AEPecker: $L_0$ Adversarial Examples are not Strong Enough

Fei Zuo, Bokai Yang, Xiaopeng Li, Lannan Luo, Qiang Zeng
University of South Carolina
{fzuo, bokai, xl4}@email.sc.edu, {lluo, zeng1}@cse.sc.edu

Abstract

Despite the great achievements made by neural networks on tasks such as image classification, they are brittle and vulnerable to adversarial examples (AEs). By adding adversarial noise to input images, adversarial examples can be crafted to mislead neural network based image classifiers. One type of AE attack in particular, known as an $L_0$ AE, has been used in several notable real-world incidents. Our observation is that, while $L_0$ corruptions modify as few pixels as possible, they tend to cause large-amplitude perturbations to the modified pixels. We consider this to be an inherent limitation of $L_0$ AEs which can be exploited. To show the weakness of $L_0$ AEs, we thwart samples of these attacks by both detecting and rectifying them. The main novelty of the proposed detector is that we convert the AE detection problem into an image comparison problem by exploiting the inherent characteristics of $L_0$ AEs. More concretely, given an image $I$, it is pre-processed to obtain another image $I'$. We use a Siamese network which is known to be effective in comparison, to take $I$ and $I'$ as the input pair. A well-trained Siamese network can automatically capture the discrepancy between $I$ and $I'$ to detect $L_0$ noises. In addition, the straightforward pre-processor based on heuristics can be deployed as an effective defense, having a high probability of removing the adversarial influence of $L_0$ perturbations. The proposed technique shows not only a high accuracy but also a resilience to the adaptive adversary, which outperforms other state-of-the-art methods. We accordingly argue that $L_0$ attacks are not strong enough.

1 Introduction

Recent years have witnessed tremendous success of DNNs in a variety of fields, especially for vision-related tasks such as object detection [33], motion tracking [37], and face recognition [31, 38]. Despite these great achievements, they are vulnerable to adversarial examples (AEs). Szegedy et al. [35] analyze the robustness of DNNs when facing adversarial attacks, and show that deep learning systems are sensitive to small adversarial perturbations. A DNN based classifier thus can be misled by AEs and generate incorrect classification results. Many AE generation methods have been proposed and multiple off-the-shelf tools are available [7, 13, 19, 29]. As DNN based techniques are increasingly applied, countermeasures against AEs become particularly important.

In order to prevent those adversarial perturbations from being perceived by human eyes, the intentional modification on that original image must be subtle. To quantitatively describe this kind of subtle modification in math, $L_p$ norms are usually used to measure the discrepancy between an original benign image $I_o$ and its corresponding AE $I_a$. According to the value of $p$, the mainstream AE generation algorithms can be categorized into three families, i.e., $L_0$, $L_2$ and $L_\infty$ attacks. Informally, $L_0$ measures the number of pixels that are modified when comparing $I_o$ and $I_a$. $L_2$ measures the Euclidean distance between $I_o$ and $I_a$, and $L_\infty$ measures the largest discrepancy between corresponding pixels in $I_o$ and $I_a$.

Our work focuses on $L_0$ AEs. A series of notable real-world AE attacks can be modeled as $L_0$ attacks; examples include deliberate graffiti on road signs [12], and maliciously crafted eyeglasses that mislead face recognition [34]. Generally, physical modifications of images and insertion of clips into videos or audio can be regarded as $L_0$ perturbations, which are all attacks that can be introduced into the physical world.

There exist some scenarios in real-life where human interactions are not taken into account such as driverless cars. In such a scenario, a traffic sign which has been maliciously altered may easily fool the vision system of an autonomous vehicle and could cause a deadly accident. To defeat the attack based on AEs, both detection and defensive techniques attract the research community’s attention. Given an input image, the detection system outputs whether it is an $L_0$ AE so that the target DNN can reject those adversarial inputs; By contrast, a defense technique is more aggressive. Given an $L_0$ AE, the defense system helps the target DNN make correct prediction by either rectifying the AE or fortifying the classifier itself.

Some AE detection methods [3, 21, 27] and defense techniques [10, 22, 39] have been proposed. Although these ap-
proaches are able to distinguish some AEs from the legitimate images or even rectify the AEs generated by other attacks at a high success rate, they either fail for $L_0$ AEs or omit discussing them. One powerful detection method, called feature squeezing [40] is capable of detecting $L_0$ AEs. However, it relies heavily on a powerful feature squeezer and an ensemble decision maker. He et al. [17] also have shown that feature squeezers, either single or joint, are not resilient to adaptive adversary. Moreover, previous work [21] has already shown that it is challenging to recover the correct classification of $L_0$ AEs by input transformation, as “it is very difficult to properly reduce the effect of the heavy perturbation”.

We identify two inherent characteristics of $L_0$ AEs which can be exploited to thwart such kind of attack. We show that by considering these two factors, a detector with a simple architecture is still able to achieve a high detection rate, where, as a side effect, the deployed pre-processor based on straightforward heuristics effectively rectifies those AEs. More importantly, the proposed method is resilient to adaptive adversary.

- Our first observation is that $L_0$ attacks limit the number of modified pixels, but not the amplitude of pixels. Thus, $L_0$ attacks tend to introduce large-amplitude perturbations, especially for targeted attacks intended to achieve a particular designated output from a DNN.

- Secondly, as $L_0$ attacks try to modify as few pixels as possible, the altered pixels only occupy a tiny proportion of the whole image. Those altered pixels are likely to distribute in a scattered manner. In other words, those corrupted parts are mainly small and isolated regions.

We consider the high-amplitude perturbations as an inherent limitation of $L_0$ AEs, and accordingly propose a novel AE detection method. Given an image $I$, it is manipulated by a pre-processor to obtain another image $I'$. The main novelty is that we then convert the AE detection problem into an image comparison problem using a Siamese network [6], which is known to be powerful in comparison. The proposed Siamese network is target model independent and only has a simple architecture with shallow layers, which takes $I$ and $I'$ as the input pair, and can automatically capture the discrepancy between the two inputs as features for detecting $L_0$ AEs.

Moreover, the proposed detection method does not depend on a powerful pre-processor to cause prediction inconsistency. Despite all this, a pre-processor with straightforward heuristics can also be used as an effective defense which is capable of cleaning adversarial perturbations in $L_0$ AEs at high probability. Specifically, we propose an inpainting-based algorithm to process images, where inpainting refers to the process of reconstructing the lost or corrupted parts of an image. Since those heavy perturbations are very difficult to eliminate with the neighborhood filter, we instead consider those pixels as lost or deteriorated parts of images and use the inpainting technique to directly restore those small defects.

Figure 1: Framework of the system. If $I$ is detected as an $L_0$ AE, then $I'$ will replace $I$ for further classifying.

We have implemented a system AEPecker, the architecture of which is shown as Figure 1. After inputting an image $I$ to a pre-processor $P$, we will obtain another image $I'$. Then the trained Siamese network will predict whether $I$ is adversarial or not based on the input image pair $(I, I')$. If $I$ is detected as an $L_0$ AE, then we regard $I'$ as a rectified image and use it to replace $I$. We have evaluated our system on its detection and defense capability using the popular image datasets CIFAR-10 and MNIST. There are two leading $L_0$ AE generation methods: JSMA [29] and CW-$L_0$ [7]. For CIFAR-10, the evaluation results show that (1) the detection rate on the CW-$L_0$ and JSMA attack is 97.1% and 99.7% respectively, both with a low false positive rate; (2) the proposed system has outstanding transferability considering such detector trained with only JSMA AEs can accurately detect AEs generated by CW-$L_0$ with a high detection rate of 99.4%, and vice versa; (3) our defense methods recover the classification accuracy from 0% (when classifying those successful AEs without using our defense method) to 87.3% for CW-$L_0$, and from 0% to 96.1% for JSMA, and meanwhile, have very small impact on benign images. We can also observe similar test results on MNIST, with more details in the following experiment section.

The key contributions of our work include:

- We point out the inherent characteristics of $L_0$ AEs, which typically contain high-amplitude perturbations to very few and isolated pixels, and propose to exploit the limitation to develop detection and defense techniques. In particular, by considering this limitation, our system shows a resilience to adaptive adversary.

- We convert the $L_0$ AE detection problem into an image comparison problem, and propose to use a Siamese network to automatically extract the discrepancy of the input pair as features to detect AEs. Its effectiveness does not rely on a powerful image pre-processing method, and achieves a very high accuracy.

- We propose an effective inpainting-based defense against $L_0$ perturbations, which can recover the correct classification at a high probability. Although the defense method only takes advantage of straightforward heuristics, it achieves the highest accuracy when dealing with $L_0$ noise, to the best of our knowledge.

The rest of the paper is organized as follows. First, we
briefly introduce several representative $L_0$ AE generation methods in Section 2. Section 3 describes our system architecture containing both the proposed Siamese detector and novel defense approach in detail. Then, we evaluate our work and present the experimental design, configuration and results in Section 4. We also empirically analyze how the adaptive adversary may influence the proposed technique in Section 5. After that, related works are reviewed in Section 6. We finally discuss the limitations of our work and draw conclusions in Section 7 and 8.

2 Adversarial Examples Generation

Adversarial examples (AEs) are sophisticatedly crafted input samples intended to fool artificial intelligence systems. The term adversarial example can be formally defined as following. For a pre-trained DNN $f$, let $x$ be an original image. An adversarial example $x_{\text{adv}}$ is such an intentionally designed input by attackers which can guide the model $f$ to make an incorrect prediction. Moreover, to hide the adversarial perturbation, the generation of $x_{\text{adv}}$ is equivalent to solve the following constrained optimization problem:

$$\min_{x_{\text{adv}}} ||x_{\text{adv}} - x||$$

s.t. $f(x_{\text{adv}}) = \tilde{y}$

$f(x) = y$

$y \neq \tilde{y}$

where $y$ and $\tilde{y}$ are respectively the prediction results of feeding $x$ and $x_{\text{adv}}$ to $f$. Depending on the manner of how $\tilde{y}$ misleads a pre-trained classifier, adversarial attacks to DNNs can be categorized as either targeted or non-targeted. The aim of non-targeted attacks is to make the image be classified as any arbitrary class except the true one. By contrast, in targeted attacks the prediction result will be misguided to a specific class which is different from the correct one.

In this study, we will focus on the discussion of $L_0$ AE attacks, where JSMA and CW-$L_0$ are two widely used and representative $L_0$ AE generation methods. Next, we will describe these AE generation methods briefly.

2.1 Jacobian Saliency Map Attack (JSMA)

The JSMA is a targeted attack based on a greedy iterative idea proposed by Papernot et al. [29]. It only updates a limited number of pixels in an original image. To determine which pixels will be manipulated, it calculates an adversarial saliency score for each pixel, and manipulates the pixels that have high adversarial saliency scores, which have more impact on misleading the target model to predict a specific label desired by attackers. The adversarial saliency score for each pixel is calculated as:

$$x_{\text{adv}}^{i,j} = x_{i,j} + \begin{cases} 0, & \text{if } \frac{\partial f_i(x)}{\partial x_j} < 0 \text{ or } \sum_{j \neq i} \frac{\partial f_j(x)}{\partial x_j} > 0 \\ \delta, & \text{otherwise} \end{cases}$$

where $i$ denotes the $i$th pixel in the image, and $f_j$ is the prediction value of the neuron $j$ in the target model’s output layer.

2.2 Carlini & Wagner Attack (CW)

CWs are a group of targeted AE generation methods developed by Carlini and Wagner [7]. The authors claim that their method outperforms other AE generation methods since the perturbation introduced by a CW are minimized.

Given a target model $f$, after introducing some small change $\delta$ to an image $x$, the objective is to guide $f(x + \delta) = t$, where $t$ is a target class desired by attackers. Then, the authors define an objective function $g$, such that $f(x + \delta) = t$ if and only if $g(x + \delta) \leq 0$. CW attacks attempt to find $\delta$ that solves:

$$\min ||\delta||_p + c \cdot g(x + \delta)$$

where $c > 0$ is a constant that adjusts the relative weighting between perturbations and misclassification, and $|| \cdot ||_p$ is the $L_p$ distance. There are three types of CW attacks determined by the different distance metrics they used: $L_0$, $L_2$ and $L_\infty$-norm. In this paper, we focus on the CW-$L_0$ attack.

3 System Design

The proposed system consists of a Siamese network (Section 3.2) which determines whether an input image is an $L_0$ AE, and a pre-processor (Section 3.1) which also can be used as a defense component to correct the classification under the existence of $L_0$ AEs. Note that as the pre-processor has a very small impact on the classification of benign images, it can be used as a defense component independently without relying on detection.

3.1 Pre-processor

The pre-processor adopted in our system is designed to reduce adversarial noises while preserving the features in images to reduce false positives. From this perspective, the proposed pre-processor also can be deployed as a defense against $L_0$ attacks.

Intuitively, failing to limit the amplitude of those altered pixels in the images will result in outlier pixels. Previous work [21] emphasizes that it is challenging to get rid of the effect of those heavy perturbations. However, we argue the outlier pixels can be fixed by applying a processor based on inpainting. In image processing, the term “inpainting” refers
to the process of reconstructing lost or corrupted regions of image data (or to remove small defects). Our idea is to treat those outliers as small corrupted regions, and the inpainting technique exactly meets the need for eliminating the $L_0$ noise.

In detail, we observe that those $L_0$ perturbations manifest themselves visually as salient noises. A mask to determine which pixels should be reconstructed can help isolate these cases. By inspecting the pixel intensity in different color channels (e.g., the R, G, B channels for color images), it is highly possible that for each altered pixel, one extreme value in at least one channel can be achieved. We define a value as extreme if it is either smaller than an upper bound $\alpha$ or larger than an lower bound $\beta$. Thus, to obtain such a mask, we first locate all pixels of which the intensity are exceptional at least one channel. Meanwhile, we noticed that such pixels that achieve extreme values in all of the three channels are often the bright parts such as the sky in a natural image. Therefore, we use a parameter $\gamma$ to help filter out such pixels in color images. According to our observation, we choose $\gamma = 0.7$ as an empirical value. Lines 4-10 in Algorithm 1 show the procedures to initially create the mask.

In addition, considering that the number of altered pixels only occupies a small portion of the image, the possibility that most of the altered pixels will assemble to form a connected region is very low. Consequently, to further exclude those unlikely candidates, we will remove those relatively large connected regions from the mask. Specifically, we use a structuring element $E$ to describe a connected region with the specified size and shape. If a connected region is larger than $E$, we will exclude such region from the mask, as Lines 11-13 in Algorithm 1 show, where $N(\cdot)$ denotes a connected neighborhood.

We thus independently produce an inpainting mask for each channel of a color image. We then take advantage of the inpainting method proposed in [36] to restore those deteriorated pixels for each channel, as Lines 14-15 in Algorithm 1 show. Figure 2 displays some concrete examples applying Algorithm 1 with $\alpha = 0.2$, $\beta = 0.8$ on CIFAR-10. The resultant images in the even numbered row show that the adversarial perturbations are almost completely eliminated. We will provide more detailed experimental results to demonstrate how our defense influences the effectiveness of $L_0$ attacks in the next Section.

The algorithm for gray images is very similar to this, but we only need to consider one channel rather than three. In other words, we can regard the algorithm for gray image as a special case of the counterpart for color image.

### 3.2 Siamese Network Based Detector

As a classic category of neural network architecture, Siamese networks [6] are widely applied among those tasks that involve detecting similarities or other relationships between two or more comparable things. Overall, a Siamese network contains two sub-networks which share one identical architecture with the exact same weights.

Given an input image $I$, when pre-processing is adopted, the input image $I$ and the pre-processed one $I'$ may be very different even if $I$ is benign. On the other hand, the discrepancy between $I$ and $I'$ may not be simply described using a single value and compared with a threshold, as adopted by feature squeezing [40]. These are the main challenges in devising an

---

**Algorithm 1: The Pre-processor based on Inpainting.**

**Input:** A color image $I$; an upper bound $\alpha$ to describe small values; a lower bound $\beta$ to describe large values; a parameter $\gamma$ to describe bright pixels in natural images; a structuring element $E$ of the specified size and shape.

**Output:** A processed color image, denoted by $S$.

```plaintext
1. Normalize $I \leftarrow \left[\min(I) - I\right] / \left[\min(I) - \max(I)\right]$;
2. Extract three channels ($I^R$, $I^G$, $I^B$) from $I$;
3. Initialize the masks $M^R$, $M^G$, $M^B \leftarrow \{0\}$;
4. for each pixel $I_i \in I$, do
   5. if ($I_i^R < \alpha) \lor ([I_i^R > \beta) \land (I_i^G \leq \gamma \lor I_i^B \leq \gamma)]$ then
      6. $M_i^R \leftarrow 1$;
   7. if ($I_i^G < \alpha) \lor ([I_i^G > \beta) \land (I_i^R \leq \gamma \lor I_i^B \leq \gamma)]$ then
      8. $M_i^G \leftarrow 1$;
   9. if ($I_i^B < \alpha) \lor ([I_i^B > \beta) \land (I_i^R \leq \gamma \lor I_i^G \leq \gamma)]$ then
      10. $M_i^B \leftarrow 1$;
6. for each pixel $M_i^\chi = M^\chi$, where $\chi = R, G, B$, do
11. if $\exists N(M_i^\chi) > E$, s.t. $M_i^\chi = 1 \land M_i^\chi \in N(M_i^\chi)$ then
   12. $M_i^\chi \leftarrow 0$;
13. for each $\chi = I^R, I^G, I^B$, do
14. $S^\chi \leftarrow$ Inpainting $\chi$ according to $M^\chi$;
15. Reconstruct $S$ with $S^R$, $S^G$ and $S^B$;
16. return $S$.
```
Figure 3: Training phrase of the AE detector based on a Siamese network.

We propose a Siamese based $L_0$ AE detector with the help of a pre-processor, which converts the AE detection problem into an image comparison problem. Once the model with fine tuned weights is established (via training), the discrepancy between $I$ and $I'$ can be extracted by the Siamese network. Taking the discrepancies as features, the model can predict whether the input image is adversarial or not.

In detail, the shallow-layered and CNN based sub-network employed in the Siamese detector possesses the architecture as following:

$$
\text{CNN} : \rightarrow \text{conv}(3 \times 3, 64) \rightarrow \text{ReLU} \\
\rightarrow \text{conv}(3 \times 3, 64) \rightarrow \text{ReLU} \rightarrow \text{dropout}(0.3) \\
\rightarrow \text{maxpool}(2 \times 2) \rightarrow \text{dropout}(0.3) \\
\rightarrow \text{Flatten} \\
\rightarrow \text{linear}([-128] \rightarrow \text{ReLU} \rightarrow \text{dropout}(0.5) \\
\rightarrow \text{linear}(128, 10) \rightarrow \text{softmax}.
$$

Figure 3 elaborates the training phrase of the proposed detector based on a Siamese network. Given an image $I$ and its pre-processed version $I'$, the training takes $\langle I, I' \rangle$ as the input pair, where the label is 0 if $I$ is not an AE (denoted as $I_o$) or 1 if $I$ is an AE (denoted as $I_a$). Our insight is that the discrepancy between the input pair $\langle I_o, I_a \rangle$ and the consistency between $\langle I_o, I'_a \rangle$ are too complex to describe using a formula, while the Siamese network is effective in learning the complexity. Moreover, the consistency and discrepancy can be learned even when a non-powerful pre-processor is adopted, such as a bit depth reducer. The result of the last layer of each of the two sub-networks is fed to a contrastive loss function [15]:

$$(1 - Y) \frac{1}{2} (D_W)^2 + (Y) \frac{1}{2} (\max(0, m - D_W))^2$$

where $D_W$ is defined as the Euclidean distance between the outputs of the two sub-networks, $Y$ is a binary label assigned to the input pair, and $m > 0$ is a margin to define a radius around the output of one of the sub-networks. Finally, once the model is successfully trained, the Siamese network can be used to determine whether $I$ is an AE.

Our evaluation further shows that, even with a relatively small dataset and a network with very few layers, the detector achieves a very high accuracy (Section 4).

## 4 Evaluation

In this section, we evaluate our system on its detection and defense capability. We first describe the experimental settings and implementation (Section 4.1) and discuss the datasets used in our evaluation (Section 4.2). We then evaluate the effect of our pre-processing method as a defense alone (Section 4.3). Next we evaluate the accuracy of our system on detecting AEs generated by JSMA and CW-$L_0$ (Section 4.4), and the efficiency in terms of training and testing (Section 4.5).

### 4.1 Experimental Settings

**Threat model.** We assume attackers have full knowledge on a trained target image classification model, but no ability to influence that model. Thus, given a trained target model, attackers can use the $L_0$ attacks including JSMA and CW-$L_0$ to generate AEs that will be misclassified by the target model.

**Target models.** We use two popular datasets for the image classification task: MNIST and CIFAR-10. For each dataset, we build up two individual models. Specifically, for MNIST, we set up a CNN based classifier [18] and reuse the same model provided by Carlini in [7]. For CIFAR-10, we select the 32-layered ResNet model based on a residual learning framework [16] and also the model architecture given in [7]. All the target models are trained from scratch. Table 1 summarizes the classification accuracy on the testing data of each trained model. The accuracy of the two classifiers on MNIST is 99.26% and 99.52%; and the accuracy of Carlini’s classifier and the ResNet model on CIFAR-10 is 78.86% and 91.96%, respectively. Note, only those images which can be correctly classified are used to generate AEs, that is all the other images will be filtered out in the following discussion.

**Attacks.** For the target models Carlini$_M$ and Carlini$_C$, we generate AEs with CW-$L_0$ by reusing the code provided in [7]. The default parameters setting suggested by Carlini and Wagner [7] are as following: the number of maximum iterations is 1000, the initial constant is 0.001, and the largest constant is $2^6$. To compare with other state-of-the-art works [23, 40], we follow these parameters settings. Furthermore, for the target CNN and ResNet model, we generate AEs with JSMA by leveraging the Adversarial Robustness

| Dataset | Target Model | Accuracy |
|---------|--------------|----------|
| MNIST   | Carlini$_M$  | 99.26%   |
|         | CNN [18]     | 99.52%   |
| CIFAR-10| Carlini$_C$  | 78.86%   |
|         | ResNet [16]  | 91.96%   |

Table 1: Classification accuracy of the target models.
We perform the following experiments on two well-known datasets, denoted as \( \text{CWL}_0 \)-Train and \( \text{CWL}_0 \)-Test. We then use the CW-L\(_0\) algorithm to generate AEs that can successfully attack the Carlini\(_0\) model [7], and create two dis-joint datasets, denoted as \( D_{\text{C}}-\text{CWL}_0\)-Train and \( D_{\text{C}}-\text{CWL}_0\)-Test. In detail, \( D_{\text{C}}-\text{CWL}_0\)-Train contains 10,000 legitimate images and 10,000 AEs. \( D_{\text{C}}-\text{CWL}_0\)-Test contains 1,000 benign images and 1,000 AEs. Next, we follow similar method on CIFAR-10 but instead using JSMA to generate AEs based on ResNet classifier [16]. As a result, we obtain two dis-joint datasets, denoted as \( D_{\text{C}}-\text{JSMA}\)-Train and \( D_{\text{C}}-\text{JSMA}\)-Test. There are 10,000 legitimate images and 10,000 AEs in training set. There are 1,000 legitimate images and 1,000 AEs in testing set.

\textbf{MNIST} contains 70,000 8-bit grayscale images of handwritten digits. Each image is assigned a label from 0 to 9. MNIST is split into the training and testing dataset, which contains 60,000 and 10,000 images, respectively. We carry out similar procedures on MNIST to create a training and testing set but using different target models. As a result, we have \( D_{\text{M}}-\text{CWL}_0\)-Train and \( D_{\text{M}}-\text{CWL}_0\)-Test based on Carlini\(_M\) model [7], as well as \( D_{\text{M}}-\text{JSMA}\)-Train and \( D_{\text{M}}-\text{JSMA}\)-Test based on CNN [18] model. The sizes of these datasets are the same as their counterparts in CIFAR-10. Considering that CIFAR-10 is a more challenging dataset comparing with MNIST, we will lay more emphasis on CIFAR-10 in the following experiments to make the results more convincing.

\textbf{Implementation.} We implement our Siamese-based detector in Python using the Keras [8] platform with TensorFlow [1] as backend. Keras provides a large number of high-level neural network APIs and can run on top of TensorFlow. The Telea’s inpainting algorithm [36] is implemented based on Open Source Computer Vision Library (OpenCV) [5].

The experiments were performed on a computer running the Ubuntu 18.04 operating system with a 64-bit 3.6 GHz Intel\(^\text{®}\)-Core\(^{\text{TM}}\) i7 CPU, 16 GB RAM and GeForce GTX 1070 GPU.
### Table 2: Evaluation of the $L_0$ attacks

| Dataset    | Attack  | Success rate |
|------------|---------|--------------|
| MNIST      | CW-L₀ [7] | 100%         |
|            | JSMA [29] | 81.6%        |
| CIFAR-10   | CW-L₀ [7] | 100%         |
|            | JSMA [29] | 99.8%        |

### Table 3: The classification accuracy on AEs in $D_{C-L₀} - Test$ after using inpainting-based pre-processors.

| $\alpha$ | $\beta$ | 0.0 | 0.1 | 0.2 |
|----------|---------|-----|-----|-----|
| 0.6      |         | 81.3% | 86.9% | 84.2% |
| 0.7      |         | 80.5% | 87.3% | 86.5% |
| 0.8      |         | 76.0% | 86.2% | 86.5% |
| 0.6      |         | 90.0% | 81.2% | 63.2% |
| 0.7      |         | 94.1% | 88.8% | 74.5% |
| 0.8      |         | 96.1% | 91.2% | 77.3% |

### Table 4: The classification accuracy on AEs in $D_{C-JSMA} - Test$ after using inpainting-based pre-processors.

| Dataset    | Attack  | Accuracy | Precision | Recall | F1 Score | FPR |
|------------|---------|----------|-----------|--------|----------|-----|
| CIFAR-10   | JSMA    | 99.85%   | 100.0%    | 99.70% | 99.85%   | 0.8 |
|            | CW-L₀   | 95.80%   | 94.64%    | 97.10% | 95.85%   | 5.5 |
| MNIST      | JSMA    | 99.80%   | 99.90%    | 99.70% | 99.80%   | 0.1 |
|            | CW-L₀   | 99.40%   | 99.30%    | 99.50% | 99.40%   | 0.7 |

### Table 5: The classification accuracy on testing datasets after applying SVD compression.

| Loss ratio | 60% | 40% | 20% |
|------------|-----|-----|-----|
| JSMA       | 27.4% | 17.9% | 5.4% |
| CW-L₀      | 44.5% | 35.1% | 21.4% |

### Table 6: The detection performance of the proposed system.

Impact on benign images. To investigate the impact of the defense methods on benign images, we first carry out an experiment on the 1,000 benign images from $D_{C-JSMA-Test}$. In detail, we use the ResNet model [16] to classify each color image after the inpainting-based defense is applied. The classification accuracy on these processed images only decreases from 100% to 95.6%. Next, we carry out a similar experiment on the 1,000 benign images from $D_{M-JSMA-Test}$ to classify each grayscale image after the inpainting-based defense is applied. As a result, the classification accuracy on these processed images only decreases from 100% to 99.7%. The results show that a very small impact is imposed on classifying benign images.

Summary. Therefore, though it is straightforward, the proposed inpainting-based algorithm is effective in defending against $L_0$ attacks such as CW-L₀ and JSMA. Moreover, our defense methods have a very small impact on benign images, which implies it can be directly applied without relying on detection.

### 4.4 Detecting $L_0$adversarial Inputs

We next evaluate the effectiveness of our system on detecting AEs generated by $L_0$ attacks.

#### 4.4.1 Detection Efficacy

We evaluate the detection performance of the proposed scheme against CW-L₀ and JSMA attack separately. In particular, the inpainting-based pre-processor is used to create input pairs to the Siamese network. The two training datasets, $D_{C-L₀-Train}$ and $D_{C-JSMA-Train}$, are used to train our system individually for 200 epochs using early stopping configured with a minimum accuracy change of 0.001 and 50 patience steps. If...
an accuracy change is less than 0.001, we consider there to be no improvement of the model performance; after 50 epochs with no improvement, the training is stopped. We save the resulting models as the base models.

We now evaluate the detection accuracy of the base models against CW-L0 and JSMA attack on \( \mathcal{D}_C - \text{CW-L0-Test} \) and \( \mathcal{D}_C - \text{JSMA-Test} \), respectively. Each dataset contains 1,000 benign images and 1,000 AEs. We plot the ROC (receiver operating characteristic) curves as the Figure 4 (a) shown. The AUC value can achieve 98.69% and 99.94%. More detailed results evaluated on the testing set are shown in Table 6. Note we consider the adversarial images as positive class, while we consider the benign images as negative class. Thus the recall value refers to the adversarial detection rate, defined as the ratio of the number of successfully detected AEs among the total number of AEs, to measure the effectiveness.

In general, the higher the detection rate, the better effectiveness of the system. However, in practice the expected distribution of adversarial and benign images are not balanced. Most of the samples should be benign, so a detector with a higher detection rate of successful AE but a higher FPR (false positive rate) would be less useful. The Table 6 illustrates that our models achieve very good performance. When facing CW-L0 attacks, the detection rate for the AEs from \( \mathcal{D}_C - \text{CW-L0-Test} \) is 97.1%, and the FPR is 5.5%. When facing JSMA attacks, the detection rate for the AEs from \( \mathcal{D}_C - \text{JSMA-Test} \) can achieve 99.7%, and the FPR is as low as 0.0%.

**Gray images.** We follow the same configurations to carry out a symmetric experiment on MNIST. However, training a Siamese network on MNIST is simpler than doing the same task on CIFAR-10 because gray images only have one channel. Thus, we only train the detector for 100 epochs using an early stopping setting with 30 patience steps.

We evaluate the detection accuracy of the base models against CW-L0 and JSMA attacks on \( \mathcal{D}_M - \text{CW-L0-Test} \) and \( \mathcal{D}_M - \text{JSMA-Test} \), respectively. We plot the ROC curves as the Figure 4 (b) shown. The AUC value can achieve 99.84% and 99.93%. In Table 6, we can observe similar results as the experiments given on CIFAR-10. When facing CW-L0 attacks, the detection rate for the AEs from \( \mathcal{D}_M - \text{CW-L0-Test} \) is 99.5%, and the FPR is 0.7%. When facing JSMA attacks, the detection rate for the AEs from \( \mathcal{D}_M - \text{JSMA-Test} \) can achieve 99.7%, and the FPR as is low as 0.1%.

**Comparison.** We compare the proposed system with other state-of-the-art AE detectors. To do this, we separately train two comprehensive models for color images and gray images. More specifically, for color images, we train the detector with both \( \mathcal{D}_C - \text{CW-L0-Train} \) and \( \mathcal{D}_C - \text{JSMA-Train} \). For gray images, we train an individual detector with \( \mathcal{D}_M - \text{CW-L0-Train} \) and \( \mathcal{D}_M - \text{JSMA-Train} \). We summarize the adversarial detection rate and FPR information in Table 7. For CIFAR-10, the detection rate of feature squeezing [40] on CW-L0 and JSMA attacks is 98.1% and 83.7% respectively, with a 4.9% FPR. NIC [23] can achieve a detection rate of 98.0% and 94.0%, respectively on CW-L0 and JSMA attacks with a 3.8% FPR. Our AEPecker can achieve the detection rate of 98.4% and 99.5% with a FPR of 2.0%. In addition, for MNIST, the detection rate of our model is comparable with feature squeezing [40] and NIC [23]. More importantly, the FPR of our detector is the lowest. Therefore, the proposed detector is more advanced than others.

### 4.4.2 Transferability

We seek to understand to what extent our system can be generalized to detect other types of L0 AEs which are not seen during training. More specifically, we examine whether the system trained on the JSMA-based AEs can detect those AEs generated by CW-L0. To do this, we train our system using \( \mathcal{D}_C - \text{JSMA-Train} \), and use \( \mathcal{D}_C - \text{CW-L0-Test} \) to test the detector. The result shows that the detection rate for successful AEs is as high as 99.4%. Symmetrically, we examine whether the system trained on the CW-L0-based AEs can detect those AEs generated by JSMA. To do this, we train our system using \( \mathcal{D}_M - \text{CW-L0-Train} \), and use \( \mathcal{D}_M - \text{JSMA-Test} \) to test the detector. The result shows that the detection rate for successful AEs is as high as 98.7%.

We can observe similar results on MNIST. If we leverage \( \mathcal{D}_M - \text{JSMA-Train} \) to train the Siamese network, and use \( \mathcal{D}_M - \text{CW-L0-Test} \) to test the detector, then the experimental result shows that the detection rate for successful AEs is as high as 96.3%. If we train our system using \( \mathcal{D}_M - \text{CW-L0-Train}, \)
We wonder what would happen if a weak pre-processor is used. As a result, the experiment results show that a perfect pre-processor is unnecessary for successful AEs through such pre-processor still cannot be classified correctly by the target model at a high possibility. Through this, we will show that a perfect pre-processor is unnecessary for our Siamese-based detector to achieve a high success rate of detection.

4.4.3 Variations in Pre-processor

We wonder what would happen if a weak pre-processor is selected. Here the adjective weak means the manipulated AEs through such pre-processor still cannot be classified correctly by the target model at a high possibility. Through this, we will show that a perfect pre-processor is unnecessary for our Siamese-based detector to achieve a high success rate of detection.

Without loss of generality, we take color images as an example to discuss. For color images such as CIFAR-10, each channel of RGB is encoded by 8 bits. As Figure 5 shows, we can reduce the original 8 bits to fewer bits without heavily decreasing the image recognizability to human eyes. However, Figure 5 also shows that it is very difficult to remove those striking adversarial perturbations introduced by $L_0$ attacks only with this kind of approach. Our experiment results shown in Table 8 suggest that processing the AEs generated by JSMA and CW-L_0 with bit depth reduction cannot obviously increase the classification accuracy of the target model. Consequently, bit depth reduction only has a very limited capability to defend against $L_0$ attacks.

Thus, we choose bit depth reduction as a representation of weak pre-processors for color images. The experiment results in Table 6 show that even when a weak pre-processor such as bit depth reduction is applied, our Siamese-based detector still can achieve an excellent performance. The detection rates for successful AEs generated by JSMA and CW-L_0 are 99.6% and 99.4%, respectively. The corresponding FPR is 0.4% and 2.1%. Xu et al. also use bit depth reduction as a pre-processor in [40]. As a comparison, the best detection rates provided by their system for successful AEs generated by JSMA and CW-L_0 are 36.5% and 4.1% respectively, with a FPR of 5%.

Summary. Therefore, the proposed Siamese-based detector outperforms the state-of-the-art method even when using the same weak pre-processor. It also suggests that our Siamese-based detector does not heavily rely on a perfect pre-processor to achieve a high detection performance.

4.5 Efficiency

Training time. It is widely known that neural networks usually require a large amount of data and time for training. However, as our sub-networks employed within the Siamese architecture are quite simple and shallow, the training—which is linear to the number of epochs and the number of training samples—is very efficient. For example, for $D_{M}$-JSMA-Train and $D_{C}$-JSMA-Train, each epoch with 20,000 images (10,000 benign images and 10,000 AEs) only takes 7 and 5 seconds, respectively. On the other hand, due to the simple and shallow sub-networks, with a relatively small training set our system can still achieve high detection accuracy (Section 4.4). Moreover, the training time is linear with respect to the number of epochs and the number of images for each epoch.

Our experiment results show that our system trained on CIFAR-10 and MNIST can converge very quickly and achieve high accuracy within 100 and 200 epochs, respectively—thus, the training only requires around 8 minutes and 23 minutes, respectively.

Testing time. Once the detector is successfully trained, making a prediction with it is very fast. For example, AEPecker only consumes approximately 0.5 ms on average to detect whether a test image from CIFAR-10 is adversarial or not.

5 Adaptive Attack

Suppose there is an attacker who knows the proposed method and tries to adapt attacks accordingly, we seek to understand the resilience of AEPecker against an adaptive attack by answering the following questions: (Q1) To what extent does the amplitude of altered pixels can be controlled when only using an $L_0$ attack? (Q2) Is that challenging for an attacker with respect to the number of epochs and the number of images for each epoch.
to adaptively generate $L_0$ AEs against our inpainting-based pre-processor?

To answer these questions, we use a similar method given in [17] to launch an adaptive $L_0$ attack. In detail, after each step of stochastic gradient descent (SGD), an intermediate distorted image is generated as a resolution of the optimizer. Each time the optimizer is run, the process gradually minimizes the number of altered pixels (denoted by $N_A$) simultaneously while keeping the targeted attack successful.

**Answer to Q1.** By using $N_E$ to represent the number of such altered pixels that possess extreme values (which are either smaller than $\alpha$ or larger than $\beta$), we consider the ratio $\rho = N_E/N_A$ as an indicator showing the degree to which an adaptive attack can control the amplitude of altered pixels. As an empirical analysis, we carry out SGD step by step on 100 randomly selected images from CIFAR-10. For each of the last 10 optimization steps, we examine the average ratio $\bar{\rho}$ on 100 intermediate distorted images.

Figure 6: An $L_0$ attack is launched on 100 randomly selected images from CIFAR-10. For each of the last 10 optimization steps, we examine the average ratio $\bar{\rho}$ on 100 intermediate distorted images.

The $i$-th last optimization step

| $\alpha$ | $\beta$ |
|----------|----------|
| 0.3      | 0.7      |
| 0.3      | 0.8      |
| 0.2      | 0.7      |

| $\alpha$ | $\beta$ |
|----------|----------|
| 0.3      | 0.7      |
| 0.3      | 0.8      |
| 0.2      | 0.7      |

The resulting image is a successful attack. Supposing that the processed version cannot successfully fool the DNN to a target class, we iteratively run SGD multiple times until a resolution is found. Our experiments show that the final number of altered pixels only takes up less than 2% of the total number of the pixels in images from CIFAR-10. In order to save computational time, we start checking and adaptive optimization after such percentage is lower than 5%. For a feasible resolution, we use $10 \times$ runtime at most to explore different optimization paths comparing with the situation without checking and adaptive optimization. The result finally shows that only 7% of cases by average can generate a successful AE to evade the inpainting-based pre-processor. Therefore, it is challenging for an attacker to adaptively generate $L_0$ AEs against the proposed pre-processor.

**Summary.** By considering the aforementioned limitations, the proposed technique shows resilience to an adaptive $L_0$ attack.

6 Related Work

Generally, the protection strategies against AE attacks fall into two groups, i.e., detection and defense. In this section, we will briefly review them both.

6.1 Detecting Adversarial Examples

An AE detector is a binary classifier which is designed to distinguish an adversarial sample from a legitimate one. There are two strategies which are often used to design AE detectors, i.e., adversarial training and predication mismatch.

6.1.1 Detector Training

Some techniques use both AEs and legitimate images to train a detector. For example, Li et al. [20] extract PCA features after inner convolutional layers of the neural network, then use a cascade classifier to detect AEs. Metzen et al. [26] use both adversarial and benign samples to train a CNN-based auxiliary network. This light-weight sub-network works with the target model to detect AEs. They usually require a large number of samples to train the model while we only need a relatively small dataset. More importantly, our detector achieves a high detection rate but low FPR for handling $L_0$ AEs.

6.1.2 Prediction Mismatch

Some techniques use the prediction mismatch strategy. For example, Bagnall et al. [3] train an ensemble of multiple models to use a rank voting mechanism to combine those outputs. In this way, an ensemble disagreement can be used to detect adversarial examples. Bi-model [27] firstly employs two pre-trained distinct models to generate features, then feeds the concatenated features to an additional binary classifier.
Also, Xu et al. claim that this kind of technique suffers high adversarial examples could successfully fool the deep learning networks even when facing adversarial examples. The reason why adversarial examples could successfully fool the deep learning model without being perceived is that attackers take advantage of the information redundancy of images to add adversarial noise. Consequently, well designed filters or denoisers can be considered a cure for adversarial images by removing unwanted noise. For instance, Liao et al. [22] propose higher-level guided denoisers to remove the adversarial noise from inputs; however, their approach is computationally expensive and their work does not show its effectiveness on \( L_0 \) attacks. Some other methods adopt compression techniques, such as PCA [4] and JPEG [9, 11, 14, 32], to filter out the information redundancy which may provide living space for adversarial perturbations in images; however, these approaches are not suitable for \( L_0 \) attacks. Furthermore, Bafna et al. [2] independently propose a defense against \( L_0 \) attacks; but their Fourier-transform-based approach is not as effective as ours, see Section 4.3.

### 6.2 Defense

The primary task of defensive techniques is to alleviate or eliminate the influences of AEs. In general, the current defensive techniques can be grouped into two major categories, that is, model enhancement and input transformation.

#### 6.2.1 Model Enhancement

The first category improves the resilience of DNNs by including AEs in the process of model training, i.e., adversarial training [13, 24]. However, this type of defense is usually less effective against black-box attacks than white-box attacks considering the training only focuses on one certain DNN. Also, Xu et al. claim that this kind of technique suffers high cost because of iterative re-training with both adversarial and normal examples [40]. Alternatively, defensive distillation is proposed, and can obstruct the DNNs from fitting too tightly to the data [30]. However, the prior work [7] demonstrates that the approach can be easily circumvented with a minimal modified attack such as a CW-\( L_0 \). Shield [10] enhances a model by re-training it with multiple levels of compressed images based upon JPEG. However, this method is still ineffective against \( L_0 \) attacks.

#### 6.2.2 Input Transformation

For the second category of defenses, researchers have averted their eyes from DNNs to the adversarial inputs themselves. In short, pre-processing the inputs before feeding them to networks is helpful for increasing the prediction accuracy even when facing adversarial examples. The reason why adversarial examples could successfully fool the deep learning

### 7 Limitations and Future Work

The proposed technique is not a panacea to detect or defend all possible attacks. As future work, we may explore whether our Siamese based detector can be generalized to detect other type of attack such as \( L_\infty \). Moreover, since we focus on the limitations of \( L_0 \) AEs in this paper, only an \( L_0 \) adversary is discussed as one adaptive attack. We may further consider other stronger type of adaptive attack. Finally, the experiments suggest that only controlling the number of altered pixels instead of the amplitude of such pixels will weaken the power of thus obtained AEs. Thus, developing a new strategy to guarantee an optimal balance between the number of altered pixels and their amplitude should become an important issue when designing a future AE generation algorithm.

### 8 Conclusions

Many of the most high-profile recent real-world attacks, such as adversarial graffiti on road signs, can be modeled as \( L_0 \) attacks. A highly accurate detection technique and an effective defense that can rectify the classification under \( L_0 \) perturbations are urgently needed. Although previous works consider that it is challenging to handle such heavy corruptions caused by \( L_0 \) attack, we argue that, by identifying and exploiting two inherent characteristics of \( L_0 \) AEs, it is practical to thwart such an attack. To show this, we propose a novel Siamese network based detection technique. Even only using a simple architecture, the proposed detector leads to a very high detection accuracy and a low FPR. In addition, the deployed pre-processor based on straightforward heuristics effectively rectifies those AEs. More significantly, by considering the inherent limitations of \( L_0 \) AEs, our method is resilient to adaptive adversary. The evaluation also shows the proposed technique outperforms other state-of-the-art methods. We accordingly argue that \( L_0 \) attacks are not strong enough.
References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, et al. Tensorflow: a system for large-scale machine learning. In OSDI, 2016.

[2] Mitali Bafna, Jack Murtagh, and Nikhil Vyas. Thwarting adversarial examples: An $L_0$-robust sparse Fourier transform. In NIPS, 2018.

[3] Alexander Bagnall, Razvan Bunescu, and Gordon Stewart. Training ensembles to detect adversarial examples. arXiv preprint arXiv:1712.04006, 2017.

[4] Arjun Nitin Bhagoji, Daniel Cullina, Chawin Sitawarin, and Prateek Mittal. Enhancing robustness of machine learning systems via data transformations. In IEEE Conference on Information Sciences and Systems (CISS), 2018.

[5] Gary Bradski et al. OpenCV. https://opencv.org, 2017.

[6] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a "Siamese" time delay neural network. In NIPS, 1994.

[7] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In IEEE Symposium on Security and Privacy (SP), 2017.

[8] François Chollet et al. Keras. https://keras.io, 2015.

[9] Nilaksh Das, Madhuri Shanbhogue, Shang-Tse Chen, Fred Hohman, Li Chen, Michael E Kounavis, and Duen Horng Chau. Keeping the bad guys out: Protecting and vaccinating deep learning with JPEG compression. arXiv preprint arXiv:1705.02900, 2017.

[10] Nilaksh Das, Madhuri Shanbhogue, Shang-Tse Chen, Fred Hohman, Siwei Li, Li Chen, Michael E Kounavis, and Duen Horng Chau. Shield: Fast, practical defense and vaccination for deep learning using JPEG compression. In SIGKDD, 2018.

[11] Gintare Karolina Dziugaite, Zoubin Ghahramani, and Daniel M Roy. A study of the effect of JPEG compression on adversarial images. arXiv preprint arXiv:1608.00853, 2016.

[12] Ivan Evtimov, Kevin Eykholt, Earlence Fernandes, Tadayoshi Kohno, Bo Li, Atul Prakash, Amir Rahmati, and Dawn Song. Robust physical-world attacks on deep learning models. In CVPR, 2018.

[13] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In ICLR, 2015.

[14] Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens van der Maaten. Countering adversarial images using input transformations. In ICLR, 2018.

[15] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In CVPR, 2006.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In CVPR, 2016.

[17] Warren He, James Wei, Xinyun Chen, Nicholas Carlini, and Dawn Song. Adversarial example defenses: Ensembles of weak defenses are not strong. In USENIX Workshop on Offensive Technologies (WOOT), 2017.

[18] Yash Katariya. Applying convolutional neural network on the MNIST dataset. https://github.com/yashk2810, 2017.

[19] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. arXiv preprint arXiv:1607.02533, 2016.

[20] Xin Li and Fuxin Li. Adversarial examples detection in deep networks with convolutional filter statistics. In ICCV, 2017.

[21] Bin Liang, Hongcheng Li, Miaoliang Su, Xirong Li, Wenchang Shi, and Xiaofeng Wang. Detecting adversarial image examples in deep neural networks with adaptive noise reduction. IEEE Transactions on Dependable and Secure Computing, 2018.

[22] Fangzhou Liao, Ming Liang, Yinpeng Dong, Tianyu Pang, Jun Zhu, and Xiaolin Hu. Defense against adversarial attacks using high-level representation guided denoiser. In CVPR, 2018.

[23] Shiqing Ma, Yingqi Liu, Guanhong Tao, Wen-Chuan Lee, and Xiangyu Zhang. NIC: Detecting adversarial samples with neural network invariant checking. In NDSS, 2019.

[24] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. arXiv preprint arXiv:1706.06083, 2017.

[25] Dongyu Meng and Hao Chen. Magnet: a two-pronged defense against adversarial examples. In CCS, 2017.
[26] Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. On detecting adversarial perturbations. \textit{arXiv preprint arXiv:1702.04267}, 2017.

[27] João Monteiro, Zahid Akhtar, and Tiago H Falk. Generalizable adversarial examples detection based on Bi-model decision mismatch. \textit{arXiv preprint arXiv:1802.07770}, 2018.

[28] Maria-Irina Nicolae, Mathieu Sinn, Minh Ngoc Tran, Ambrish Rawat, Martin Wistuba, Valentina Zantedeschi, Nathalie Baracaldo, et al. Adversarial robustness toolbox v0.3.0. \textit{arXiv preprint arXiv:1807.01069}, 2018.

[29] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In \textit{IEEE European Symposium on Security and Privacy (EuroS&P)}, 2016.

[30] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In \textit{IEEE Symposium on Security and Privacy (SP)}, 2016.

[31] Omkar M Parkhi, Andrea Vedaldi, and Andrew Zisserman. Deep face recognition. In \textit{British Machine Vision Conference (BMVC)}, 2015.

[32] Aaditya Prakash, Nick Moran, Solomon Garber, Antonella DiLillo, and James Storer. Protecting JPEG images against adversarial attacks. In \textit{IEEE Data Compression Conference (DCC)}, 2018.

[33] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In \textit{NIPS}, 2015.

[34] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K Reiter. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In \textit{CCS}, 2016.

[35] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In \textit{ICLR}, 2014.

[36] Alexandru Telea. An image inpainting technique based on the fast marching method. \textit{Journal of Graphics Tools}, 9(1), 2004.

[37] Naiyan Wang and Dit-Yan Yeung. Learning a deep compact image representation for visual tracking. In \textit{NIPS}, 2013.

[38] Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. A discriminative feature learning approach for deep face recognition. In \textit{ECCV}, 2016.

[39] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille. Mitigating adversarial effects through randomization. In \textit{ICLR}, 2018.

[40] Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. In \textit{NDSS}, 2018.