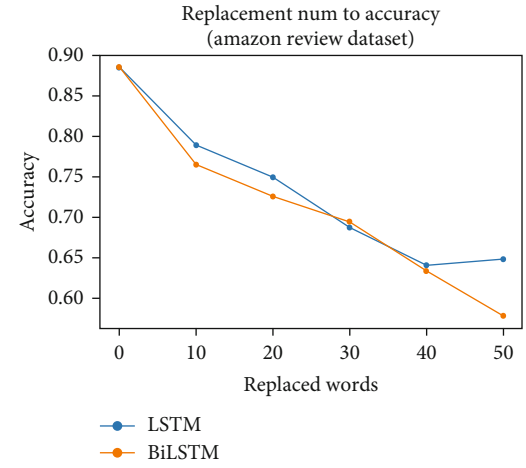

Figure 4: Accuracy changes with the number of replaced words (Amazon review dataset).

4.1.2. Models. Since we consider the context of text data when generating adversarial examples, the recurrent neural network- (RNN-) based model is the most suitable model for the experiment. To provide high accuracy, we finally choose LSTM (Long Short-Term Memory) and BiLSTM (Bidirectional Long Short-Term Memory) as the trained models to attack.

Tables 3 and 4 show the classification accuracy of these models on the Yelp review dataset and the Amazon review dataset. According to these tables, we can find that the two deep learning models could achieve good performance when solving the sentiment analysis task. Specifically, the accuracy rate of both two models can reach as high as 95% on the Yelp review dataset, while the accuracy rate of the models on the Amazon review dataset exceeds 88%. In the following

Table 7: A generated adversarial example by the VSWR algorithm (Yelp review dataset).

	Original text	Adversarial example
Text	i went here for the lunch buffet and was blown away! not the greatest location, i was a little weary at first. but, i was proven wrong because the food was out of this world. i have been to numerous thai places around the valley and this is by far the **best**!	i went here for the lunch buffet and was blown away! not the greatest location, i was a little weary at first. but, i was proven wrong because the food was out of this world. i have been to numerous thai places around the valley and this is by far the **Rest**!
Label	Positive	Negative

Table 8: A generated adversarial example by the VSWR algorithm (Amazon review dataset).

	Original text	Adversarial example
Text	while not expensive to initially purchase, this system is expensive in the long run. if you mop your floors regularly or have a lot of rooms to mop, you'll be constantly replacing pads and cleaner and batteries, none of which are cheap.additionally, this system does not clean my laminate floors well. it leaves them cloudy/ hazy and they never look or feel 100% clean afterwards. when i go over with a damp paper towel where i just mopped, it always shows a fair amount of dirt. and looking at the pad when you remove it from the mop you can see why. the way it is designed, the most pressure is on the front edge of the mop. the center and back of the pad do not receive nearly as much pressure and pick up almost nothing.in short, for small jobs and irregular cleaning, this mop will do. otherwise, it is inefficient, expensive **and** ultimately **frustrating**.	while not expensive to initially purchase, this system is expensive in the long run. if you mop your floors regularly or have a lot of rooms to mop, you'll be constantly replacing pads and cleaner and batteries, none of which are cheap.additionally, this system does not clean my laminate floors well. it leaves them cloudy/ hazy and they never look or feel 100% clean afterwards. when i go over with a damp paper towel where i just mopped, it always shows a fair amount of dirt. and looking at the pad when you remove it from the mop you can see why. the way it is designed, the most pressure is on the front edge of the mop. the center and back of the pad do not receive nearly as much pressure and pick up almost nothing.in short, for small jobs and irregular cleaning, this mop will do. otherwise, it is inefficient, expensive **andd** ultimately **frustarating**.
Label	Negative	Positive

parts, we show that the trained two deep learning models would achieve bad performance against generated adversarial examples.

4.1.3. Attack Performance. We use the method proposed in this paper to process the test dataset and then evaluate the effectiveness of each model, respectively. For the same model, as the number of words which are replaced in the original text increases, the prediction accuracy of the trained deep neural network model becomes lower and lower. As shown in Tables 5 and 6, with the number of replaced words (by its visual similar word), the models' prediction accuracy decreases.

In Figures 3 and 4, we show the change of the models' accuracy when the number of replaced words increases on two datasets. The x-axis represents the number of replaced words in the original text, and the y-axis denotes the accuracy of the deep learning models. From the figures, the original accuracy of both models is higher than 95% and 88%, respectively, when no word is replaced. When we replace more words as the x-axis, the accuracy of both models decreases as the two curves in the figures.

In Tables 7 and 8, we show some generated adversarial texts on the two datasets by the VSWR algorithm. In Table 7, an original text from the Yelp review dataset is recognized as "positive" by the BiLSTM model. However, as the algorithm only changes "best" to "Rest," the model classifies the generated text as "negative." Similarly, an original text from the Amazon view dataset is recognized as "negative"; by changing two words to their visual similar words, the generated text is classified as "positive." Clearly, the trained models would achieve bad performance against the generated adversarial examples, which implies the effectiveness of our proposed method.

5. Discussion

From Figures 3 and 4, we can find that the BiLSTM model is more susceptible to the influence of adversarial examples compared with the LSTM model. This is because the BiLSTM model fully considers the relationship between the scored words and the context. In addition, the extracted keywords by our method for the replacement in generating adversarial examples are more suitable for the BiLSTM model. During the preprocessing step of the dataset, we filter out the texts that are composed with only a small number of words; this is because humans could easily recognize short texts that are modified. Hence, we select relatively long texts in the dataset for the attack experiment. During our experiments, when the number of modified words is small, the attack effect on the two models is not good, but when we increase the number of modified words to 25% of the original text (on Yelp review dataset), the classification accuracy of the model can be reduced from 0.95 to 0.55. This result was also confirmed on the Amazon review dataset; the change of the words can reduce the model accuracy from 0.88 to 0.57.

Actually, some existing adversarial attacks could largely reduce the prediction accuracy of the neural network models. However, some of them change the characters in a

word or split a word by some special symbols; the generated adversarial texts can be easily noticed by humans since some generated words do not exist in the vocabulary. In our paper, we select words that look similar to the original one for replacement, which could successfully fool both humans and deep neural network models. Although we need to modify 50 words to achieve good attack performance for the dataset, the modified words look quite similar to the original one and only 25% of words are modified on average, which is also acceptable.

However, there are also some questions that can be concluded by the examples above. For example, those words with strong emotions such as "interesting" and "bad" have not been changed during the algorithm, which means the trained deep learning models do not mainly rely on these words for classifying.

In this paper, we only verify the security vulnerabilities of deep learning models by the adversarial attack methods. Indeed, there are also many other methods to show security vulnerabilities. For example, we can modify the training data such that the trained deep learning model cannot study the correct data distribution; this kind of attack is also called data poisoning. In addition, some methods are proposed to steal privacy of deep learning models, such as inferring data from the training set, stealing parameters of the models. These methods would cause privacy leakage of deep learning models.

6. Conclusions

In this paper, we explore the security vulnerabilities of deep learning models by adversarial attacks. Specifically, we propose the visual similar word replacement method to attack several deep learning. This method firstly sorts the importance of the words in the original dataset and selects the words that have the greatest influence on the classification result as the keywords. At the same time, the original data is processed to obtain a word list containing all the words in the original dataset, and then, we use the word found in the word list whose Levenshtein distance between keywords is 1 to replace the keyword. The replaced text is the generated adversarial example of the original text. We also conducted experiments on the sentiment analysis datasets, and the results proved that the adversarial examples generated by this method could successfully attack the deep learning models such that they would make misclassification. In addition, as the number of modified words increases, the impact on the neural network model becomes more and more significant.

In the future, we will try to extend this method to attack more classification models for other textual analysis tasks, such as text generation, spam filtering, and machine translation. At the same time, we also try to improve the proposed VSWR method such that less words could be selected for replacement.

Data Availability

The Yelp review dataset is the comment data of the Yelp website. In our work, we filter out the texts in the dataset that contain words less than 50, since the changes of short texts are easier to be noticed by humans. Readers who are interested can get the dataset processed in https://drive.google.com/drive/folders/1AWhczX50NVyr-gciAIz96Vtb-WVWS9L5D?usp=sharing.

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

Acknowledgments

This work is supported in part by the National Key R&D Program of China 2019YFB1706003, the Guangdong Key R&D Program of China 2019B010136003, and the National Natural Science Foundation of China under Grant Nos. 61972106 and 61902082.

References

[1] 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," 2019, https://arxiv.org/abs/1907.11932.

[2] Z. Gu, L. Wang, X. Chen et al., "Epidemic risk assessment by a novel communication station based method," *IEEE Transactions on Network Science and Engineering*, 2021.

[3] L. Wang, J. Niu, and S. Yu, "SentiDiff: combining textual information and sentiment diffusion patterns for Twitter sentiment analysis," *IEEE Transactions on Knowledge and Data Engineering*, vol. 32, no. 10, pp. 2026–2039, 2020.

[4] Z. Gu, W. Hu, C. Zhang, H. Lu, L. Yin, and L. Wang, "Gradient shielding: towards understanding vulnerability of deep neural networks," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 2, pp. 921–932, 2021.

[5] Z. Gu, Y. Su, C. Liu et al., "Adversarial attacks on license plate recognition systems," *Computers, Materials & Continua*, vol. 65, no. 2, pp. 1437–1452, 2020.

[6] W. Zou, S. Huang, J. Xie, X. Dai, and J. Chen, "A reinforced generation of adversarial examples for neural machine translation," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2019, http://arxiv.org/abs/1911.03677.

[7] M. Sundermeyer, R. Schlüter, and H. Ney, "LSTM neural networks for language modeling," in *Thirteenth annual conference of the international speech communication association*, 2012.

[8] Z. Gu, Y. Cai, S. Wang et al., "Adversarial attacks on content-based filtering journal recommender systems," *Computers, Materials & Continua*, vol. 64, no. 3, pp. 1755–1770, 2020.

[9] H. Hossein, S. Kannan, B. Zhang, and R. Poovendran, "Deceiving Google's perspective API built for detecting toxic comments," 2017, https://arxiv.org/abs/1702.08138.

[10] P. Yang, J. Chen, C.-J. Hsieh, J. L. Wang, and M. I. Jordan, "Greedy attack and Gumbel attack: generating adversarial examples for discrete data," *Journal of Machine Learning Research*, vol. 21, pp. 1–36, 2020.

[11] J. Ebrahimi, A. Rao, D. Lowd, and D. Dou, "Hotflip: white-box adversarial examples for text classification," 2017, https://arxiv.org/abs/1712.06751.

[12] Z. Gong, W. Wang, B. Li, D. Song, and W. S. Ku, "Adversarial texts with gradient methods," 2018, https://arxiv.org/abs/1801.07175.

[13] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2014, https://arxiv.org/abs/1412.6572.

[14] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 372–387, Saarbruecken, Germany, 2016.

[15] R. Jia and P. Liang, "Adversarial examples for evaluating reading comprehension systems," 2017, https://arxiv.org/abs/1707.07328.

[16] H. Zhang, H. Zhou, N. Miao, and L. Li, "Generating fluent adversarial examples for natural languages," 2020, https://arxiv.org/abs/2007.06174.

[17] J. Gao, J. Lanchantin, M. L. Soffa, and Y. Qi, "Black-box generation of adversarial text sequences to evade deep learning classifiers," in *2018 IEEE Security and Privacy Workshops (SPW)*, pp. 50–56, San Francisco, CA, USA, 2018.

[18] D. Jin, Z. Jin, J. T. Zhou, and P. Szolovits, "TextFool: fool your model with natural adversarial text," 2019, http://groups.csail.mit.edu/medg/ftp/psz-papers/2019%20Di%20Jin.pdf.