QUERY-FREE EMBEDDING ATTACK AGAINST DEEP LEARNING

Yujia Liu, Weiming Zhang, Nenghai Yu

University of Science and Technology of China
yjcaihon@mail.ustc.edu.cn, {zhangwm, ynh}@ustc.edu.cn

ABSTRACT

Deep neural networks are vulnerable to *adversarial examples*, subtly perturbed images which can fool networks to output incorrect classification results. To deceive deep learning models, in this paper, instead of utilizing the weakness of networks themselves, we present *Embedding Attack*, which is to attack the common image resizing operation in the deep learning preprocessing pipeline. By this attack, adversaries can embed a small target image into a benign image to produce adversarial examples without querying the target network. When the adversarial example is resized to the required shape, the embedded target image will be recovered. We design embedding attacks for three common image resizing methods and prove that our algorithms are optimal when the target image can be fully recovered. Furthermore, we design a universal embedding attack that enables adversarial examples to work under different resizing methods.

Index Terms— Adversarial examples, image resizing, embedding attack

1. INTRODUCTION

Deep learning has led to major breakthroughs in recent years and has become a critical part of artificial intelligence. Deep neural networks (DNNs) are currently the most powerful tool in deep learning. Despite this, Szegedy *et al.* [1] first proposed the existence of subtle perturbation to an image which can fool the classifier based on deep neural networks. Adversarial examples might pose a serious threat to security-critical systems based on deep learning, such as autonomous driving [2], face recognition [3] and voice command recognition [4]. The existence of adversarial examples prompts people to design more robust deep learning models.

Many researchers have proposed various effective approaches capable of reliably constructing adversarial perturbations. They can be divided into white-box attacks and black-box attacks based on whether adversaries know the inner information of the model. Most white-box attacks need the gradient of the loss function of the network. Ian Goodfellow *et al.* [5] presented a fast and efficient method to generate adversarial examples called the Fast Gradient Sign Method (FGSM), which explores the gradient direction of the loss function and adds a perturbation with a fixed norm. Papernot *et al.* [6] proposed an iterative algorithm called Jacobian Saliency Map Attack (JSMA), which is based on the mapping between inputs and outputs of DNNs by constructing adversarial saliency maps. More interestingly, Moosavi-Dezfooli et al. [7] found that there exists a universal perturbation which can make a model misclassify all the natural images.

While in black-box attacks, adversarial examples are generated without the knowledge of the DNN, adversaries only have access to its output. By querying the model a large number of times, adversaries can train a substitute network to attack [8] or iteratively modify the input image until the model outputs the target results [9]. They usually require a large number of queries to the model to collect sufficient information for generating adversarial examples.

In this paper, instead of attacking the deep neural network itself, we concentrate on image resizing in the deep learning processing pipeline. To be more precise, we make use of the fact that input images need to be resized to the required shape before being fed into the network. We do not need to know any information about the network, except for the required input image size. Better still, we do not even need to query the target network at all. The required image sizes of the famous pre-trained networks are usually small and fixed (224×224 [10, 11, 12, 13] or 299×299 [14]) to ensure the efficiency of both training and predicting. In practical, deep learning classifiers are however likely to receive input images with various shapes, *e.g.* in the field of cloud services. Therefore, classifiers need to resize the images to the required shape first.

We present an approach to crafting query-free adversarial examples, called *Embedding Attack*. In our attack, all we need to know is the input image size. Our basic idea is to embed a target image of required size into a relatively large original image to obtain an adversarial example, which will be recognized as the original label by human eyes and will be transformed into the target image once being resized by the target DNN. Since the nearest-neighbor, bilinear and bicubic interpolation are the most popular resizing methods and are commonly used in deep learning frameworks (Tensorflow, Caffe, Torch, *etc.*), we design different embedding attacks for each of them. We further prove that our embedding algorithms are optimal under the ℓ_1-norm measurement.

Although a similar idea Downsampling Attack was pro-

978-1-5386-9552-4/19/$31.00 ©2019 IEEE
DOI 10.1109/ICME.2019.00073

posed in [15], it is limited to the situation that the original image size is an integer multiple of the required input image, and it is just for the nearest-neighbor interpolation resizing method. While our embedding attack can be applied to any resizing ratio and three resizing methods. In addition, to improve the practicality of the embedding attacks, we design a universal attack, the adversarial examples generated by which can work in any of these three different resizing methods.

Our contributions can be summarized as follows:

- We present a query-free adversarial examples generation method, which utilizes the resizing operation in the deep learning processing pipeline and needs no information about the DNN except for the input shape.
- We propose three embedding attacks against three popular resizing methods, *i.e.* nearest-neighbor, bilinear and bicubic interpolation. These methods are used in most deep learning frameworks as default or recommended resizing methods.
- Considering the situation without prior knowledge to the resizing method the DNN uses, we further show a universal attack method, which combines the above three attacks so that one adversarial example can still be successful in different resizing methods.

2. COMMON IMAGE RESIZING METHODS

Image resizing is a basic operation of image processing and is also an essential preprocessing step in deep learning frameworks. There exist various image resizing algorithms [16, 17, 18], but those used in deep learning frameworks are several traditional methods based on interpolation [15], including nearest-neighbor, bilinear and bicubic interpolation [19]. For every pixel point in the resized image, these interpolation algorithms try to find the corresponding point set in the original image. And every pixel value in the resized image is the weighted average of the corresponding pixel set. Different resizing methods have their own pixel sets and weights calculation methods.

2.1. Nearest-neighbor Interpolation Method

Nearest-neighbor interpolation is the most basic and simplest image resizing algorithm. For an original image with shape $w_o \times h_o$ and target shape $w_t \times h_t$, the resizing ratio is $(w_o/w_t, h_o/h_t)$. Assume (u_0, v_0) is a point in the resized image, its corresponding point in the original image is

$$(u, v) = (u_0 \times w_o/w_t, v_0 \times h_o/h_t). \qquad (1)$$

As shown in Fig.1 (a), $\mathcal{S} = \{(i,j), (i,j+1), (i+1,j), (i+1,j+1)\}$ are pixel values of four neighbor points of (u, v). The pixel value $p_r(u_0, v_0)$ on the resized image equals to the one of nearest point in $\mathcal{S}$.

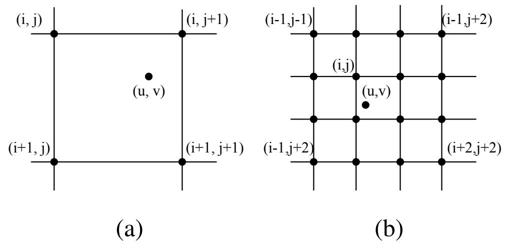

Fig. 1: (a) Nearest-neighbor / Bilinear Interpolation Resizing (b) Bicucic Interpolation Method of Image Resizing

2.2. Bilinear Interpolation Method

Bilinear interpolation considers the same four neighbor points with the nearest-neighbor, as shown in Fig 1 (a). Instead of using the nearest point, pixel values $p_r(u_0, v_0)$ in the resized image are calculated by the weighted average of all four pixel values p_o on original image:

$$\begin{aligned} p_r(u_0, v_0) =& (i+1-u)(j+1-v) \cdot p_o(i,j) \\ &+ (u-i)(j+1-v) \cdot p_o(i+1,j) \\ &+ (i+1-u)(v-j) \cdot p_o(i,j+1) \\ &+ (u-i)(v-j) \cdot p_o(i+1,j+1), \end{aligned} \qquad (2)$$

Bilinear interpolation is more complex than the nearest-neighbor method, but it does not have the shortcoming of pixel discontinuity in the resized image. It is the default image resizing method in some popular deep learning frameworks, such as Tensorflow, Caffe, and Torch.

2.3. Bicubic Interpolation Method

Bicubic interpolation method has relatively large computation overhead, but the visual effect is the best of the three. It uses sixteen neighbor points of (u, v), as shown in Fig. 1 (b). Similar to the bilinear, it calculates the weighted average of these sixteen pixels.

$$p_r(u_0, v_0) = \sum_{m=i-1}^{i+2} \sum_{n=j-1}^{j+2} W(u-m)W(v-n)p_o(m,n) \qquad (3)$$

where,

$$W(x) = \begin{cases} (a+2)\left|x\right|^3 - (a+3)\left|x\right|^2 + 1 & |x| \le 1 \\ a\left|x\right|^3 - 5a\left|x\right|^2 + 8a\left|x\right| - 4a & 1 < |x| < 2 \\ 0 & \text{otherwise} \end{cases}$$

and a is usually set to -0.5.

3. OUR EMBEDDING ATTACK ALGORITHMS

Normally, to deceive a deep network, one needs to feed a well-crafted input (known as an adversarial example), which misleads the DNN model to give a wrong answer. However, DNN is not the only element in the deep learning pipeline,

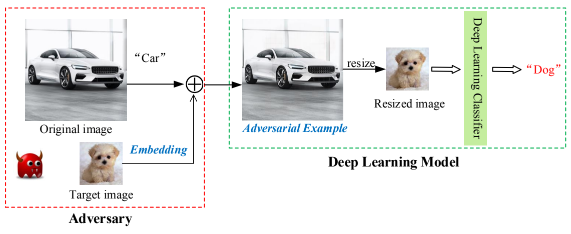

Fig. 2: Overview of the Embedding Attack

resizing operation is also a necessary preprocessing in it. As shown in Fig. 2, an adversary can apply our embedding attack to embed a target image (target label: "Dog") with the same size as the DNN input into a large-size original image (original label: "Car"). When the deep learning model receives the generated image, *i.e.* the adversarial example, it will resize it and feed the resized image into the DNN classifier. Because the resized image is our target image, the DNN classifier will output the result "Dog".

In this section, we present our algorithms in details. We design different attack algorithms aimed at different image resizing methods in deep learning frameworks.

3.1. Attack on Nearest-neighbor Interpolation

The nearest-neighbor interpolation method takes only one point in a four-point neighborhood for image resizing, regardless of the pixel values of the other three points. Therefore, we only need to replace the pixel of each chosen point in the original image with the corresponding pixel value in the target image, as shown in Fig. 3(a). In this way, we are able to embed the target image into the original image, which will turn into the target image once being resized to the target shape.

3.2. Attack on Bilinear Interpolation Method

Bilinear interpolation is the most commonly used method in image resizing, and is the default method in popular deep learning frameworks (*e.g.* Tensorflow and Torch). In Fig. 3(b), the new pixel value calculated by the bilinear interpolation method is a weighted average of all four points in the neighborhood. Therefore, different from the nearest-neighbor interpolation, when embedding the target image into the original, we need to modify four pixels.

Our main idea is to replace the pixels of every four-point neighborhood with carefully calculated values, which can guarantee that the resized image will turn into the target, and the total perturbations are the smallest. To find the optimal replacement, for each four-point neighborhood, we perform the following operations:

Step 1. Compute the resizing ratio and find the mapped point for each pixel in the target image by Eq. (1).

Step 2. Compute weights w of four pixels in the neighborhood of the mapped point by Eq. (2) and sort them.

Step 3. Compute the resized pixel p_r from the four neighbor pixels in original image, and calculate the required modification $\delta = (p_t - p_r)/w$ for four pixels in descending order by weights, where p_t is pixel in target image.

These three steps will be repeated until all the four points in the neighborhood are updated. In each iteration, the pixel with the heaviest weight will be modified first, which ensures the modification is the smallest. To be noted, this pseudo-code is shared by the attacks on bilinear and bicubic interpolation methods, as we will state in the next subsection, these two algorithms have the same processing flows, and the differences between them are the mapped points and the calculation method of corresponding weights.

3.3. Attack on Bicubic Interpolation Method

Bicubic interpolation is also provided by most image processing libraries. It has the best visual effect among these three methods. Every pixel value in the resized image depends on sixteen points (in a 4×4 neighborhood) in the original image. Similar to the bilinear interpolation, this pixel value is the weighted average of sixteen pixels. Therefore, our attack on the bicubic interpolation follows the same processing flows, so we omit the detailed steps here.

3.4. Proof of the Optimality

Our attack on the nearest-neighbor interpolation is obviously the only attack solution, so this does not need any proof. Here, we provide the proofs of the other two attacks. Because the bilinear and bicubic interpolation follow the same processing flows, we prove the optimality of them simultaneously. Due to the independence of each four(sixteen)-point neighborhood, we only need to prove the optimality of one neighborhood.

Theorem 1. *Our attacks on the bilinear and bicubic interpolation are the optimal solutions under the measurement of ℓ_1-norm.*

Proof. Assume that the neighborhood contains k pixels ($k = 4$ for bilinear interpolation and $k = 16$ for bicubic interpolation), their values are $p_0, p_1, ..., p_k$, and the corresponding weights are $w_0 \geq w_1 \geq ... \geq w_k$. The weighted average of these k points is p_r, *i.e.*

$$w_0 \cdot p_0 + w_1 \cdot p1 + \cdots + w_k \cdot p_k = p_r.$$

Assume that there exists a group of δ $(\delta_0, , \cdots, \delta_k)$, satisfying

$$w_0 \cdot (p_0 + \delta_0) + w_1 \cdot (p_1 + \delta_1) + \cdots + w_k \cdot (p_k + \delta_k) = p_t.$$

Considering $p_t > p_r$, we have $\delta_i > 0 (i = 0, 1, \cdots, k)$. Assume that $(p_0 + \delta_0) < 255$ and $\delta_j > 0 (i = 0, 1, \cdots, k)$, which means that δ_0 is not set to its upper bound.

Let ϵ satisfy $0 < \epsilon \leq \min\{255 - \delta_0, (w_j/w_0) \cdot \delta_j\}$, and $\delta_0^{'} = \delta_0 + \epsilon$ and $\delta_j^{'} = \delta_j - (w_0/w_i) \cdot \epsilon$, we have

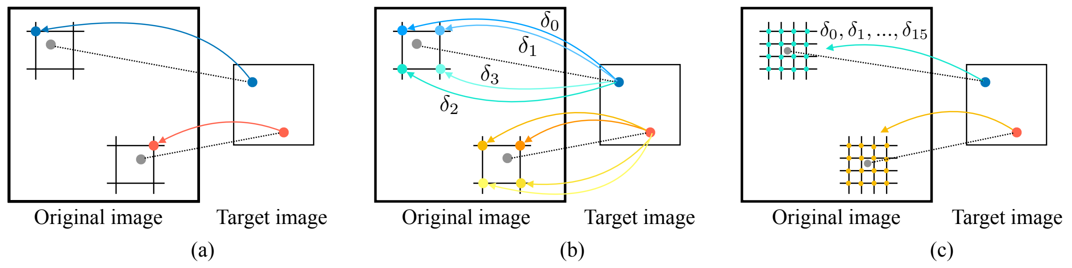

Fig. 3: Attacks on (a) Nearest Interpolation (b) Bilinear Interpolation (c) Bicubic Interpolation

$$w_0(p_0 + \delta_0^{'}) + w_j(p_j + \delta_j^{'}) + \sum_{i \neq 0,j} w_i(p_i + \delta_i)$$
$$= w_0(p_0 + \delta_0 + \epsilon) + w_j(p_j + \delta_j - \frac{w_0}{w_j} \cdot \epsilon) + \sum_{i \neq 0,j} w_i(p_i + \delta_i)$$
$$= w_0(p_0 + \delta_0) + w_j(p_j + \delta_j) + \sum_{i \neq 0,j} w_i(p_i + \delta_i) = p_t$$

We can find that $\delta_0^{'} + \delta_j^{'} = \delta_0 + \delta_j + \frac{w_j - w_0}{w_j} \cdot \epsilon < \delta_0 + \delta_j$, which means that through the above transformation, we can always find a better solution when $w_i > w_j$, and $(p_i + \delta_i) < 255$ and $\delta_j > 0$. Keep repeating the above processes, we will obtain the same solution as presented in this work. □

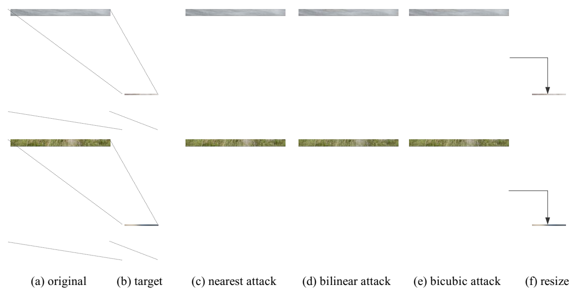

Fig. 4: Three embedding attacks on different image resizing methods. The actual sizes of the original image and the target image are $\mathbf{1300} \times \mathbf{1300}$ and $\mathbf{299} \times \mathbf{299}$ respectively. The three kinds of adversarial examples can all completely turn into the target image after being resized.

3.5. Universal Embedding Attack

In some situations, we do not know which resizing method that the deep learning classifier uses. So we design a universal attack that enables the adversarial examples to work in any of these three resizing methods. The idea is embedding the target image to the original with the above three algorithms one by one. The nearest-neighbor, bilinear, and bicubic attack require respectively one, four, and sixteen points for each neighborhood. And the point required by the nearest-neighbor attack is included by the bilinear, and the points required by the bilinear are included by the bicubic. According to this feature, we design this universal attack, the embedding steps are as follows:

Step 1. Compute the resizing ratio and find the mapped point for each pixel in the target image by Eq. (1).

Step 2. Compute weights w of sixteen pixels in the neighborhood of the mapped point by Eq. (2) and sort them in descending order. Then, replace the biggest weight with 1.0.

Step 3. Run bicubic embedding attack with current weights.

In Step 2, the point with the biggest weight is the mapped point in nearest-neighbor interpolation, and because we replace its weight by 1.0, the pixel value of this point will be substituted by the target in Step 3, and this is actually the nearest-neighbor embedding attack. Note that only the nearest-neighbor and bicubic embedding attacks are performed in Step 3, the reason why the bilinear embedding attack is not included is that it has similar weights distribution with the bicubic attack.

The adversarial examples generated by the universal attack can fool the DNN model with any of the three resizing methods. Except for the nearest-neighbor resizing method, the adversarial examples cannot be recovered to the target image totally, but they are enough to fool the DNN model.

4. EXPERIMENTS

In this section, we illustrate the effectiveness of the embedding attacks on nearest-neighbor, bilinear, bicubic interpolation, as well as the universal embedding attack on them all. The deep neural network we use in our experiments is Inception-v3 [14] with the required input size 299 × 299 and all the implementation is based on Tensorflow.

4.1. Effectiveness of the Embedding Attack

First, we show the effectiveness of the three embedding attacks through several samples in Fig. 4. For a given original image, we construct three adversarial examples with a required resizing ratio respectively by carrying out the nearest-neighbor, bilinear, and bicubic embedding attacks. And we resize them with the image resizing method which is consistent with the attack. All of the three adversarial examples can

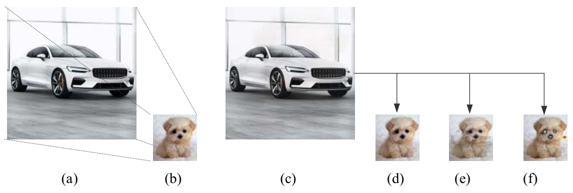

Fig. 5: Universal embedding attack and the resized results with different resizing methods: (a) original image (b) target image (c) adversarial example (d) nearest resized (e) bilinear resized (f) bicubic resized.

be resized into the target image. In general, before attacking the target model, we only need to know its required input size. Then we carry out the corresponding embedding attack, and the adversarial example can successfully fool the classifier after the necessary resizing operation in the deep learning model. In the embedding attack, as long as the target image itself can be classified correctly as the given target class by the classifier, the success rate will be 100%.

Then we illustrate the effect of the universal embedding attack, as shown in Fig. 5. We construct a universal adversarial example based on a large-size original image, and we can see that its visual quality is no worse than the non-universal adversarial examples in Fig. 4. We next resize it with the three image resizing methods. Although the target image cannot be fully recovered, the test result shows that the resized images are enough to fool the model to classify them into the target class. The visual effect is not important as long as it can reach the adversarial purpose.

4.2. Visual Quality

The visual quality of the adversarial examples generated by embedding attacks is greatly affected by the resizing ratio of the original image and the target image, as shown in Fig. 6. We use structural similarity index (SSIM) [20] to evaluate the visual quality of the processed images. We can see that, in all the four attacks, the SSIM increase with the resizing scale. This can be explained by the principle of interpolation-based resizing methods. In all of these methods, every point in the resized image requires to modify up to sixteen pixels in the original image, and has nothing to do with other pixels. Similarly, in our embedding attack, for each point of the target image, we only modify at most sixteen points in the original image. In other words, the number of modified points will not exceed sixteen times the number of points in the target image. Therefore, when the specified size of the model is fixed, the larger the size of the original image, the smaller the proportion of modification points, which results in better visual quality. An example is shown in Fig. 7.

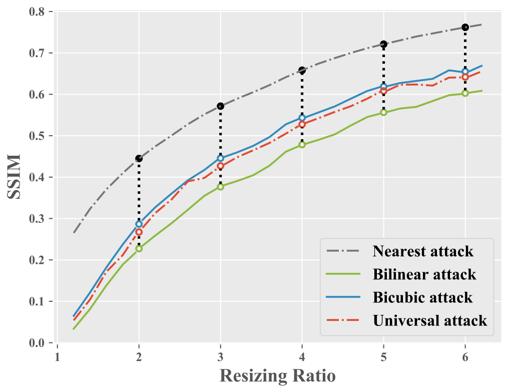

Fig. 6: SSIM of different attacks with different resizing ratios.

5. DISCUSSIONS

In general, a targeted attack means generating an adversarial example belonging to a given class while similar to the original image. But in our embedding attacks, the target image is independent of the original image. So we are free to choose the target image, as long as it belongs to the target class.

In order to ensure the visual quality of the generated adversarial examples, we should select the target image as similar as possible to the original image, especially in the aspect of the global tune and the location of areas with the complex texture. From the experiment figures, we can still see some modified traces in the result when the size of the original image is not very large. So one of our further researches will be how to improve the visual quality of the adversarial examples.

In addition, there might be some possible defense methods against our embedding attacks. First, before an image is resized to the required input shape of a deep learning model, we can preprocess it with some methods like Gaussian filter, which will reduce the impact of our modifications on the image resizing results. Another possible defense is deploying other image resizing methods that are not based on interpolation in deep learning frameworks. Besides, we can also design deep neural networks that can handle images with any shape.

6. CONCLUSIONS

The existence of adversarial examples proves that deep neural network itself is not robust to the input. Before an input is fed into a network, image resizing is a necessary operation in the deep learning pipeline. So not only the DNN itself can be deceived, but the preprocessing operations can also be attacked.

In this paper, we propose the embedding attacks on three common image resizing methods in most deep learning frameworks. By these attacks, for any large-size original image, we can construct high visual quality adversarial examples without querying the target model or knowing its inner information. When the adversarial example is resized

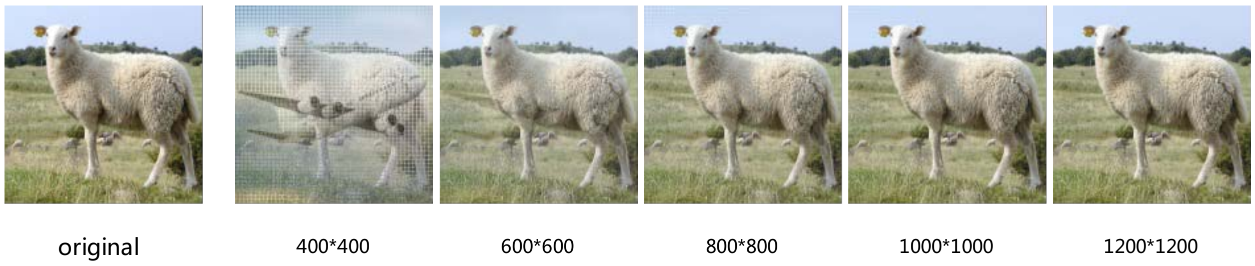

Fig. 7: Visual quality in case of different original image sizes. The embedded target image size is fixed $\mathbf{299} \times \mathbf{299}$. and the original image sizes vary from $\mathbf{400} \times \mathbf{400}$ to $\mathbf{1200} \times \mathbf{1200}$. In order to facilitate the observation and comparison, we process them into the same size to show the visual effects.

to the required size before entering the classifier, it will be transformed into the target image which belongs to the target class. Besides, we theoretically prove that our embedding method is optimal when the target image can be fully recovered. Furthermore, in order to improve the practicality of our attack, we design a universal embedding algorithm, which is able to attack all the three common image resizing methods.

7. REFERENCES

[1] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, et al., "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[2] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, et al., "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[3] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, et al., "Deep face recognition.," in *BMVC*, 2015.

[4] Geoffrey Hinton, Li Deng, Dong Yu, et al., "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, 2012.

[5] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[6] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, et al., "The limitations of deep learning in adversarial settings," in *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*. IEEE, 2016.

[7] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard, "Universal adversarial perturbations," *arXiv preprint*, 2017.

[8] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow, "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples," *arXiv preprint arXiv:1605.07277*, 2016.

[9] Nina Narodytska and Shiva Prasad Kasiviswanathan, "Simple black-box adversarial perturbations for deep networks," *arXiv preprint arXiv:1612.06299*, 2016.

[10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012.

[11] Christian Szegedy, Wei Liu, Yangqing Jia, et al., "Going deeper with convolutions," Cvpr, 2015.

[12] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[14] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[15] Qixue Xiao, Kang Li, Deyue Zhang, and Yier Jin, "Wolf in sheep's clothing-the downscaling attack against deep learning applications," *arXiv preprint arXiv:1712.07805*, 2017.

[16] Chulhee Lee, Murray Eden, and Michael Unser, "High-quality image resizing using oblique projection operators," *IEEE Transactions on Image Processing*, 1998.

[17] Arrate Muñoz, Thierry Blu, and Michael Unser, "Least-squares image resizing using finite differences," *IEEE Transactions on Image Processing*, 2001.

[18] Daisuke Sekiwa and Akira Taguchi, “Enlargement of digital images by using multi-neural networks,” *Electronics and Communications in Japan*, 2001.

[19] J Anthony Parker, Robert V Kenyon, and Donald E Troxel, “Comparison of interpolating methods for image resampling,” *IEEE Transactions on medical imaging*, 1983.

[20] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli, “Image quality assessment: from error visibility to structural similarity,” *IEEE transactions on image processing*, 2004.