International Journal of Computational Intelligence Systems
Vol. **14(1)**, 2021, *pp.* 1934–1947
DOI: https://doi.org/10.2991/ijcis.d.210624.001; ISSN: 1875-6891; eISSN: 1875-6883
https://www.atlantis-press.com/journals/ijcis/

Research Article

A Multimodal Adversarial Attack Framework Based on Local and Random Search Algorithms

Zibo Yi*, Jie Yu, Yusong Tan, Qingbo Wu

College of Computer, National University of Defense Technology, No. 109, Deya Road, Kaifu District Changsha, Hunan Province, China

ARTICLE INFO

Article History
Received 29 Jul 2020
Accepted 22 Jun 2021

Keywords
Adversarial attack
Multimodal applications
Adversarial image
Adversarial text
Local search
Random search

ABSTRACT

Although many problems in computer vision and natural language processing have made breakthrough progress with neural networks, adversarial attack is a serious potential problem in many neural network- based applications. Attackers can mislead classifiers with slightly perturbed examples, which are called adversarial examples. As the existing adversarial attacks are specific to application and have difficulty in general usage, we propose a multimodal adversarial attack framework to attack both text and image classifiers. The proposed framework firstly generates candidate set to find the substitution words or pixels and generate candidate adversarial examples. Secondly, the framework updates candidate set and search adversarial examples with three local or random search methods [beam search, genetic algorithm (GA) search, particle swarm optimization (PSO) search]. The experiments demonstrate that the proposed framework effectively generates image and text adversarial examples. Comparing the proposed methods with other image adversarial attacks in MNIST dataset, the PSO search in the framework has 98.4% attack success rate which outperforms other methods. Besides, the beam search has the best attack efficiency and human imperception in both MNIST and CIFAR-10 dataset. Comparing with other text adversarial attacks, the beam search in the framework has an attack success rate of 91.5%, which outperforms other existing and the proposed search methods. In attack efficiency, the beam search also outperforms other methods, meaning that we can craft text adversarial examples with less perturbation using beam search.

1. INTRODUCTION

With the development of neural network-based deep learning, many problems in computer vision (CV) and natural language processing (NLP) have made breakthrough progress. However, adversarial attack [1] can mislead neural network based applications in CV and NLP domain. For example, by wearing specific eyeglasses, an attacker will evade being recognized or impersonate another individual when he attacks face recognition system [2]. An example of text adversarial attack is that a tiny modification of inappropriate content such as online harassment can deceive Google's toxic comments detector [3]. Adversarial attack is a hot topic in CV domain [4–8]. In order to prove that text is also vulnerable to adversarial attack, several researchers propose text adversarial attacks [8–11] against natural language applications. Although there are many adversarial attacks proposed to reveal the vulnerability of image and text neural network classifier, how to implement an adversarial attack that can generate both text and image adversarial examples still remains a hard problem.

The biggest difference between image and text adversarial attack is that text is discrete data while image can be processed as continuous data. The pixel values can be calculated as continuous data and be clipped to be integers between 0 and 255 when generating adversarial images. Although words can be embedded into a continuous space, they are still discrete points in the space. The gradient methods used in image adversarial attack are no longer valid in attacking text classifier. There are several methods for crafting adversarial text. Gradient descent and nearest point searching method [12], linguistic synonym replacement [13,14], global and local searching for substitute words [15,16] are typical methods to generate adversarial text. However, these methods have difficulty in general usage. For instance, linguistic substitution method is specific to application. Besides, the attacks against text classifier cannot directly attack image classifier, and vice versa.

In this paper, a multimodal adversarial attack framework is proposed to generate both text and image adversarial examples. With this framework, text and image are formalized into a unified data type thus the attacks in the framework can be both applied to text and image classifiers. There are two main challenges to construct such a multimodal adversarial attack framework.

One challenge is how to unify text and image data so that the attack can be formalized to a solvable problem. In the proposed framework, both text and image data can be represented as tensor, i.e., a multi-dimensional array. Each word in a text is represented as a word vector, so text is a (text length, word vector dimension) shape

*Corresponding author. Email: yizibo14@nudt.edu.cn

tensor. Similarly, image is a (image height, image width, color channel) shape tensor. After unifying image and text data, we introduce a distance function d to measure the difference between the adversarial and original example. Then, in our unified framework, the adversarial attack is formalized as finding a tensor that satisfies (1) can mislead the classifier and (2) d is as small as possible.

The other challenge is that the search space is huge so that we need some techniques to improve the search efficiency. We combine a process named candidate set generation (CSG) and local/random search methods to solve the huge search space problem. CSG generates a candidate set which consists of limited number of tensors in each searching iteration. The candidate set aims to contain an adversarial example to mislead classifier. The candidate set is updated with several variants of search methods if it doesn't contains an adversarial example.

The workflow of the proposed multimodal adversarial attack framework are as follows. First, the text and image data are preprocessed to be tensors. Then, in the CSG the saliency map is proposed to measure the influence of pixels or words on the classification result. The saliency map is to determine the modification priority of each pixel or word. According to the modification priority, several pixels or words are modified and added to the candidate set. If the candidate set contains an adversarial example, the adversarial attack problem is solved. Otherwise, a variety of local/random search methods, such as beam search, genetic algorithm (GA) and particle swarm optimization (PSO), are applied to update the candidate set until it contains an adversarial example. By using this framework, multimodal adversarial examples can be crafted successfully. In summary, the contributions of this paper are as follows:

- We propose a multimodal adversarial attack framework which can generate both image and text adversarial examples. It formalizes the image and text data as tensor and then searches adversarial examples in the space of tensor.
- We propose saliency map to measure the modification priority of each pixel or word. Using saliency map, the CSG process is proposed to generate images or texts that are most likely to be adversarial examples.
- We combine CSG and 3 local/random search methods to solve the huge search space problem. The 3 search methods are variants of the standard beam search, GA and PSO.

2. RELATED WORK

CV and natural language applications are vulnerable to adversarial attacks. Both image and text classifiers can be misled by tiny perturbed examples. The image adversarial attacks are proposed earlier than text adversarial attacks. Neural network classifiers usually use gradient descent to decrease a loss function in order to improve its generalization performance. Thus the attackers use gradient ascent to mislead the classifier and propose several gradient-based adversarial attacks. LBGFS [1], fast gradient sign method (FGSM) [17], basic iterative method (BIM) [18] are typical gradient-based image adversarial attacks.

After that, more sophisticated adversarial attacks are proposed to solve specific problems. Papernot *et al.* [8] propose transferability to attack the black-box image classifier. Sharif *et al.* [2] propose a specially made eyeglasses to attack face recognition system in real life. In 2016, Papernot *et al.* [5] propose Jacobian-based Saliency Map Attack (JSMA), in which saliency map measures the significance of each pixel's impact on classification results. Continuously selecting the most significant pixel to modify during the attack improves the attack efficiency and dramatically reduces the amount of pixels that need to be modified. A more extreme attack is one pixel attack [7], which only modifies one pixel to make the classifier go wrong. In 2017, Carlini and Wagner [19] propose a set of attacks (C&W Attacks) that regard the attack as a constrained optimization problem and achieve good results against defense methods based on gradient smoothing such as knowledge distillation [20].

The text adversarial attack is also widely studied in the literature. There are two categories of text adversarial attack: character-level attack and word-level attack. Character-level attack perturbs the text by modifying character. Hotflip [10] is for the character RNN classifier. It is also a gradient-based white box text adversarial attack method. Gao *et al.* [21] propose simple character-level transformations to change the original classification of black-box text classifier with minimal edit distance of the perturbation. Hosseini *et al.* [3] propose several effective attempts to attack Google's Perspective API.

Word-level attack perturbs the text by modifying word. Liang *et al.* [14] point out the difficulty of text adversarial attack and propose three strategies including inserting, deleting, and modifying to perturb text. These strategies regard the classifier as a white box, calculate the significance of each word's impact, and then change the corresponding position of the text. Samanta and Mehta [9] use an adversarial attack against the IMDB [22] Film Review Classifier and the Twitter Gender Classifier to make a serious error in inferring the results. Papernot *et al.* [12] propose that forward derivation can be used for RNN. They propose the RNN version of FGSM. Ren *et al.* [23] propose probability weighted word saliency (PWWS) for text adversarial attack. The change of the classification probability is used as the weight of the original word saliency (WS). This strategy increase attack success rate and reduce the word substitution rate. TextBugger [13] uses the method of finding important words and then replace it. There are also methods using local heuristic search to generate adversarial examples. Alzantot *et al.* [24] propose the application of GAs to find adversarial text. Kuleshov *et al.* [16] propose using local greedy search to attack spam detectors and fake news detectors. Jin *et al.* [25] show that the pretrained BERT [26] which has the best language model performance is also vulnerable to adversarial attacks. Other methods such as graph model [27], game theory [28], and KL divergence [29] can all be used to construct adversarial text.

To the best of our knowledge, our framework is the first multimodal adversarial attack framework that can generate both adversarial images and adversarial texts. In the proposed framework, we use a unified formalization for text and image examples and 3 local/random search algorithms for searching adversarial examples. Compared to other adversarial attacks, the advantages of the search method in the proposed framework are as follows. In each iteration, the amount of perturbations are small (especially beam search) and the perturbations are toward the direction of label's probability maximum decreasing, resulting in more efficient attacks compared to other methods, i.e., achieving higher attack success rates by using

smaller perturbations. Beam search perturbs examples slightly and generates adversarial examples in the neighborhood of the original examples in each iteration. Thus, the generated adversarial examples are similar to the original example. It is difficult for humans to perceive the perturbations in the adversarial example. The disadvantage is that the GA and PSO in the framework may generate more noise in the examples due to the gene mutation and particle movement. However, the random search algorithms can find the adversarial examples faster than the beam search.

3. METHOD

3.1. Adversarial Attack Problem

Both text and image data can be represented as tensor. An image consists of *height* × *width* pixels. Each pixel has several *channels*. For example, the grayscale image has a single channel, while the color image contains RGB 3 channels. So image data is *height* × *width* × *channel* shaped tensor. A sentence consists of *length* words. Each word can be represented by a word vector embedded in a space with specific *dimension*. So text data is *length* × *dimension* shaped tensor. Let tensor x denote an image or text tensor, $c(\cdot)$ denote the output of classifier c. Adversarial attack can be formalized as a constrained optimization problem:

$$\underset{x'}{\operatorname{argmin}}\, d\left(x, x'\right) \ s.t.\ c\left(x'\right) \neq c(x) \tag{1}$$

d denotes the distance between the example before and after being modified. So d stands for the perturbation amount. Adversarial attack is to find a x' close to x, such that $c\left(x'\right) \neq c(x)$.

There are several difficulties in solving the constrained optimization problem (1). First, it cannot be solved with gradient descent based methods. Although words are embedded in a continuous space, the words are still discrete points in this space. So the gradient descent of word vector cannot always find the corresponding word. Second, the search space of x is extremely large. In order to solve these two problems, we consider selecting some examples near x and putting them in a candidate set, and then updating candidate set with several search method, which let the probability of original label to be as small as possible. The multimodal adversarial attack framework is described in the following subsections. The framework mainly solve two problems: (1) how to generate a candidate set and (2) how to search an adversarial example.

3.2. Multimodal Adversarial Attack Framework

The workflow of the framework is shown in Figure 1. Two most important parts of this framework are CSG and update. During the CSG, M words or pixels that have the greatest impact on classification results are selected. For each word or pixel, N substitute words or pixels are selected. Therefore, $M \times N$ examples can be generated in each CSG process.

After CSG process, $M \times N$ examples are generated. Then several local/random search strategies update the candidate set and search the examples that can mislead classifier. These search strategies include beam search, GAs, and PSO. These three strategies regard the candidate set as beam, population, and particle swarm and update them in different ways. The size of the candidate set is fixed and does not change before and after the update. While updating the candidate set, the examples in the set are checked to see if there is any adversarial example with limited perturbation.

3.3. Candidate Set Generation

The CSG process includes two parts: selecting influential pixel or word and selecting substitute pixel or word. First it is necessary to measure the influence of pixels or words on the classification results. We propose the saliency map $S\left(\boldsymbol{x}_i, y\right)$ to measure a pixel or word's influence. In an image or text tensor $\boldsymbol{x} = \left(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_i, \ldots, \boldsymbol{x}_n\right)$, we modify $\boldsymbol{x}_i$ along the gradient of probability, then the tensor becomes $\boldsymbol{x}' = \left(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_i - \varepsilon \cdot \frac{\partial P(y|\boldsymbol{x},\theta)}{\partial \boldsymbol{x}_i}, \ldots, \boldsymbol{x}_n\right)$. Let $P(y \mid \boldsymbol{x}, \theta)$ denote the probability of classifier with parameters θ classifies $\boldsymbol{x}$ as y. The change of probability, which is denoted by ΔP_i, can be used to measure the influence of $\boldsymbol{x}_i$ on y.

$$\Delta P_i = P(y \mid \boldsymbol{x}, \theta) - P\left(y \mid \boldsymbol{x}', \theta\right) \tag{2}$$

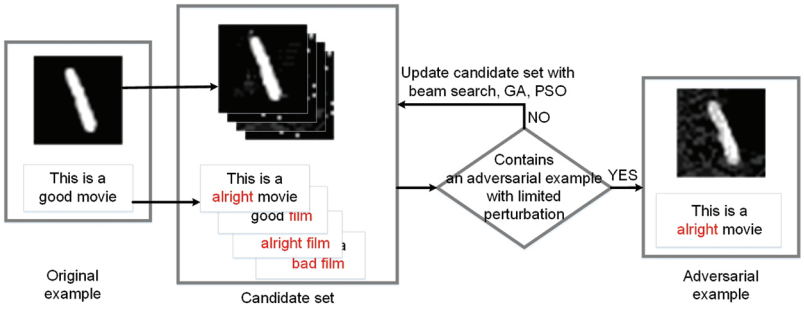


Figure 1 | Workflow of the multimodal adversarial attack framework. There are two mainly parts in the framework: candidate set generation (CSG) and its update. The CSG process finds several examples which is near the original example to consist a candidate set. Then several search strategies are utilized to update the candidate set and look for adversarial examples. The candidate set is checked to see whether it contains an adversarial example with limited perturbation.

ΔP_i can only reflect the change of label y's probability when slightly modifying $\boldsymbol{x}_i$. An attacker usually hopes that the probability of another label $\hat{y}$ increase significantly. We denote other label's maximum probability change with $\Delta\widehat{P_i}$.

$$\Delta\hat{P}_i = \max_{\hat{y}\neq y}\left[P\left(\hat{y} \mid \boldsymbol{x}', \theta\right) - P(\hat{y} \mid \boldsymbol{x}, \theta)\right] \tag{3}$$

If a pixel or word's slightly modification cause the maximum probability changes (ΔP_i and $\Delta\hat{P}_i$), the pixel or word is the most influential one. Thus the saliency map is calculated as follows:

$$S\left(\boldsymbol{x}_i, y\right) = \Delta P_i \cdot \Delta\hat{P}_i \tag{4}$$

Note that $\boldsymbol{x}'$ is calculated by $\boldsymbol{x}$ moving along the opposite direction of the gradient of $\boldsymbol{x}_i$, so $\Delta P_i > 0$. Since the sum of probabilities is 1 and $\boldsymbol{x}'$ makes the probability of label y lower, it must make the probability of some other label greater, i.e., $\max_{\hat{y}\neq y}\left[P\left(\hat{y} \mid \boldsymbol{x}', \theta\right) - P(\hat{y} \mid \boldsymbol{x}, \theta)\right] > 0$. Therefore, S will have a greater value in the following case: the probability of the original label has a greater decrease and the probability of another label has a greater increase.

Second, it is necessary to find substitute pixel or word. For an influential pixel or word $\boldsymbol{x}_i$, the priority of the substitute pixel or word $\boldsymbol{x}_i'$ is

$$\rho_{\boldsymbol{x}_i'} = 1/\|\boldsymbol{x}_i - \varepsilon \cdot \nabla_{\boldsymbol{x}_i} P(y \mid \boldsymbol{x}, \theta) - \boldsymbol{x}_i'\|_2 \tag{5}$$

The closest point of $\boldsymbol{x}_i - \varepsilon \cdot \nabla_{\boldsymbol{x}_i} P(y \mid \boldsymbol{x}, \theta)$ has the highest priority, for the reason that $\boldsymbol{x}_i - \varepsilon.\nabla_{\boldsymbol{x}_i} P(y \mid \boldsymbol{x}, \theta)$ is the gradient descent result of $\boldsymbol{x}_i$ and it makes $P(y \mid \boldsymbol{x}, \theta)$ decrease most. If $\boldsymbol{x}_i$ is a pixel, itis possible to get noninteger values or values outside the range (0,255) when we calculate the pixel value in Eq. (5). In our implementation, to get the correct pixel value, we set those values greater than 255 to be 255, those values less than 0 are set to 0. Then we take the integer part of the floating numbers as the final pixel values. If $\boldsymbol{x}_i$ is a word, the NLP toolkit gensim[1] is used to obtain several word vectors that are closest to the given vector. The CSG is described with Algorithm 1. The saliency map $S\left(\boldsymbol{x}_i, y\right)$ is sorted (Algorithm 1, line 2). Then M pixels or words with largest saliency are selected (Algorithm 1, line 3). For each pixel or word, N substitute pixels or words that make the largest ρ are selected (Algorithm 1, line 5). At last, the candidate set contains $M \times N$ examples.

3.4. Candidate Set Update

From the previous subsection we can know that the CSG process generates $M \times N$ examples. In order to search adversarial example, an intuitive idea is to use global search. The global search tree has $M \times N$ branches. Thus the computational complexity of global search is $O\left((M \times N)^L\right)$ when the search tree has L layers. In practice, the global search method is impossible to complete the task of searching adversarial example because of its huge complexity. Therefore, in this framework local/random search methods are proposed to update candidate set and search adversarial examples. The search methods include beam search, GA, PSO.

[1]https://radimrehurek.com/gensim

Algorithm 1: Candidate set generation

Input: The classifier to be attacked, c. The example before perturbed, $(\boldsymbol{x}, y)$. The number of pixels or words in $\boldsymbol{x}$, T. The saliency map, S. The number of selected words or pixels, M. The number of substitutions for each selected word or pixel, N.

Output: Candidate set, Ω.

1 $\Omega \leftarrow \varnothing$
2 $I \leftarrow argsort([S(\boldsymbol{x}_1, y), ..., S(\boldsymbol{x}_T, y)])$ ▷ Get the indexes sorted by S.
3 $I \leftarrow reversed(I)[0:M]$ ▷ Reverse the indexes and get M indexes with greatest S.
4 **foreach** *i in I* **do**
5 $\quad \{\boldsymbol{x}_i^{(1)}, \boldsymbol{x}_i^{(2)}, ..., \boldsymbol{x}_i^{(N)}\} \leftarrow$ top N of $\rho_{\boldsymbol{x}_i'}$
6 $\quad$ **foreach** *j in* $[1, 2, ..., N]$ **do**
7 $\quad\quad \Omega \leftarrow \Omega \cup \{[\boldsymbol{x}_1, ..., \boldsymbol{x}_i^{(j)}, ..., \boldsymbol{x}_T]\}$
8 **return** Ω

3.4.1. Search adversarial example with beam search

Completely searching all the generated examples is impossible in practice. So the beam search is utilized to constrain the search branches. The beam search select a certain number of examples for the next iteration. The examples with smallest $P\left(y \mid \boldsymbol{x}', \theta\right)$ are selected. With beam search, the computational complexity dramatically decrease because only a certain number of examples in each layer of search tree.

Algorithm 2 describes how to find adversarial example with beam search. In each iteration, every example $\boldsymbol{x}'$ in candidate set can generate $M \times N$ examples (Algorithm 2, line 10). Thus the candidate set will contain $(M \times N)^2$ examples. Inthe beginning of next iteration, we can select $M \times N\boldsymbol{x}'$ with smallest $P\left(y \mid \boldsymbol{x}', \theta\right)$ from the candidate set (Algorithm 2, line 4). Then the search space is fixed to $M \times N$. With this method, the adversarial example can be found within several iterations.

3.4.2. Search adversarial example with GA

In GA, the candidate set is regarded as a biological population. Examples in the candidate set are individuals. GA aims at making the biological population evolve toward a better optimization objective. To search adversarial examples, the individuals in the population should evolve into individuals with smaller $P\left(y \mid \boldsymbol{x}', \theta\right)$. The crossover and mutation process are introduced to maintain the gene's stability and diversity.

In the crossover process, the examples with greater $1/P\left(y \mid \boldsymbol{x}', \theta\right)$ are more likely to reproduce the next generation (in Algorithm 3, line 6, 7, $\boldsymbol{x}_A$ and $\boldsymbol{x}_B$ are selected according to $1/P\left(y \mid \boldsymbol{x}', \theta\right)$'s distribution). To reproduce the next generation, images or texts are both

Algorithm 2: Search adversarial example with beam search

Input: The classifier to be attacked, c. The example before perturbed, $(\boldsymbol{x}, y)$. The limited perturbation amount, Υ.
Output: Adversarial example, $\boldsymbol{x}^*$.

1 $\Omega \leftarrow CSG(\boldsymbol{x}, M, N)$
2 $\boldsymbol{x}^* \leftarrow \boldsymbol{x}$
3 **while** $c(\boldsymbol{x}^*) = y$ ***and*** $d(\boldsymbol{x}^*, \boldsymbol{x}) < \Upsilon$ **do**
4 $\quad \Theta \leftarrow$ select $M \times N$ $\boldsymbol{x}'$ with smallest $P(y|\boldsymbol{x}', \theta)$ from Ω
5 $\quad \Omega \leftarrow \varnothing$
6 $\quad$ **foreach** $\boldsymbol{x}'$ ***in*** Θ **do**
7 $\quad\quad$ **if** $c(\boldsymbol{x}') \neq y$ **then**
8 $\quad\quad\quad \boldsymbol{x}^* \leftarrow \boldsymbol{x}'$
9 $\quad\quad\quad$ **break**
10 $\quad\quad \Omega \leftarrow \Omega \cup CSG(\boldsymbol{x}', M, N)$
11 **return** $\boldsymbol{x}^*$

Algorithm 3: Search adversarial example with genetic algorithm

Input: The classifier to be attacked, c. The example before perturbed, $(\boldsymbol{x}, y)$. The limited perturbation amount, Υ.
Output: Adversarial example, $\boldsymbol{x}^*$.

1 $\Omega \leftarrow CSG(\boldsymbol{x}, M, N)$
2 $\boldsymbol{x}^* \leftarrow \boldsymbol{x}'$ which has smallest $P(y|\boldsymbol{x}', \theta)$ where $\boldsymbol{x}' \in \Omega$
3 **while** $c(\boldsymbol{x}^*) = y$ ***and*** $d(\boldsymbol{x}^*, \boldsymbol{x}) < \Upsilon$ **do**
4 $\quad \mathbb{P} \leftarrow [\frac{1}{P(y|\boldsymbol{x}^{(1)}, \theta)}, \ldots, \frac{1}{P(y|\boldsymbol{x}^{(M\times N)}, \theta)}]$ where $\boldsymbol{x}^{(i)} \in \Omega$
5 $\quad$ Normalize $\mathbb{P}$
6 $\quad \boldsymbol{x}_A \leftarrow$ random select $\boldsymbol{x}' \backsim \mathbb{P}$ from Ω
7 $\quad \boldsymbol{x}_B \leftarrow$ random select $\boldsymbol{x}' \backsim \mathbb{P}$ from Ω
8 $\quad \boldsymbol{x}_C, \boldsymbol{x}_D \leftarrow Crossover(\boldsymbol{x}_A, \boldsymbol{x}_B)$
9 $\quad \widehat{\boldsymbol{x}_C} \leftarrow Mutation(\boldsymbol{x}_C), \widehat{\boldsymbol{x}_D} \leftarrow Mutation(\boldsymbol{x}_D)$
10 $\quad \Omega \leftarrow \Omega \cup \{\boldsymbol{x}_C, \widehat{\boldsymbol{x}_C}, \boldsymbol{x}_D, \widehat{\boldsymbol{x}_D}\}$
11 $\quad$ Remove 4 examples with the greatest $P(y|\boldsymbol{x}', \theta)$ from Ω
12 $\quad \boldsymbol{x}^* \leftarrow \boldsymbol{x}'$ which has smallest $P(y|\boldsymbol{x}', \theta)$ where $\boldsymbol{x}' \in \Omega$
13 **return** $\boldsymbol{x}^*$

segmented into two parts and then reassembled to be new examples. The crossover of GA is illustrated in Figure 2. The generated individuals $\boldsymbol{x}_C, \boldsymbol{x}_D$ mutate into two new individuals $\widehat{\boldsymbol{x}_C}$, $\widehat{\boldsymbol{x}_D}$ (Algorithm 3, line 9). In mutation process, for example, modifying one pixel or one word of $\boldsymbol{x}_C$ can generate a set $CSG(\boldsymbol{x}_C, M, N)$. Then we can select $\widehat{\boldsymbol{x}_C}$ which has the smallest $P(y \mid \widehat{\boldsymbol{x}_C}, \theta)$ in this set as the mutation result. In each generation, crossover and mutation process can generate 4 examples, thus 4 examples with the greatest $P(y \mid \boldsymbol{x}', \theta)$ are removed (Algorithm 3, line 11). In the end of each iteration, the example with best attack performance is selected (Algorithm 3, line 12) to be checked whether it can attack classifier successfully.

3.4.3. Search adversarial example with PSO

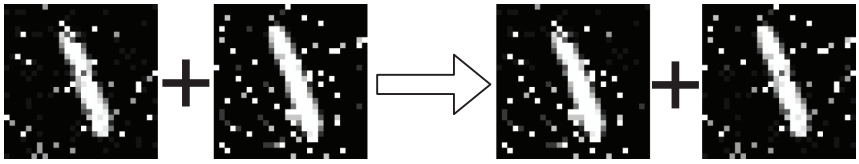
PSO regard the examples as particles. The particles have positions in a high- dimensional space. The positions are represented as tensors. In this search method, we make the particles move towards positions which have smaller $P(y \mid \boldsymbol{x}', \theta)$.

At the beginning of each iteration, $\boldsymbol{x}_g^{(i)}$ and $\boldsymbol{x}^*$ need to be updated (Algorithm 4, line 5–9). $\boldsymbol{x}_g^{(i)}$ is the best position of $\boldsymbol{x}^{(i)}$. $\boldsymbol{x}^*$ is the best position all the particles have ever reached. All the particles need to update their positions. There are $M \times N$ particles in the swarm. Each particle has the momentum of maintaining its own velocity, moving toward $\boldsymbol{x}_g^{(i)}$ and moving toward $\boldsymbol{x}^*$. The updated velocity of particles are calculated with these 3 momentums (Algorithm 4, line 13). $\mathbb{T}$ denotes the tensor representation of examples. We move $\boldsymbol{x}^{(i)}$ along

Algorithm 4: Search adversarial example with particle swarm optimization

Input: The classifier to be attacked, c. The example before perturbed, $(\boldsymbol{x}, y)$. The limited perturbation amount, Υ. The hyper-parameters of PSO, w, ϕ_p, ϕ_g.
Output: Adversarial example, $\boldsymbol{x}^*$.

1 $\Omega \leftarrow CSG(\boldsymbol{x}, M, N)$
2 $\boldsymbol{x}_g^{(i)} \leftarrow \boldsymbol{x}^{(i)}$ **for each** $\boldsymbol{x}^{(i)} \in \Omega$
3 $\boldsymbol{x}^* \leftarrow$ select $\boldsymbol{x}^{(i)}$ with smallest $P(y|\boldsymbol{x}^{(i)}, \theta)$ from $\{\boldsymbol{x}_g^{(1)}, \ldots, \boldsymbol{x}_g^{(M\times N)}\}$
4 **while** $c(\boldsymbol{x}^*) = y$ ***and*** $d(\boldsymbol{x}^*, \boldsymbol{x}) < \Upsilon$ **do**
5 $\quad$ **foreach** i ***in*** $[1, 2, \ldots, M \times N]$ **do**
6 $\quad\quad$ **if** $P(y|\boldsymbol{x}^{(i)}, \theta) < P(y|\boldsymbol{x}_g^{(i)}, \theta)$ **then**
7 $\quad\quad\quad \boldsymbol{x}_g^{(i)} \leftarrow \boldsymbol{x}^{(i)}$
8 $\quad\quad$ **if** $P(y|\boldsymbol{x}^{(i)}, \theta) < P(y|\boldsymbol{x}^*, \theta)$ **then**
9 $\quad\quad\quad \boldsymbol{x}^* \leftarrow \boldsymbol{x}^{(i)}$
10 $\quad \Theta \leftarrow CSG(\boldsymbol{x}^{(1)}, M, N) \cup \ldots \cup CSG(\boldsymbol{x}^{(M\times N)}, M, N)$ where $\boldsymbol{x}^{(i)} \in \Omega$
11 $\quad$ **foreach** i ***in*** $[1, 2, \ldots, M \times N]$ **do**
12 $\quad\quad r_p, r_g \leftarrow$ random float number in (0, 1)
13 $\quad\quad \boldsymbol{v}^{(i)} \leftarrow w\boldsymbol{v}^{(i)} + \phi_p r_p(\mathbb{T}_{\boldsymbol{x}_g^{(i)}} - \mathbb{T}_{\boldsymbol{x}^{(i)}}) + \phi_g r_g(\mathbb{T}_{\boldsymbol{x}^*} - \mathbb{T}_{\boldsymbol{x}^{(i)}})$
14 $\quad\quad \boldsymbol{x}^{(i)} \leftarrow$ select $\boldsymbol{x}'$ with smallest $||\mathbb{T}_{\boldsymbol{x}^{(i)}} + \boldsymbol{v}^{(i)} - \boldsymbol{x}'||_2$ from Θ
15 $\quad$ Update Ω with new $\boldsymbol{x}^{(i)}$
16 **return** $\boldsymbol{x}^*$



(a) The crossover of images

I don't care if some citizens voted this movie to be rotten . If you want the truth this is a very alright movie ! It has every thing a movie should have , you really should get this one .

+

I don't care if some people voted this movie to be bad . If you want the truth this is a very good movie ! It has every thing a movie should have , you really should get this paradis .

⇩

I don't care if some citizens voted this movie to be rotten . If you want the truth this is a very good movie ! It has every thing a movie should have , you really should get this paradis .

+

I don't care if some people voted this movie to be bad . If you want the truth this is a very alright movie ! It has every thing a movie should have , you really should get this one .

(b) The crossover of texts

Figure 2 The crossover of genetic algorithm. To reproduce the next generation from two examples, we propose the crossover process of images and texts. Images or texts are both segmented into two parts and then reassembled to be new examples. For example, the two sentences in the candidate set are truncated from the middle, and then the first half and the second half of the different sentences are assembled together.

the velocity $\boldsymbol{v}^{(i)}$, thus $\boldsymbol{x}^{(i)}$ has a new position $\mathbb{T}_{\boldsymbol{x}^{(i)}} + \boldsymbol{v}^{(i)}$. We select $\boldsymbol{x}'$ which is closest to $\mathbb{T}_{\boldsymbol{x}^{(i)}} + \boldsymbol{v}^{(i)}$ as the updated example (Algorithm 4, line 14).

With this method, the particle swarm will move toward position with smaller $P\left(y \mid \boldsymbol{x}', \theta\right)$. After several iterations, the particle swarm will include the example that can mislead the classifier.

4. EXPERIMENTS

4.1. Experimental Setup

4.1.1. Dataset

MNIST[2] is a handwritten digits recognition dataset. It has a training set of 60,000 examples and a test set of 10,000 examples. Each example is an image with its label. The image has 28 × 28 pixels which contain only one color channel: the gray level.

CIFAR-10[3] is a color image dataset. Each example in the dataset is a 32 × 32 image with a label from 10 categories. There are 50,000 images in the training set and 10,000 images in the testing set. We use MNIST and CIFAR-10 to study the effects of multimodal adversarial attack against image classifier.

The data set used in the text adversarial attack is IMDB [22]. It contains 50,000 film reviews, including 25,000 in training sets and 25,000 in test sets. Each review has a rating score which is range from 1 to 10. Reviews with score greater than 7 are labeled as positive and reviews with score less than 3 are labeled as negative, so the classifier on IMDB conducts a 2-category classification. In addition to the 50,000 labeled reviews, the training set has another 50,000 unlabeled reviews, which can be used to unsupervised train language models.

4.1.2. Experiment platform

The hardware platform used in this experiment is a workstation with an i7-5930k CPU and 2 NVIDIA GeForce TITAN X GPUs. The operating system is Ubuntu 16.04. The TensorFlow [30] version is 1.4.1.

4.1.3. Target classifier

The LeNet [31] is the target image classifier and attacked by the proposed adversarial attacks. LeNet has an excellent performance on handwritten digits recognition. The LeNet is implemented withtensorflow [30] framework. The classifier has 2 convolutional and max pooling layers, 2 fully connection layers and a dropout layer. In the first convolutional layer, there are 32 kernels with size 5 × 5. Note that the "SAME" padding is used in the convolutional layer. The padding technique extends the edge of the 28 × 28 images and maintains its size before and after convolution. After the first 2 × 2 max pooling layer and a ReLU [32] activation function, the outputs are 32 feature images with size 14 × 14. The second convolutional uses

[2]http://yann.lecun.com/exdb/mnist/

[3]https://www.cs.toronto.edu/kriz/cifar-10-python.tar.gz

64 kernels with size 5×5. Similarly, the output of the second 2×2 max pooling layer are 64 feature images with size 7×7. Then there is a fully connection layer which takes 3136 ($= 7 * 7 * 64$) neurons as the input and outputs 1024-dimensional vector. There is a dropout layer with keep probability 0.5 after first fully connection layer. Last layer is also a fully connection layer which outputs 10-dimensional vector. The greatest float number in the 10-dimensional vector indicates the label of original input image. We use MNIST training set to train the LeNet model. The model achieves 99.2% accuracy on MNIST digital recognition test set.

A double layer bidirectional LSTM [33] network is the target text classifier. The hidden state number of LSTM cell is 512. The dimension of the word vector is 256. The vocabulary size is 86,934. The sequence length is 400. The training of RNN has two phases: language model pretraining and classification model training. To pretrain the language model, we use the output of current LSTM cell to predict the next word. Through the adam optimization method [34] with 0.001 initial learning rate, all of the labeled and unlabeled reviews in the training set are utilized to pretrain the language model and word embedding. After pretraining, we train the classification model, the LSTM structure are the same but a fully connected layer and a softmax layer is added in order to predict the sentence's probability of each class. The loss function is cross entropy. The training phase cost about 4 hours on the experimental platform.

4.1.4. Evaluation metrics

We propose the following metrics to evaluate the performance of the attacks.

Attack success rate is the ratio of successfully misled examples to the correctly classified examples. The success rate is defined with R. We denote a indicator function with I (*condition*). If *condition* is true then $I($ *condition* $) = 1$, otherwise $I($ *condition* $) = 0$.

$$R = \frac{\sum_{(\boldsymbol{x},y)\in\mathcal{D}} I(c(\boldsymbol{x}) = y) \cdot I\left(c\left(\boldsymbol{x}^*\right) \neq y\right)}{\sum_{(\boldsymbol{x},y)\in\mathcal{D}} I(c(\boldsymbol{x}) = y)} \tag{6}$$

Image perturbation rate evaluates the image's perturbation amount.

$$\delta_{image}\left(\boldsymbol{x}^*, \boldsymbol{x}\right) = \frac{\left\|\boldsymbol{x}^* - \boldsymbol{x}\right\|_2}{\left\|\boldsymbol{x}\right\|_2} \tag{7}$$

Text perturbation rate evaluates the text's perturbation amount. It is the ratio of the changed words to the length of text.

$$\delta_{text}\left(\boldsymbol{x}^*, \boldsymbol{x}\right) = \frac{1}{len(\boldsymbol{x})} \sum_{i=1}^{len(\boldsymbol{x})} I\left(\boldsymbol{x}_i^* \neq \boldsymbol{x}_i\right) \tag{8}$$

4.1.5. Baselines

To compare the proposed multimodal adversarial attack framework with other image adversarial attacks, we implement FGSM [17], BIM [18], JSMA [5], threshold attack [35], HopSkipJump [36] attack as the baselines. FGSM simply applies gradient ascent to modify images while BIM iteratively applies gradient ascent until attack successes or exceeds the perturbation limit. JSMA modifies images pixel by pixel. Threshold attack restricts the number of modified pixels. HopSkipJump is a black-box attack, but it develops an estimation method for calculating gradient. They are both iteration based attacks. These attacks against MNIST and CIFAR-10 classifier are implemented using ART,[4] which is an adversarial attack and defense tool box.

As for the text adversarial attack, we compare the proposed multimodal attacks with other 4 existing attacks: the greedy search attack [16], WS attack [9], [23] PWWS, fast gradient projection method [37] (FPGM). The greedy search attack, which is similar to beam search, selects some synonym words which can maintain the semantic and syntactic similarity in each iteration. Then it find one word that has the smallest probability of being classified as its original label after substitution. WS calculates best substitution word for each position. Then it calculates each position's influence on classification result and substitute words in order of the influences. PWWS is an improved WS. Its calculation of WS is weighted with probability change. Thus the WS of PWWS considers both each position's influence and probability change. FGPM is also a synonym substitution-based text adversarial attack. Its substitution is scored by the product of gradient magnitude and the projected distance between the original word and the candidate word in the gradient direction.

4.2. Multimodal Adversarial Attacks Against Image Classifiers

4.2.1. Attack effectiveness

To prove that the proposed framework is effective in attacking image classifier, we implement 5 baseline methods and Algorithms 2–4 to attack the target model. In the attack we can find that if we do not limit the perturbation amount, the attack will always succeed. Therefore, there should be restrictions in the algorithm, i.e., $d\left(\boldsymbol{x}^*, \boldsymbol{x}\right) < \Upsilon$. In the image attack, the perturbation rate (see Eq. (7)) are restricted to be less than 1.0. According to this limitation, the attack success rate in MNIST and CIFAR-10 dataset of each method is shown in Tables 1 and 2. All of the proposed attacks (beam search, GA search, PSO search) in the framework have over 90% success rate against image classifier, which indicates the framework are effective in generating image adversarial examples.

The examples in MNIST and CIFAR-10 are input into the framework to generate adversarial examples. Figures 3 and 4 show some of the adversarial examples. It can be seen from the figures that the modification pattern of FGSM and BIM are very similar. FGSM is a gradient ascent method, which adds all pixels with gradient sign. BIM iteratively adds gradient sign. This is the reason why FGSM and BIM has similar modification pattern. GA, PSO, and threshold add noises to the images in order to mislead the classifier. The reason why PSO generates these noises is that the particles change simultaneously in multiple directions, and the particles may move to an unpredictable location in space. GA generates noises due to its mutation on unpredictable positions. Threshold attack uses differential evolution (DE) to modify original images. The randomness in DE causes the noises in adversarial images. The generated examples of beam search and JSMA are similar to the original example. The reason is that these attacks modify images pixel by pixel. The pixels

[4]https://adversarial-robustness-toolbox.readthedocs.io/en/latest/

Table 1 Comparing the proposed attacks with the baseline attacks on MNIST dataset.

	Attack Success Rate (1.0 Perturbation)	Averaged Attack Time
FGSM	59.2%	7.01 ms $^{+0.11\ \text{ms}}_{-0.17\ \text{ms}}$
BIM	95.9%	211.75 ms $^{+0.51\ \text{ms}}_{-0.38\ \text{ms}}$
JSMA	96.8%	577.43 ms $^{+54.04\ \text{ms}}_{-48.80\ \text{ms}}$
Threshold	68.7% $^{+2.7\%}_{-3.3\%}$	264.47 ms $^{+3.59\ \text{ms}}_{-3.13\ \text{ms}}$
HopSkipJump	97.3%	23.32 s $^{+0.14\ \text{s}}_{-0.06\ \text{s}}$
Beam search	94.7%	13.87 s $^{+0.52\ \text{s}}_{-0.49\ \text{s}}$
GA search	92.8% $^{+0.9\%}_{-3.5\%}$	4.88 s $^{+0.99\ \text{s}}_{-1.10\ \text{s}}$
PSO search	97.4% $^{+1.0\%}_{-1.8\%}$	3.31 s $^{+1.11\ \text{s}}_{-0.85\ \text{s}}$

Table 2 Comparing the proposed attacks with the baseline attacks on CIFAR-10 dataset.

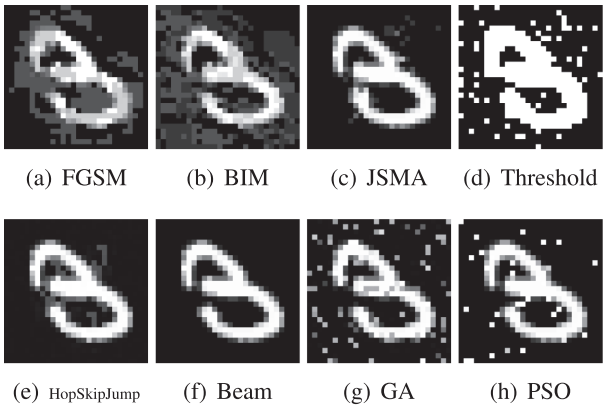
	Attack Success Rate (1.0 Perturbation)	Averaged Attack Time
FGSM	84.5%	9.47 ms $^{+0.15\ \text{ms}}_{-0.08\ \text{ms}}$
BIM	96.1%	387.43 ms $^{+0.43\ \text{ms}}_{-1.05\ \text{ms}}$
JSMA	97.4%	347.17 ms $^{+40.70\ \text{ms}}_{-41.21\ \text{ms}}$
Threshold	73.7% $^{+1.6\%}_{-3.1\%}$	592.11 ms $^{+1.54\ \text{ms}}_{-1.36\ \text{ms}}$
HopSkipJump	96.7%	47.37 s $^{+0.97\ \text{s}}_{-0.89\ \text{s}}$
Beam search	97.7%	30.52 s $^{+0.44\ \text{s}}_{-0.48\ \text{s}}$
GA search	92.6% $^{+2.5\%}_{-2.1\%}$	12.53 s $^{+0.44\ \text{s}}_{-0.48\ \text{s}}$
PSO search	94.1% $^{+1.7\%}_{-2.0\%}$	9.70 s $^{+0.44\ \text{s}}_{-0.48\ \text{s}}$

on the edge of the objects in images are usually more influential on classification result. Modifying the object edge generates examples that are very similar to the original example. HopSkipJump tends to modify influential local pixels to generate adversarial examples, so its attack pattern is that local area of image being modified.

4.2.2. Attack efficiency

As can be seen from Tables 1 and 2, the attacks proposed in the framework can effectively produce adversarial examples. In MNIST dataset, PSO can achieve the best attack success rate 98.4%. Although PSO has the best success rate, its adversarial examples are easily to be perceived by humans. The beam search is better in deceiving human perception. In CIFAR-10 dataset, beam search can achieve the best success rate. To quickly generate adversarial images, BIM and JSMA is better because of their short averaged attack time and relatively greater attack success rate. The FGSM and pixel attack is not recommended. Their attack success rates are too low.

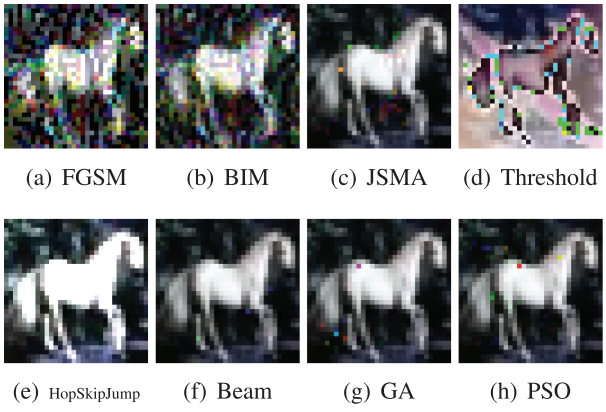
In image adversarial attack, a high efficiency means the attack has better success rate with less perturbation rate. Figures 5 and 6 shows



(a) FGSM (b) BIM (c) JSMA (d) Threshold

(e) HopSkipJump (f) Beam (g) GA (h) PSO

Figure 3 Comparison of the generated image examples in MNIST dataset. The modification pattern of FGSM and BIM are similar. Both of them craft adversarial images by modifying the entire pixels at the same time. GA, PSO, and threshold add noise to the images in order to mislead the classifier. JSMA, beam search, and HopSkipJump tend to modify only the influential pixels to minimize the amount of modification.



(a) FGSM (b) BIM (c) JSMA (d) Threshold

(e) HopSkipJump (f) Beam (g) GA (h) PSO

Figure 4 Comparison of the generated image examples in CIFAR-10 dataset.

the attack success rate–perturbation curves. The point in the figure represents an attack's success rate with a restricted perturbation rate. For example, a perturbation rate of up to 0.147 can make a beam search successfully attack 70% of the examples, so the point (0.147, 0.7) is on the curve of beam search attack. We can conclude with Figures 5 and 6 that in the baseline methods, JSMA has the best attack efficiency. In MNIST dataset, the attack success rate of beam search is 2.1% less than JSMA, but its perturbation rate is nearly half of JSMA. In CIFAR-10 dataset, beam search achieves greater attack success rate with less perturbation. Therefore, the beam search in the framework has better attack efficiency than other attack methods.

4.2.3. Human evaluation

In this part we study the imperceptibility of adversarial images. An imperceptible adversarial example is more likely to evade human perception so we implement a human evaluation. We randomly select images for human evaluation from adversarial examples generated by the 5 baseline attacks and 3 proposed attacks. For each attack we select 100 images, 50% of which are from original images and the rest are from adversarial images. So there are 700 images for one volunteer to determine whether the images are adversarial or not. We ask 8 volunteers to do the test. We use precision and recall to evaluate volunteers' results. Precision is the proportion of recognized real adversarial examples to the adversarial examples recognized by volunteers. Recall is the proportion of recognized real adversarial examples to the real adversarial examples. High precision indicates that volunteers can identify adversarial example precisely. High recall indicates that the volunteers can find the adversarial examples completely.

Volunteers averaged precision and recall are shown in Figure 7. We can see from the figure that the precisions of volunteers are above 84%, meaning that human can identify adversarial examples precisely. However, the recalls of each attack are various. BIM attack has the highest recall 91.5%, indicating that most of the adversarial examples generated by BIM are recognized by human. Beam search attack has the lowest recall 57. 1%. A low recall indicates that the adversarial examples can easily evade volunteers' perception. In other words, about half of the adversarial examples evade human detection. So we can conclude that the adversarial images of beam search attack have the best imperceptibility.

4.2.4. Study of hyperparameters

In this section, we study the effect of different hyperparameters on experimental results. The hyperparameters used in the proposed framework are: the candidate set size $M \times N$, the perturbation limit Υ, the beam search size, the PSO parameters (w, ϕ_p, ϕ_g. See Algorithm 4), the crossover rate, the mutation rate. In PSO, we use the parameters $\phi_p = \phi_g = 2.0$ and $w = 0.9$ as recommended by Higashi *et al.* [38] In beam search, the beam search size is the number of branches, which are restrict to $M \times N$. In each iteration of GA, we select two examples in the candidate set to execute the crossover process, so the crossover rate is $\frac{2}{M \times N}$. The crossover process generates two examples. Then the two generated examples are mutated to make the label's probability decrease, so the mutation rate is $\frac{2}{M \times N}$.

The above hyperparameters are all related to the candidate set size $M \times N$. To explore the proper $M \times N$, Table 3 shows attack success rate R of beam search when it attacking MNIST classifier. When different hyperparameters $M \times N$ are chosen, the success rates are various. We choose $M \times N = 4 \times 2$ in the image adversarial attacks.

Another hyperparameter that affects the success rate of the attack is the perturbation limit Υ. We also present the success rate R of the GA attack on the dataset MNIST when various Υ are set. Table 4 shows that the attack success rate R will be greater if larger perturbation limit Υ is set. We found that the attack success rate would be in a reasonable range if the perturbation limit $\Upsilon = 1.0$.

4.3. Multimodal Adversarial Attack Against Text Classifier

4.3.1. Attack effectiveness

Text adversarial attack can always succeed if we don't limit the perturbation amount. An extreme case is that a text example

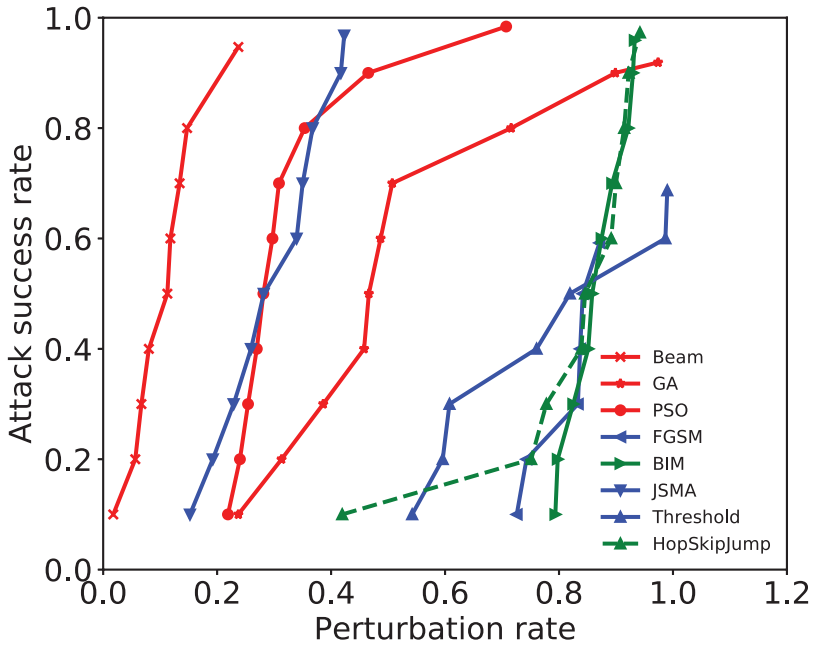


Figure 5 The attack success rate–perturbation curve in MNIST dataset. It shows the range of perturbation rate and attack success rates. For example, the adversarial images' perturbation rates of beam search range between 0 and 0.25, which is the least perturbation rate in all attacks. Beam search can obtain an attack success rate over 90%. Other attacks need more perturbations to achieve the same attack success rate. Thus, the attack efficiency of beam search outperforms other attack methods.

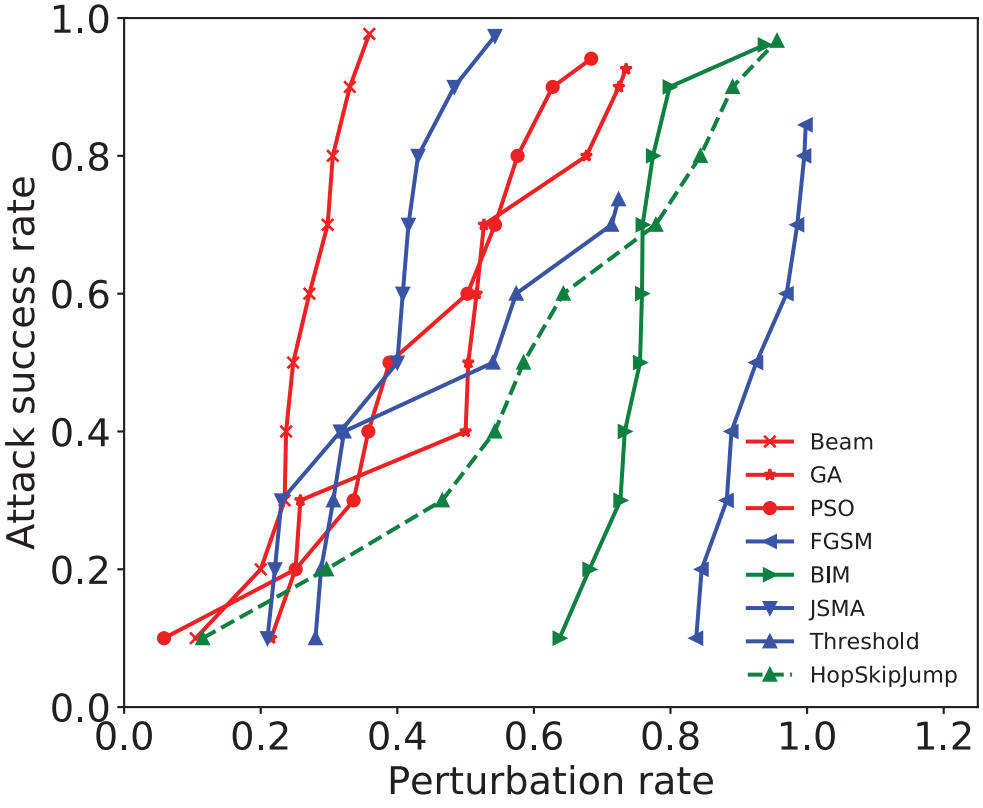


Figure 6 The attack success rate–perturbation curve in CIFAR-10 dataset.

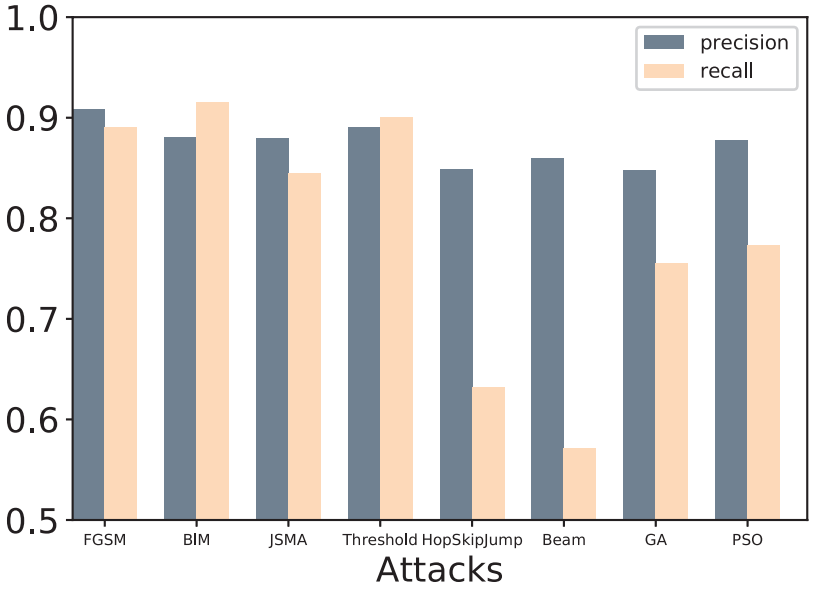


Figure 7 | Human testers' averaged precision and recall of recognizing adversarial images.

Table 3 | The attack success rate of beam search when various $M \times N$ are set.

$M \times N$	2×2	2×3	3×2	2×4	4×2
R	93.8%	92.6%	92.7%	94.5%	94.7%

Table 4 | The attack success rate of GA when various Υ are set.

Υ	0.5	0.6	0.7	0.8	0.9	1.0
R	46.6%	48.6%	50.7%	71.6%	89.8%	91.9%

is unlimited modified until all of its words are changed. We use text perturbation rate (see Eq. (8)) to limit modification. So the $d(\boldsymbol{x}^*, \boldsymbol{x}) < \Upsilon$ in the proposed adversarial attacks becomes $\delta_{text}(\boldsymbol{x}^*, \boldsymbol{x}) < \Upsilon$, while Υ is restricted to 20%.

With the perturbation restriction, we implement the baseline attacks and the 3 attacks in the framework. Some examples of adversarial text are shown the substitute words in Table 5 should be marked in red color. The examples generated by GA and PSO have more diversity. This may be caused by the randomness of GA and PSO algorithm. However the randomness will cause more unnecessary modification. The performances of these 7 attacks are shown in Table 6. The attack success rate of beam search is the best (91.5%) among these attacks. Greedy search can be regarded as a special beam search. In each iteration, the greedy search select only one example with smallest original label probability, while beam search select several examples. Although GA do not achieve a high success rate, it has advantage in the time overhead of finding adversarial examples. WS and PWWS have relatively low attack success rates at 20% perturbation. The reason is that in every position of the sentence, only one substitute word is selected. Comparing with N candidate substitute words are selected in our framework, WS and PWWS are more difficult to find adversarial texts.

4.3.2. Attack efficiency

In text adversarial attack, a higher efficiency means the attack has better success rate with less perturbation rate. As we restrict the perturbation with $\delta_{text}(\boldsymbol{x}^*, \boldsymbol{x}) < \Upsilon$, we test various Υ for 10 times, which range from 0 to 0.2. Their corresponding attack success rate of the proposed and baseline attacks are shown in Figure 8. For example, we have 91.5% examples successfully attacked when we set $\Upsilon = 0.2$ in beam search attack. If we set $\Upsilon = 0.1$, the success rate of beam search attack will be 65.9%. The success rate - perturbation curves are shown in Figure 8. We can conclude from it that beam search has the best attack efficiency comparing with other attacks.

The attack efficiency of GA and PSO are relatively low. GA and PSO search method simultaneously modify multiple words, which will cause unnecessary modifications when attack success. Comparing the rest 5 attacks, the beam and greedy search attack are better in efficiency. The reason is that beam and greedy's search coverage is wider. WS, PWWS, and FGPM only select one substitute word in each position, while our methods have N candidate substitute words in each position. Thus beam and greedy search can find more suitable words to modify text, resulting in their less text perturbation rate.

4.3.3. Human evaluation

In this part, we evaluate the imperceptibility of adversarial texts. We randomly select 100 sentences of each attack for human evaluation. Half of the sentences are clean text from test set, and rest of them are adversarial texts. Thus there are 600 sentences for one human tester to determine whether the sentences are adversarial or not. We ask 8 volunteers to do the test, the same as we do in the image human evaluation.

The averaged precision and recall are 0.868 and 0.706 respectively. As analyze in Section 4.2.3, high precision indicates that volunteers can identify adversarial examples precisely, and a relatively

Table 5 Comparison of the text examples.

Original	I don't care if some people voted this movie to be bad. If you want the truth this is a very good movie ! It has everything a movie should have, you really should get this one.
Greedy	I don't care if some people voted this movie to be bad. If you want the truth this is a very alright movie ! It has everything a movie should have, you really should get this one.
WS	I don't care if some citizens voted this movie of be bad. If you want the truth this is une very good movie ! It has everything a movie should have, you really should get this one.
PWWS	I don't care if some people voted this movie to be bad. If you want the truth this is une very good movie ! It has everything une movie should have, you really should get this one.
FGPM	I don't care if some citizens voted this movie to be rotten. If you want the veracity this is a very good movie ! It has everything a movie should have, you really should get this one.
Beam	I don't care if some people voted this movie to be bad. If you want the truth this is a eminently good movie ! It has everything a movie should have, you really should get this one.
GA	I don't care if some citizens voted this movie to be rotten. If you want the truth this is a very good movie ! It has everything a movie should have, you really should get this paradis.
PSO	I don't care if some people adopted this movie to be rotten. If you want the truth this is a very good movie ! It has everything a movie would ha, you really should get this one.

Table 6 Comparing the proposed attacks with other text adversarial attacks.

	Attack Success Rate (20% Perturbation)	Averaged Attack Time
Beam search	91.5%	95.3 s
GA	63.4%	6.6 s
PSO	55.4%	241.2 s
Greedy search	90.7%	36.5 s
WS	54.8%	16.2 s
PWWS	59.3%	39.6 s
FGPM	88.1%	112.4 s

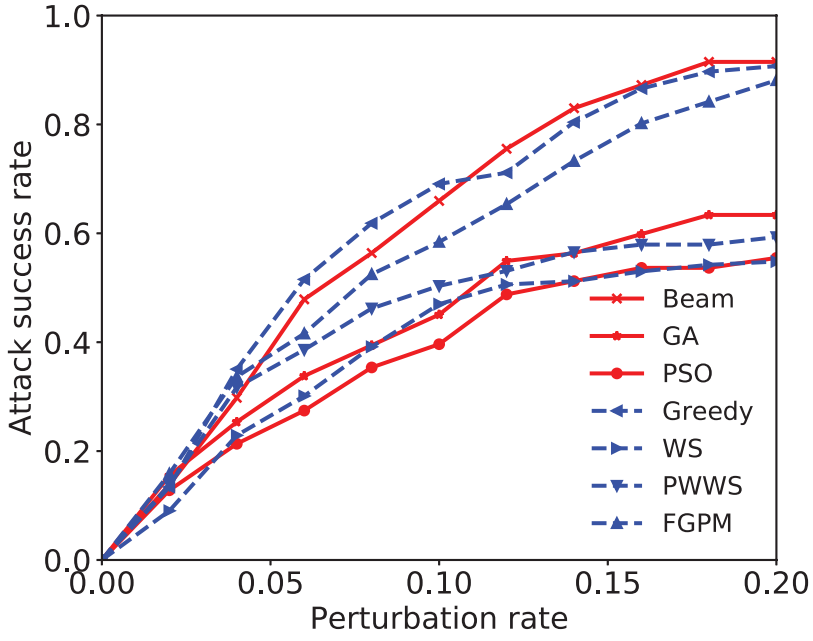


Figure 8 The attack success rate–perturbation curve of text adversarial attacks.

low recall indicates that the volunteers could not completely find the adversarial examples. To investigate what kind of adversarial texts can easily evade human detection, we group the adversarial texts according to the perturbation rate (see Eq. (8)). There are 4 groups (group 1 : $0 < \delta_{\text{text}} \leqslant 0.05$,..., group 4: $0.15 < \delta_{\text{text}} \leqslant 0.20$). The recalls of these groups are: 0.280, 0.426, 0.614, 0.735. Therefore, the adversarial texts with small perturbation rate are more likely to evade human perception. So beam search attack has the best imperceptibility because of its higher attack efficiency and less perturbation rate.

5. CONCLUSIONS

In this paper, a framework is proposed to craft multimodal adversarial examples. The image and text examples are formalized as tensors. In the framework, CSG process is proposed to generate multiple candidate examples from one example. Then 3 local/random search methods are utilized to update the candidate examples and search the adversarial example. Experiments show that the framework can craft adversarial examples that have good attack success rates against text and image classifiers. The time overhead of GA search are better than beam search in both image and text attack. But the beam search has better performance in attack efficiency. In the human evaluation, the adversarial example generated by beam search are more likely to evade human detection.

CONFLICTS OF INTEREST

The authors declare they have no conflicts of interest.

AUTHORS' CONTRIBUTIONS

Paper writing and experiment implementation, Zibo Yi; Methodology, Zibo Yi and Jie Yu; Revising and reviewing, Jie Yu, Yusong Tan, Qingbo Wu; Supervision, Jie Yu, Yusong Tan, Qingbo Wu; Funding acquisition, Jie Yu.

Funding Statements

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

REFERENCES

[1] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, R. Fergus, Intriguing properties of neural networks, arXiv preprint arXiv:1312.6199, 2013.

[2] M. Sharif, S. Bhagavatula, L. Bauer, M.K. Reiter, Accessorize to a crime: real and stealthy attacks on state-of-the-art face recognition, in ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 2016.

[3] H. Hosseini, S. Kannan, B. Zhang, R. Poovendran, Deceiving Google's perspective API built for detecting toxic comments, arXiv preprint arXiv:1702.08138, 2017.

[4] B. Luo, L. Yannan, L. Wei, Q. Xu, Towards imperceptible and robust adversarial example attacks against neural networks, in National Conference on Artificial Intelligence, New Orleans, Louisiana, USA, 2018, pp. 1652–1659.

[5] N. Papernot, P.D. Mcdaniel, S. Jha, M. Fredrikson, Z. Berkay Celik, A. Swami, The limitations of deep learning in adversarial settings, in IEEE European Symposium on Security and Privacy, Saarbruecken, Germany, 2016, pp. 372–387.

[6] N. Narodytska, S.P. Kasiviswanathan, Simple black-box adversarial attacks on deep neural networks, in 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Honolulu, HI, USA, 2017, pp. 1310–1318.

[7] J. Su, D. Vasconcellos Vargas, K. Sakurai, One pixel attack for fooling deep neural networks, IEEE Trans. Evol. Comput. 23 (2019), 828–841.

[8] N. Papernot, P.D. Mcdaniel, J. Goodfellow, S. Jha, Z. Berkay Celik, A. Swami, Practical black-box attacks against machine learning, in Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, Abu Dhabi, UAE, 2017, pp. 506–519.

[9] S. Samanta, S. Mehta, Towards crafting text adversarial samples, arXiv preprint arXiv:1707.02812, 2017.

[10] J. Ebrahimi, A. Rao, D. Lowd, D. Dou, 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, 2018, pp. 31–36.

[11] R. Jia, P. Liang, Adversarial examples for evaluating reading comprehension systems, in Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Copenhagen, Denmark, 2017, pp. 2021–2031.

[12] N. Papernot, P. McDaniel, A. Swami, R. Harang, Crafting adversarial input sequences for recurrent neural networks, in MILCOM 2016- 2016 IEEE Military Communications Conference, IEEE, Baltimore, MD, USA, 2016, pp. 2016–2016.

[13] J. Li, S. Ji, T. Du, B. Li, T. Wang, Textbugger: generating adversarial text against real-world applications, arXiv preprint arXiv:1812.05271, 2018.

[14] B. Liang, H. Li, M. Su, P. Bian, X. Li, W. Shi, Deep text classification can be fooled, in Proceedings of the 27th International Joint Conference on Artificial Intelligence, Stockholm, Sweden, AAAI Press, 2018, pp. 4208–4215.

[15] A. Yi-Ting Tsai, M.-C. Yang, H.-Y. Chen, Adversarial attack on sentiment classification, in Proceedings of the 2019 ACL Workshop Black-boxNLP: Analyzing and Interpreting Neural Networks for NLP, Florence, Italy, 2019, pp. 233–240.

[16] V. Kuleshov, S. Thakoor, T. Lau, S. Ermon, Adversarial examples for natural language classification problems in International Conference on Learning Representations, Vancouver, Canada, 2015.

[17] I.J. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples, in International Conference on Learning Representations, San Diego, CA, USA, 2015.

[18] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, A. Vladu, Towards deep learning models resistant to adversarial attacks, arXiv: Machine Learning, 2017.

[19] N. Carlini, D.A. Wagner, Towards evaluating the robustness of neural networks, in 2017 IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, 2017, pp. 39–57.

[20] N. Papernot, P. McDaniel, X. Wu, S. Jha, A. Swami, Distillation as a defense to adversarial perturbations against deep neural networks, in 2016 IEEE Symposium on Security and Privacy (SP), IEEE, San Jose, CA, USA, 2016, pp. 582–597.

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

[22] A.L. Maas, R.E. Daly, P.T. Pham, D. Huang, A.Y. Ng, C. Potts, Learning word vectors for sentiment analysis, in Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Portland, OR, USA, 2011, pp. 142–150.

[23] S. Ren, Y. Deng, K. He, W. Che, Generating natural language adversarial examples through probability weighted word saliency,in ACL 2019: The 57th Annual Meeting of the Association for Computational Linguistics, Florence, Italy, 2019, pp. 1085–1097.

[24] M. Alzantot, Y. Sharma, A. Elgohary, B.-J. Ho, M. Srivastava, K.-W. Chang, Generating natural language adversarial examples, in Proceedings of the 2018 Conference on Emplrlcal Methods in Natural Language Processing, Brussels, Belgium, 2018, pp. 2890–2896.

[25] D. Jin, Z. Jin, J.T. Zhou, P. Szolovits, Is Bert really robust? A strong baseline for natural language attack on text classification and entailment, in AAAI 2020: The Thirty-Fourth AAAI Conference on Artificial Intelligence, New York, USA, 2020.

[26] J. Devlin, M.-W. Chang, K. Lee, K.N. Toutanova, Bert: pre-training of deep bidirectional transformers for language understanding, in Proceedings of the 2019 Conference of the North American Chapter of the Associatlon for Computatlonal Llngulstlcs: Human Language Technologies, Volume 1 (Long and Short Papers), Minneapolis, MN, USA, 2018, pp. 4171–4186.

[27] B. Kulynych, J. Hayes, N. Samarin, C. Troncoso, Evading classifiers in discrete domains with provable optimality guarantees, arXiv preprint arXiv:1810.10939, 2018.

[28] J. Gilmer, R.P. Adams, I. Goodfellow, D. Andersen, G.E. Dahl, Motivating the rules of the game for adversarial example research, arXlv preprint arXlv:1807.06732, 2018.

[29] T. Miyato, A.M. Dai, I. Goodfellow, Adversarial training methods for semi-supervised text classification, arXiv preprint arXiv:1605.07725, 2016.

[30] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, *et al.*, Tensorflow: large-scale machine learning on heterogeneous distributed systems, arXlv preprint arXlv:1603.04467, 2016.

[31] Y. LeCun, P. Haffner, L. Bottou, Y. Bengio, Object recognition with gradient-based learning, in: D.A. Forsyth, J.L. Mundy, V.di Gesú, R. Cipolla (Eds.), Shape, Contour and Grouping in Computer Vision, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, Germany, 1999, pp. 319–345.

[32] X. Glorot, A. Bordes, Y. Bengio, Deep sparse rectifier neural networks, in Proceedings of the Fourteenth International Conference on Artlficial Intelligence and Statistics, Ft. Lauderdale, FL, USA 2011, vol. 15, pp. 315–323.

[33] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (1997), 1735–1780.

[34] D.P. Kingma, J. Lei Ba, Adam: a method for stochastic optimization, in ICLR2015: International Conference on Learning Representations, San Diego, CA, USA, 2015.

[35] S. Kotyan, D. Vasconcellos Vargas, Adversarial robustness assessment: why both l_0 and l_∞ attacks are necessary, arXiv preprint arXiv:1906.06026, 2019.

[36] J. Chen, M.I. Jordan, M.J. Wainwright, Hopskipjumpattack: a query-efficient decision-based attack, in 2020 IEEE Symposium on Security and Privacy (SP), Los Alamitos, CA, USA, 2020, pp. 1277–1294.

[37] X. Wang, Y. Yang, Y. Deng, K. He, Fast gradient projection method for text adversary generation and adversarial training, arXiv preprint arXiv:2008.03709, 2020.

[38] N. Higashi, H. Iba, Particle swarm optimization with Gaussian mutation, in Proceedings of the 2003 IEEE Swarm Intelligence Symposium SIS'03 (Cat. No. 03EX706), Indianapolis, IN, USA, 2003, pp. 72–79.