Research Article

A Sparsity-Limitation-Based High-Dimensional Distribution Searching Algorithm for Adversarial Attack

Chenxi Zhu,1 Haobo Wang,1 Yan Zhuang,1 Jie Li,1,2 and Yangjie Cao1

1School of Cyber Science and Engineering, Zhengzhou University, Zhengzhou 450000, China
2Department of Computer Science and Engineering, Shanghai Jiaotong University, Shanghai 200240, China

Correspondence should be addressed to Chenxi Zhu; chenxi_zhu@gs.zzu.edu.cn

Received 20 January 2022; Revised 6 February 2022; Accepted 14 February 2022; Published 21 March 2022

Academic Editor: Min Xia

Copyright © 2022 Chenxi Zhu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Deep neural networks-based image classification systems could suffer from adversarial attack algorithms, which generate input examples by adding deliberately crafted yet imperceptible noise to original inputs. To reduce the impact on human visual sense and to ensure adversarial attack ability, the input image needs to be modified by pixels in considerable iterations which is time consuming. By using sparse mapping network to map the input into a higher dimensional space, searching space of adversarial perturbation distribution is enlarged to better acquire perturbation information. Taking both searching speed and searching effectiveness into consideration, sparsity limitation is introduced to suppress unnecessary neurons during parameter updating process. Based on different eye sensitivity of different colors, maps of each color channel are disturbed by perturbations with different strengths to reduce visual perception. Numerical experiments confirm that compared with the state-of-the-art adversarial attack algorithms, the proposed SparseAdv performs a relatively high attack ability, better imperceptible visualization, and faster generation speed.

1. Introduction

In recent years, artificial intelligence technology has ushered in rapid development, in which deep learning has emerged strongly. Deep learning has demonstrated outstanding results in a variety of domains, including image classification [1–3], speech recognition [4], object detection [5], and self-driving [6]. Though deep learning network shows excellent performance, its complicated internal structure and interpretability have raised security concerns, as such if a model might be compromised to make wrong prediction when exposed to adversarial attacks [7]. These attacks deliberately craft the input examples to mislead the deep learning network, and such modifications could be hard to be detected by bare human eyes [8]. The crafted examples are named as adversarial examples.

Discovery of adversarial examples has gained increasing attentions and researchers began to explore adversarial attack algorithms. According to generation methods of adversarial examples, the examples could be generated whether by directly modifying each input example or training a generator to achieve mass generation. The former generation methods are traditional adversarial example generation methods which update the input examples guided by feedback from specially designed loss function [7]. These methods achieve high successful attack rate, whereas sacrifice generation efficiency due to time cost on the change of each individual example. Especially in image classification domain, traditional generation methods need to alter all pixel values of an image during each updating iteration. For example, Carlini and Wagner (CW) attack [9] cost around two minutes to generate an adversarial example. Therefore, the latter generation methods which could speed up generation of adversarial examples have been proposed. AdvGAN [10] trained a generative adversarial network (GAN) to learn mapping from an original example to the adversarial one and the trained GAN could immediately alter the original input to its adversarial version. Moreover, authors of [11] utilized autoencoder as an auxiliary generator...
to generate adversarial priors and thus reduced iterations for searching adversarial distributions. Instead of updating each pixels on the original input during the training process, the trained generator is able to mass generate adversarial examples in a fast speed.

The barely seeking local optimum, i.e., Nash equilibrium, with fluctuate results challenges the training process. Additionally, other generation methods mainly use generator as supplementary means, which still need abundant time-consuming iterations for further adversarial alteration. We try to realize and propose that an individual generator is enough to realize generator-based adversarial attacks and propose a new generator structure which maps the data from input space to high-dimensional space, and thus excavates more effective information for disturbing. Moreover, Kullback-Leibler divergence (KLD) is adopted for sparsity limitation on neurons in hidden layers. With sparsity limitation, computation complexity caused by high dimensional mapping could be reduced. To make the attack more imperceptible to human eyes, we conduct suitable operations on maps from different color channels based on biological characteristics of human visual system. Our contributions are summarized as follows:

(i) We search the required information in high-dimensional space and train a sparse mapping network as a generator, which can raise dimension of the input to dig out much effective perturbation and speed up generation process of adversarial examples

(ii) A KLD loss is added in loss function for sparsity limitation, so that neurons in the generator are selectively activated to make the training procedure more efficient

(iii) Based on different eye sensitivity of different colors, maps of each color channel are disturbed by perturbations with different strengths to reduce visual perception. Experimental results show that SparseAdv achieves higher successful rates of attacking and is also less perceptible in visual sense

The rest of this paper is organized as follows. Section 2 briefly overviews the related works of adversarial attacks. In Section 3, we specifically state the attack methodology of SparseAdv. We present the experimental results in Section 4 and conclude the paper in Section 5.

2. Related Work

2.1. White-Box Adversarial Attacks. Earlier adversarial attacks were mostly white-box adversarial attacks, in which attackers had access to all information (i.e., model structure, parameter, and hyper-parameter) of a target model. Adversarial attack was firstly proposed by Szegedy et al. [8] in 2013, they found that subtle modification of an example was able to mislead a deep learning model, which was then widely researched by experts from various fields. The example with nonsensitive (to human eye) yet misleading (to deep learning model) perturbation was recognized as adversarial example. Goodfellow et al. [7] proposed fast gradient sign method (FGSM), which used gradient direction to disturb clean examples and demonstrated that adversarial examples had unstable transferability among different target classifiers. Based on FGSM, a basic iterative method (BIM) [12] was subsequently proposed to superimpose the noise step by step. So that feedback of the gradient information could be fully used under the same disturbance and BIM was able to successfully achieve better results with a smaller disturbance effect. Considering that the complex nonlinear model might change drastically in a very small range, projected gradient descent [13] method used gradient feedback to limit the noise to a small range. CW attack [9] defined two impact factors according the two final goals—imperceptible visual effect to human and misleading result to the recognition model. CW attack updated the input by optimizing the two factors and showed excellent performance.

2.2. Black-Box Adversarial Attacks. In real scenarios, black-box attacks which do not need any information of the target model and show high transferability among various models have gained attention. The transferability of adversarial example refers to the successful attacking rate of the example set to a target model, could be almost the same when this set attacks another model. Therefore, concept of transferability means the ability to keep the performance when an adversarial example set changes the task environment, i.e., the target model to be attacked. As a result, transferability of adversarial examples indicates flexibility and adaptability of an adversarial attack algorithm while facing different task environments. Moreover, higher transferability of adversarial example set in white-box research could make it more transferable to black-box adversarial attack. Dong et al. [14] proposed the Momentum Iterative Fast Gradient Sign Method (MI-FGSM) [14] in the Neural Information Processing Systems 2017 competition. MI-FGSM integrated the momentum term into iterative process, stabilizing the update direction and getting rid of undesirable local maximum. Diversity input method [15] proposed to enhance data diversity by randomly transforming clean examples, thereby improving transferability of adversarial examples. Translation Invariant Method [16] proposed a translation-invariant attack algorithm to generate adversarial examples that were less sensitive to white-box model and so that transferred better among black-box models. AutoZoom [11] added adversarial perturbation in low-dimensional space and mapped the perturbation to high-dimensional embedding space by autoencoder, thus reducing the mean query counts in finding successful adversarial examples. Inspired by AutoZoom, TREMBA [17] utilized AutoZoom’s framework for gaining perturbation prior. Then by applying a standard black-box attack method such as Natural Evolution Strategy (NES) [18] on the embedding space, adversarial perturbations could be found efficiently for a target model.

2.3. Generator-Based Adversarial Attacks. When the adversarial examples are generated by altering the input to maximize the classification loss [8], it requires multiple iterations to modify on the input. Such iterations lower the efficiency of generating adversarial examples and thus consume much
time while facing the tasks that need to generate a plenty of adversarial examples. Generator-based adversarial attacks are thus proposed for fast adversarial example generation, including generating adversarial examples by general generators with neural network structure and by GAN. AutoZoom [11] and TREMBA [17], which were the black-box adversarial attacks mentioned above, both utilized autoencoder as a generator to generate perturbation prior. However, iterations on the input were still required for achieving the whole attack process. As for GAN, it is a generative network constituted by a generator and discriminator, which could be trained to learn the mapping between clean examples or random noise and adversarial examples. An adversarial example could be directly derived from trained network that generating iterations on each input is skipped [10, 19, 20]. Xiao et al. [10] proposed AdvGAN to generate an adversarial perturbation to be mapped from an original input by its generator, which was then added to this corresponding original image. The discriminator in AdvGAN judged whether the input example composed of original image and perturbation was an adversarial example. Based on AdvGAN, AdvGAN++ [19] proposed by Jandial et al. introduced latent vectors from the target classifier as the input of GAN to generate adversarial examples. Rob-GAN [20] introduced adversarial examples into GAN that not only could accelerate the training process by rapidly generating adversarial examples but also improve the quality of generating image and the robustness of discriminator. Zhao et al. [21] introduced Natural GAN model which could search adversarial example vector in low-embedding latent space and generate more targeted and natural adversarial perturbations. Deb et al. [22] focused on adversarial face synthesis and utilized human identity matching information to train GAN to gain face adversarial examples.

GAN performs well in speeding up adversarial example generation, whereas the two loss functions in generator and discriminator makes the convergence point hard to reach, which challenges the training process [23]. We propose to abandon discriminator which is used for confining pixel value to reduce visual effect on bare eye and conduct special noise adding method based on biological characteristics of the human eyes for better visualization. Therefore, the retained single loss function in training model could ease the convergence process. Researchers have performed diverse noise adding operations to balance visual effects and attack ability. Shamsabadi et al. [24] and Bhattad et al. [25] introduced unrestricted perturbations by exploiting image semantics such as color and texture to generate effective and photorealistic adversarial examples. Based on superpixel and attentional map, Dong et al. [26] preserved the attack ability even in highly constrained modality space, and it was robust to image processing based defense and steganalysis-based detection. Above methods focus on the machine’s characteristics for recognizing images, which omit an important fact that perturbations added on adversarial examples are expected to be imperceptible to humans rather than machines. Human eyes are unevenly sensitive to red, green, and blue colors [27, 28], while convolution operations, which are conducted by machines, on maps from each color channel are the same. So that different perturbation values on different channels are necessary to achieve better balance between attack ability and visual effect. More specifically, weaker perturbations are added on maps from the color channel, to which human eyes are more sensitive, and stronger perturbations are added on maps from other color channels.

3. Attack Methodology

In this section, we first overview SparseAdv and then introduce each procedure in detail.

3.1. Overview of SparseAdv. In image classification area, adversarial examples are the clean image inputs added with specifically designed noise which could not only deceive the deep learning models based classifiers but are also least noticeable to human visual sense. The adversarial example generating framework SparseAdv is designed for the above task. As shown in Figure 1, SparseAdv contains a sparse mapping neural network that maps input image from input space to high-dimensional space to search special distribution which could potentially be the noise to mislead the categorization of target classifiers. This special distribution extracted by neural network in SparseAdv is then combined with its original image and fed into classifier. The image, disturbed by the searched distribution, tends thus to be classified into a category different from its true labeled class. Therefore, an adversarial example is then produced by the SparseAdv framework in theory. The loss function contains two parts, one of which is designed that the predicted label is guided to be the class with the secondary confidence [29] within four networks, which are ResNet-50 [30], VGG-16 [31], GoogleNet [32], and MobileNet-v2 [33], in our research. The other part introduces KLD, which is an indicator for sparsity measure. Suppression on KLD achieves sparsity limitation that benefits effective application of each neuron in a neural network. And then the whole loss function is fed back to update the parameters of sparse mapping neural network in SparseAdv. To increase the transferability, the overall loss function is constituted by the loss functions from each attack process on the four target networks. Successful adversarial attack means to generate perturbations which could change categorization result of the classifier, while to deceive human visual sense. Since human visualization is the focus, we follow the analysis of human eyes’ biological characteristics in [27, 28] and learn that those are differently sensitive to different colors. Thus, different operations are conducted on maps from each color channel. For the insensitive color to human eyes, relatively strong noises are added on maps from corresponding color channels, while weak disturbing is conducted to reduce observability for the sensitive color. Note that strength of the noise is represented by the thickness of blue lines in Figure 1.

3.2. Formal Description. Once get the input image \(x\), a sparse mapping network is used as a generator \(G(\cdot)\) and extract perturbation distribution \(\mathcal{G}(\cdot)\) from input space. We combine the generator operation with general adversarial attack algorithm, in which the input \(x\) with its adversarial perturbation \(\delta\) is updated by minimizing the loss function \(L_{adv}(x, y)\) defined as follows from CW attack [29], which leads the label to another class with the secondary confidence:
where \((x, y)\) is the input \(x\) with its label \(y\) in a training set, \(\mathcal{C}(\cdot)\) is the target classifier, and \(\kappa\) is a margin parameter which controls the strength of adversarial examples. So that the generated perturbation \(\delta\) is substituted by \(G(x; \theta)\), generator \(G(\cdot)\) with input \(x\) and parameter \(\theta\), in Eq. (2):

\[
L_{\text{adv}}(x, y) = \max \left\{-\kappa, \mathcal{C}[x + \delta] - \max_{c \neq y} \mathcal{C}[x + \delta] \right\},
\]

(1)

Another penalty term KLD (i.e., \(\text{KL}(\rho \| \hat{\rho}_j(x)) = \sum_{x} \rho \cdot \log (\rho / \hat{\rho}_j(x))\)) is added to achieve sparsity limitation. Note that detailed introduction of KLD function is demonstrated in Section 3.4. Thus, the final loss function \(L_{\text{final}}\) is defined as follows:

\[
L_{\text{final}}(x, y) = L_{\text{adv}}(x, y) + \text{KL}(\rho \| \hat{\rho}_j(x)).
\]

(3)

Instead of directly updating the input image, SparseAdv updates the parameters \(\theta\) of the generator. The upgrade process of parameter \(\theta\) in sparse mapping network \(\mathcal{G}\) according to loss function \(L_{\text{final}}(x, y)\) is as follows:

\[
\theta = \arg\min_{\theta} L_{\text{final}}(\mathcal{C}[G(x; \theta)], y).
\]

(4)

The trained network \(\mathcal{G}\) is thus able to generate adversarial perturbations once given the input images. Attack process is obtained by averaging out their loss functions as a new overall loss function shown below:

\[
L_{\text{adv,trans}}(x, y) = \frac{1}{N} \left\{ \sum_{n=1}^{N} L_{\text{final}}(x, y) \right\},
\]

(5)

where \(N\) denotes the number of target classifiers and \(n\) denotes the classifier index.

3.3. Sparse Mapping Network. Under recognition task, hidden layers in deep neural networks are designed for obtaining features, each layer of which chooses key information from the anterior layer for correct feature extraction. Therefore, size of the feature map generated by each hidden layer is diminishing as the layer goes deeper. General adversarial example generator follows such design, whereas ignores the fact that the key information under adversarial attack task requires disturbing ability rather than correct feature. Above analysis motivates us to abandon general structure design of neural network, which usually conducts down-sampling during feature extraction procedure, and raises the dimension of input image at
shallow layers to enlarge the searching space for perturbation information.

In our SparseAdv framework, the sparse mapping network consists of an up-sampling module, which mainly utilizes the deconvolutional layer for dimension expansion, and a down-sampling module, which uses the convolutional layer for effective information extraction and the max-pooling layer for dimension adjusting. As shown in Figure 2, the right part of the figure demonstrates the structure of sparse mapping network, while the left part of the figure gives the detailed design of two key blocks including Conv3D block and ConvTP3D block. Each Conv3D block contains a three-dimensional convolutional layer (i.e., Conv3D in the figure), a rectified linear unit (ReLU) layer, and a batch normalization layer. Each ConvTP3D block has the same structure with the Conv3D block, except the first layer which is substituted by a three-dimensional deconvolutional layer (i.e., ConvTP3D in the figure).

The deconvolutional layer is designed for up-sampling, which is used to enlarge searching space in SparseAdv algorithm. Specifically, it is a special convolutional operation, before which the size of input image is enlarged by supplementing zeros in proportion. Figure 3 gives an example for deconvolutional operation, in which a $3 \times 3$ input is scaled up to the size of $5 \times 5$ after deconvolutional operation. The batch normalization layer speeds up the training process by using a higher learning rate, thus the retraining process consumes much less time. A max-pooling layer is invoked for down-sampling after some convolutional layers to adjust data dimensions to the same size as the input image. Note that the introduction of nonlinear factors—ReLU—makes the model more controllable.

As a kind of GAN, AdvGAN contains a generator and a discriminator, each of which aims to converge its own loss function to optimal result. Generator of AdvGAN is used for misleading the target classifier, while the discriminator of it confines value change of pixels in an algorithm-allowed range. It would be a challenge on the overall gradient descent process because both functions have distinguish gradient directions [23]. Moreover, generator structure of GAN is designed as a normal autoencoder which maps the input to low-dimensional space, limiting access to effective perturbation information. We use sparse mapping network as is introduced above and conduct color-channel-based perturbation adding methods by analyzing biological characteristics of the human eyes to avoid loss function for those algorithm restrictions on pixel values. With our specially designed dimension raising structure and color-channel-based perturbation adding method, single loss function of SparseAdv is easy to be converged to expected point.

3.4. Kullback-Leibler Divergence. KLD is a method to describe difference between two probability distributions. Given a data set $X$ and its two probability distribution $P$ and $Q$, KLD $KL(P\|Q)$ is defined as follows:

$$KL(P\|Q) = \sum_{i \in X} P(i) \cdot \log \left( \frac{P(i)}{Q(i)} \right). \quad (6)$$
When there is a tremendous quantity of neurons in a neural network’s hidden layers, some limitations are necessary for reducing computational complexity. Thus, KLD is added to the loss function to achieve sparsity limitation on sparse mapping network.

More specifically, given an input $x$, let $a_j(x)$ denote activation degree of the hidden neuron $j$, the average activation degree of $j$ could be formalized as follows:

$$
\hat{\rho}_j = \frac{1}{M} \sum_{m=1}^{M} a_j(x^{(m)}),
$$

where $M$ denotes the number of samples in data set and $m$ denotes index of each sample. We could define an sparsity parameter $\rho$, which is generally a low value close to 0 (e.g., $\rho = 0.05$). Thus, the average activation degree $\hat{\rho}_j$ is expected to approach $\rho$ for sparsity limitation. To achieve such limitation, an additional penalty factor is added to the loss function to punish those cases where $\rho$ and $\hat{\rho}_j$ are significantly different, so that the average activation degree of hidden neurons is kept within a small range. We choose KLD as a metric to measure the difference between $\rho$ and $\hat{\rho}_j$ as follows:

$$
\text{KL}(\rho \parallel \hat{\rho}_j) = \sum_{i \in X} \rho \cdot \log \left( \frac{\rho}{\hat{\rho}_j} \right).
$$

**Figure 4:** Visual comparisons on three channels.

| Attack     | ResNet-50 Success (%) | Time | VGG-16 Success (%) | Time | GoogleNet Success (%) | Time | MobileNet-v2 Success (%) | Time |
|------------|-----------------------|------|---------------------|------|-----------------------|------|--------------------------|------|
| AdvGAN     | 98.20                 | 17.32 s | 95.72               | 16.39 s | 98.16               | 17.99 s | 97.77                   | 18.32 s |
| TREMBA     | 99.52                 | 437.7 s | 100.0               | 316.9 s | 99.85               | 280.2 s | 100.0                   | 144.8 s |
| ColorFool  | 89.72                 | 26.98 h | 91.25               | 24.50 h | 85.29               | 29.88 h | 89.23                   | 30.93 h |
| SparseAdv  | 97.26                 | 12.22 s | 96.32               | 13.58 s | 99.02               | 14.78 s | 95.35                   | 12.64 s |

*The four compared algorithms include AdvGAN, TREMBA, ColorFool, and our SparseAdv. There are two performance metrics containing attack successful rate (%) and time of batch (1000) generation.*
When $\rho$ is equal to $b\rho_j$, $\text{KL}(\rho \| \hat{\rho}_j)$ is equal to 0 and its value increases monotonically as the difference between $\rho$ and $b\rho_j$ increases. Therefore, minimizing the penalty $\text{KL}(\rho \| \hat{\rho}_j)$ has the effect of bringing $b\rho_j$ closer to $\rho$, achieving sparsity limitation on the neural network.

3.5. Operations on Different Color Channels. We propose to choose a map from one color channel to be strongly disturbed and add weak perturbation on maps from other two channels. Exploration of biological characteristic of human eye shows that visual cones have different sensitivities to distinguished colors including red, green, and blue, according to the research of Dartnall et al. [27]. Cicerone et al. [28] measured the number of visual cones and found that the ratio of red-sensitive, green-sensitive, and blue-sensitive cones is approximately 40:20:1. Such findings indicate that human eyes are least sensitive to blue. However, convolutional neural networks (CNN) multiply the same weights of maps from three channels during feature extraction. Motivated by this finding, map from $B$ channel is chosen to be added relatively strong perturbation since the sensitivity of blue color to human visualization is the lowest while its influence as a perturbation on CNN is equally of importance as the rest of two channels. Maps from the two other channels are perturbed by weaker perturbation compared with those added on maps from blue color channel. We implement the above methodology, attempting to achieve better visualization effects without confining pixel values of the adversarial examples. To verify the effectiveness of our color-channel-based operation, three groups of adversarial examples are generated, with each group containing three generated examples being strongly perturbed on $R$, $G$, and $B$ channels, respectively. The adversarial examples are shown in Figure 4.

4. Experiments and Evaluations

4.1. Experimental Setup. The performance of SparseAdv on four networks: ResNet-50 [30], VGG-16 [31], GoogleNet [32], and MobileNet-v2 [33], was evaluated on dataset ImageNet-1000 [34]. Various experiments were set for different tasks: (1) for algorithm efficiency evaluation, SparseAdv was compared to AdvGAN [10], TREMBA [17], and ColorFool [24] in terms of generation efficiency of adversarial examples; (2) a transfer matrix was demonstrated to show transferability among the four networks (ResNet-50, VGG-
4.3. Transferability Evaluation. White-box attack was and are the most similar to the original images. realized better visualization through our special perturbation understood by the four algorithms are shown in Figure 5. It can be seen from correctly classification (i.e., record the time of generating 1000 adversarial-samples) were used for evaluating.

50, VGG-16, GoogleNet, and MobileNet-v2) and adversarial conduct on each one of the four source models (ResNet-16, GoogleNet, and MobileNet-v2) and other networks including AlexNet [2], DenseNet [35], ResNet-152 [30], and ResNet-34 [30]; (3) SparseAdv was compared to AdvGAN [10], TREMBA [17], and ColorFool [24] to demonstrate its performance on reducing visual sense. Additionally, key parameters of the compared experiments are the default given optimal parameters by the previous researches as demonstrated in the literature for fair comparison.

Two indicators including attack successful rate (i.e., statistics the ratio of examples that are misclassified after attack from correctly classified before) and time of batch generation (i.e., record the time of generating 1000 adversarial-samples) were used for evaluating.

4.2. Comparison Experiments among Algorithms

4.2.1. Algorithm Efficiency. We set 1000 samples as a batch and compared the sample generating time by different algorithms. As shown in Table 1, SparseAdv achieves relatively high attack successful rate among the comparison results, whereas time consumption of each algorithm for generating adversarial examples demonstrates that SparseAdv saves considerable time compared to other algorithms.

4.2.2. Visual Comparisons. Several adversarial examples generated by the four algorithms are shown in Figure 5. It can be seen that those adversarial examples look similar, but SparseAdv realizes better visualization through our special perturbation adding approach. Compared to the rest of the adversarial attack algorithms, ours have finer-grained scratch at the background and are the most similar to the original images.

4.3. Transferability Evaluation. White-box attack was firstly conducted on each one of the four source models (ResNet-50, VGG-16, GoogleNet, and MobileNet-v2) and adversarial attack was transferred to the rest of the three models to evaluate the transferability of the proposed algorithm. The same attack was then conducted simultaneously on the four models. That is, the average of their four loss functions was used as an overall loss function for updating sparse mapping network to generate universal adversarial examples. We used those universal adversarial examples to test their transferability on the four target classifiers, and further tested their transferability on other target classifiers (AlexNet, DenseNet, ResNet-152, and ResNet-34). As shown in Table 2, the first four models in the left column are source models. And the four models in the top row are target models. Attack successful rate is used as transferability metric. As can be seen in the four rows of the table, values on the diagonal line from the upper left to the lower right are higher than the others. While the target model is different from the source model, attack successful rate drops down to different degrees. Structures and parameters of a neural network are formulated from the feedback of its loss function [7]. The descendent and converge of the loss function guide the search to perturbation distributions. So, the searched perturbation distributions have strong correlation with such network’s structure. When adversarial examples transfer among target models which have similar structures, they tend to be more transferable, and vice versa. That is the reason why adversarial examples’ transferability shown in Table 2 is unstable.

Results in Table 3 reveal that transferability of the generated universal adversarial examples are more stable and higher. Because the universal adversarial examples are generated by the network which is used to simultaneously attack the four networks, they would then contain general information related to the four models’ structures. Thus, the perturbation distributions would contain general structure information for more other models with higher probability, therefore easier to be transferable to more models which share this general structure information. Experimental results demonstrate that the four target models could be attacked with successful rate of no less than 95%. In addition, more neural networks with similar structure of all four target models could be successfully attacked by the universal adversarial examples, with a successful rate of no less than 75%.

5. Conclusion

In this paper, the purpose is to realize adversarial attack with better balance among attack ability, less time consumption, and more imperceptible visual result. Searching space is enlarged to obtain adversarial information, which could be used as the adversarial perturbation of the input example. A sparse mapping network is trained for fast adversarial perturbation generation and save considerable time than the traditional adversarial attack algorithms which need significant iterations for updating each input with pixels. To improve searching efficiency, KLD loss is introduced to selectively suppress unnecessary neurons, thus effectively reduces computation complexity. Based on eye sensitivity of different color channels, we add stronger perturbations on the B channel and add relatively weak noise on R and G channels to minimize the modifications under relatively eye-sensitive channels, for the goal of reducing the visual perception of the perturbations. SparseAdv could search universal adversarial information that contains general structure information for more models. Thus, the adversarial examples generated by those universal adversarial information have relatively high transferability. Experimental results show that the proposed SparseAdv spends 13.30 s on average, for generating 1000 adversarial examples, with an average successful rate of 96.99%, against four target deep learning models.

Table 3: Transferability of universal adversarial examples.

| Universal adversarial example | MobileNet-v2 | ResNet-50 | VGG-16 | GoogleNet | AlexNet | DenseNet | ResNet-152 | ResNet-34 |
|--------------------------------|-------------|-----------|--------|-----------|---------|----------|------------|----------|
|                               | 95.98       | 97.32     | 96.65  | 95.06     | 75.46   | 86.75    | 82.09      | 88.24    |

*The transferability is represented by attack successful rate (%) when the universal adversarial examples are fed into different deep learning models.
Data Availability
The python data used to support the findings of this study have been deposited in the Github repository (https://github.com/AuroraZhu/SparseAdv.git).

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

Acknowledgments
This work was supported in part by the National Natural Science Foundation of China under Grant 61972092, 61932014 and the Collaborative Innovation Major Project of Zhengzhou (20XTZX06013).

References
[1] D. Lu and Q. Weng, "A survey of image classification methods and techniques for improving classification performance," International Journal of Remote Sensing, vol. 28, no. 5, pp. 823–870, 2007.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," Advances in Neural Information Processing Systems, vol. 25, pp. 1097–1105, 2012.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: surpassing human-level performance on ImageNet classification," in In Proceedings of the IEEE international conference on computer vision, pp. 1026–1034, Santiago, Chile, 2015.

[4] O. Abdel-Hamid, A.-r. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional neural networks for speech recognition," IEEE/ACM Transactions on audio, speech, and language processing, vol. 22, no. 10, pp. 1533–1545, 2014.

[5] Z.-Q. Zhao, P. Zheng, S.-t. Xu, and X. Wu, "Object detection with deep learning: a review," IEEE transactions on neural networks and learning systems, vol. 30, no. 11, pp. 3212–3232, 2019.

[6] A. I. Maqueda, A. Loquercio, G. Gallego, N. García, and D. Scaramuzza, "Event-based vision meets deep learning on steering prediction for self-driving cars," in In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5419–5427, Salt Lake City, 2018.

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

[8] C. Szegedy, W. Zaremba, I. Sutskever et al., "Intriguing properties of neural networks," https://arxiv.org/abs/1312.6199.

[9] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in In 2017 ieee symposium on security and privacy (sp), pp. 39–57, San Jose, CA, USA, 2017.

[10] C. Xiao, B. Li, J. Y. Zhu, W. He, M. Liu, and D. Song, "Generating adversarial examples with adversarial networks," https://arxiv.org/abs/1801.02610.

[11] C.-C. Tu, P. Ting, P.-Y. Chen et al., "AutoZoom: autoencoder-based zeroth order optimization method for attacking black-box neural networks," in Proceedings of the AAAI Conference on Artificial Intelligence, pp. 742–749, Honolulu, Hawaii, USA, 2019.

[12] A. Kurakin, I. Goodfellow, and S. Bengio, “Adversarial machine learning at scale,” https://arxiv.org/abs/1611.01236.

[13] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, “Towards deep learning models resistant to adversarial attacks,” https://arxiv.org/abs/1706.06083.

[14] Y. Dong, F. Liao, T. Pang et al., "Boosting adversarial attacks with momentum," in In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 9185–9193, Salt Lake City, 2018.

[15] C. Xie, Z. Zhang, Y. Zhou et al., “Improving transferability of adversarial examples with input diversity," in In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2730–2739, Long Beach California, June 2019.

[16] Y. Dong, T. Pang, H. Su, and J. Zhu, "Escaping defenses to transferable adversarial examples by translation-invariant attacks," in In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4312–4321, Long Beach California, 2019.

[17] Z. Huang and T. Zhang, "Black-box adversarial attack with transferable model-based embedding," https://arxiv.org/abs/1911.07140.

[18] D. Wierstra, T. Schaul, T. Glasmachers, Y. Sun, J. Peters, and J. Schmidhuber, "Natural evolution strategies," The Journal of Machine Learning Research, vol. 15, no. 1, pp. 949–980, 2014.

[19] S. Jandial, P. Mangla, S. Varshney, and V. Balasubramanian, "AdvGAN++: harnessing latent layers for adversary generation," in In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops, Seoul Korea, 2019.

[20] X. Liu and C.-J. Hsieh, "Rob-GAN: generator, discriminator, and adversarial attacker," in In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 11234–11243, Long Beach California, June 2019.

[21] Z. Zhao, D. Dua, and S. Singh, "Generating natural adversarial examples," https://arxiv.org/abs/1710.11342.

[22] D. Deh, J. Zhang, and A. K. Jain, "AdvFaces: Adversarial face synthesis," https://arxiv.org/abs/1908.05008.

[23] L. Mescheder, S. Nowozin, and A. Geiger, “The numerics of GANs,” In Advances in Neural Information Processing Systems, vol. 30, 2017.

[24] A. S. Shamsabadi, R. Sanchez-Matilla, and A. Cavallaro, “ColorFool: semantic adversarial colorization,” in In IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1151–1160, 2020.

[25] A. Bhattad, M. J. Chong, K. Liang, B. Li, and D. A. Forsyth, “Unrestricted adversarial examples via semantic manipulation,” https://arxiv.org/abs/1904.06347.

[26] X. Dong, J. Han, D. Chen et al., “Robust superpixel-guided attentional adversarial attack," in In IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 12985–12994, 2020.

[27] D. HJA, B. JK, and M. JD, "Human visual pigments mesopic–long-wavelength-sensitive to middle-wavelength-sensitive photopigments from the eyes of seven persons," Proceedings of the Royal society of London. Series B. Biological sciences, vol. 220, pp. 115–130, 1983.

[28] C. M. Cicerone and J. L. Nerger, “The relative numbers of long-wavelength-sensitive to middle-wavelength-sensitive photopigments from the eyes of seven persons,” Proceedings of the Royal society of London. Series B. Biological sciences, vol. 220, pp. 115–130, 1983.
cones in the human fovea centralis,” Vision Research, vol. 29, no. 1, pp. 115–128, 1989.

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

[30] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778, Las Vegas, Nevada, USA, 2016.

[31] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” https://arxiv.org/abs/1409.1556.

[32] C. Szegedy, W. Liu, Y. Jia et al., “Going deeper with convolutions,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1–9, Boston, Massachusetts, USA, 2015.

[33] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “MobileNetV2: inverted residuals and linear bottle-necks,” in In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4510–4520, Boston, 2018.

[34] O. Russakovsky, J. Deng, H. Su et al., "ImageNet large scale visual recognition challenge," International Journal of Computer Vision, vol. 115, no. 3, pp. 211–252, 2015.

[35] G. Huang, Z. Liu, V. Laurenis, and K. Q. Weinberger, “Densely connected convolutional networks,” in In Proceedings of the IEEE conference on computer vision and pattern recognition 2017, pp. 4700–4708, Las Vegas, Nevada, USA, 2016.