PAPER *Special Section on Enriched Multimedia — Multimedia Technologies Enhancing User Experience —*

Effects of Image Processing Operations on Adversarial Noise and Their Use in Detecting and Correcting Adversarial Images

Huy H. NGUYEN[†a)], ***Student Member***, **Minoru KURIBAYASHI**[††], **Junichi YAMAGISHI**[†,†††], ***Members***, ***and*** **Isao ECHIZEN**[†,†††,††††], ***Fellow***

SUMMARY Deep neural networks (DNNs) have achieved excellent performance on several tasks and have been widely applied in both academia and industry. However, DNNs are vulnerable to adversarial machine learning attacks in which noise is added to the input to change the networks' output. Consequently, DNN-based mission-critical applications such as those used in self-driving vehicles have reduced reliability and could cause severe accidents and damage. Moreover, adversarial examples could be used to poison DNN training data, resulting in corruptions of trained models. Besides the need for detecting adversarial examples, correcting them is important for restoring data and system functionality to normal. We have developed methods for detecting and correcting adversarial images that use multiple image processing operations with multiple parameter values. For detection, we devised a statistical-based method that outperforms the feature squeezing method. For correction, we devised a method that uses for the first time two levels of correction. The first level is label correction, with the focus on restoring the adversarial images' original predicted labels (for use in the current task). The second level is image correction, with the focus on both the correctness and quality of the corrected images (for use in the current and other tasks). Our experiments demonstrated that the correction method could correct nearly 90% of the adversarial images created by classical adversarial attacks and affected only about 2% of the normal images.

key words: *adversarial machine learning, image processing operation, detecting adversarial image, correcting adversarial image, data cleansing, deep neural network*

1. Introduction

Despite the success of deep learning in both academia and industry, deep neural networks (DNNs) are vulnerable to adversarial attacks [1], and this has attracted much attention and effort. Besides traditional logical attacks in which adversarial noise is added to image or audio files, attackers can now create physical adversarial examples [2]–[6]. When autonomous systems have become mainstream, physical adversarial attacks may threaten their safety and reliability. Besides white-box attacks, in which attackers have full knowledge of the inner configuration of the target models, attackers will also be able to perform black-box attacks, which are more likely since attackers need acquire only the models' outputs [7]. Moreover, attackers are able to create universal adversarial perturbations that are applicable to multiple inputs [8] and to create adversarial examples that can be used to attack multiple DNNs [9]. Such adversarial examples could be used to directly attack DNN-based systems or to poison the training data of DNNs to corrupt their models (a "data poisoning attack") [10].

Approaches to counter adversarial attacks can be classified into four groups: adversarial example detection, adversarial training, input pre-processing, and randomization or private keying. Some approaches were designed for multiple domains while others were simply designed for a single domain like the image one. For adversarial example detection, a statistical-based approach is commonly used [11], [12]. Another approach is to build a detector that takes raw images [13] or features from intermediate layers of the targeted DNN [14], [15] as input. Ma *et al.* used local intrinsic dimensionality to characterize the adversarial subspace [16]. Xu *et al.* presented a feature squeezing method in which the differences in the DNN's outputs between the normal and squeezed images are used for detection [17]. Liang *et al.* subsequently proposed an adaptive noise reduction method [18]. For adversarial training, there are several approaches including distillation [19], obfuscating gradients [20], optimizing the saddle point formulation [21], and applying the reverse cross-entropy loss function [22]. For input pre-processing, Guo *et al.* trained DNNs on transformed images (to which cropping, total variance minimization, and/or quilting operations had been applied) so as to mitigate adversarial noise [23]. Prakash *et al.* proposed using a DNN-based adaptive JPEG encoder to pre-process the input [24]. For randomization or private keying, Taran *et al.* proposed a key-based diversified aggregation mechanism to defend against gray- and black-box adversarial attacks [25]. Several adversarial databases have been independently created for evaluation, but guidelines for creating them have not been reported in detail [11], [17], [23].

In this paper, we present methods for detecting and correcting adversarial examples in the digital image domain. By limiting our focus to the digital image domain, we can use more domain knowledge and can use many potential efficient and cheap image processing operations for our framework. We target classical adversarial attacks that do not consider optimizing the robustness of adversarial perturbations

Manuscript received March 27, 2021.
Manuscript revised July 12, 2021.
Manuscript publicized October 5, 2021.

[†]The authors are with The Graduate University for Advanced Studies, Kanagawa-ken, 240–0193 Japan.
[††]The author is with Graduate School of Natural Science and Technology, Okayama University, Okayama-shi, 700–8530 Japan.
[†††]The authors are with National Institute of Informatics, Tokyo, 101–8430 Japan.
[††††]The author is with University of Tokyo, Tokyo, 113–8654 Japan.
a) E-mail: nhhuy@nii.ac.jp

DOI: 10.1587/transinf.2021MUP0005

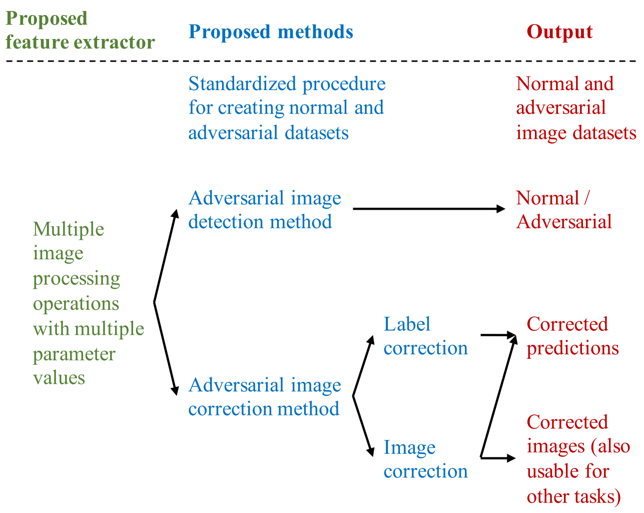

Proposed feature extractor
Proposed methods
Output
Standardized procedure for creating normal and adversarial datasets
Normal and adversarial image datasets
Multiple image processing operations with multiple parameter values
Adversarial image detection method
Normal / Adversarial
Adversarial image correction method
Label correction
Image correction
Corrected predictions
Corrected images (also usable for other tasks)

Fig. 1 Visualization of our contributions.

against transformations. These attacks generally require less computation than robust ones, so attackers can easily create a massive number of adversarial examples in a short period of time. One practical use is for data poisoning. We target both black-box and white-box attack scenarios.

Our approach is to use multiple image processing operations with multiple parameters for detection and correction. These parameters include the quality factor (QF) of JPEG compression, the scaling factor, the size of the Gaussian blur kernel, and the rotation angle. We hypothesized and then verified that the classification labels of the adversarial images change when the values of the parameters change while the classification labels of normal images remain mostly unchanged. Our approach to detection and correction does not require any modification or re-training of the target DNNs. In summary, our contribution is three-fold (visualized in Fig. 1):

- We introduce a standardized procedure for creating a normal dataset and an adversarial dataset. The latter includes several state-of-the-art targeted and non-targeted adversarial attacks [7]. Targeted attacks are attacks that try to change the output label of a DNN to a predefined label while non-targeted attacks are attacks that aim to make the output label of a DNN different from the original one. Both datasets are derived from the ImageNet validation set [26]. The adversarial dataset created using this procedure has protocols for evaluating seen and unseen attacks to measure the models' generalizability. This standardized procedure is expected to promote fair comparisons and reproducible research in adversarial machine learning (ML).
- We present a method that uses operation-oriented characteristics for detecting adversarial images. It is unrealistic to use an exhaustive search to find the optimal combination of multiple image processing operations with multiple parameter values. We thus observed the changes in DNN output with an increase in the strength of image processing operations, which work as a kind of noise removal filter. Using simulation, we quantitatively evaluated and identified the best choice of operation-oriented characteristics. This method is more advanced than the feature squeezing method [17], which uses only three image processing operations with manually set parameter values.
- We present a method that uses for the first time two levels of adversarial image correction: label correction (to restore the correct labels of the adversarial images for the current task) and image correction (to mitigate the adversarial noise in the adversarial images so that the targeted DNN can correctly work on them and making these images usable in other tasks). In label correction, only the outputs of the DNNs are of interest while in image correction, the quality and usability of the corrected images as well as the labels are of interest. Our proposed method is heuristic and is based on using multiple image processing operations with multiple parameter values. Unlike Guo *et al.*'s method [23], our method does not require re-training of the DNNs and is applicable to their pre-trained models. Unlike Prakash *et al.*'s method [24], our method uses multiple standardized image processing operations rather than a customized DNN-based JPEG encoder, which is not robust to adversarial attacks.

The rest of the paper is organized as follows: In Sect. 2, we introduce our standardized procedure for creating the normal and adversarial datasets used for the experiments described in this paper. Next, in Sect. 3, we discuss the effects of using multiple image processing operations with multiple parameter values on both normal and adversarial images. The observed distinctive effects are used as the backbone for the proposed adversarial image detection method described in Sect. 4 and the image correction method described in Sect. 5. We summarize the key points and discuss future work in Sect. 6.

2. Dataset Creation

2.1 Overview

Although several normal and adversarial datasets have been independently created, detailed guidelines for their creation are lacking [11], [17], [23]. We thus created our own datasets as shown in Fig. 2 for use in our experiments. We applied several policies when designing them:

- They must be large enough to be used to train both handcrafted and convolutional neural network (CNN) based methods.
- The adversarial dataset must contain images for various types of adversarial attacks including both targeted and non-targeted ones [7]. This is crucial to building protocols for seen and unseen attacks.
- Since the datasets were derived from the ImageNet database, we followed its protocol by utilizing the top-5 accuracy as the metric for all experiments. A correct

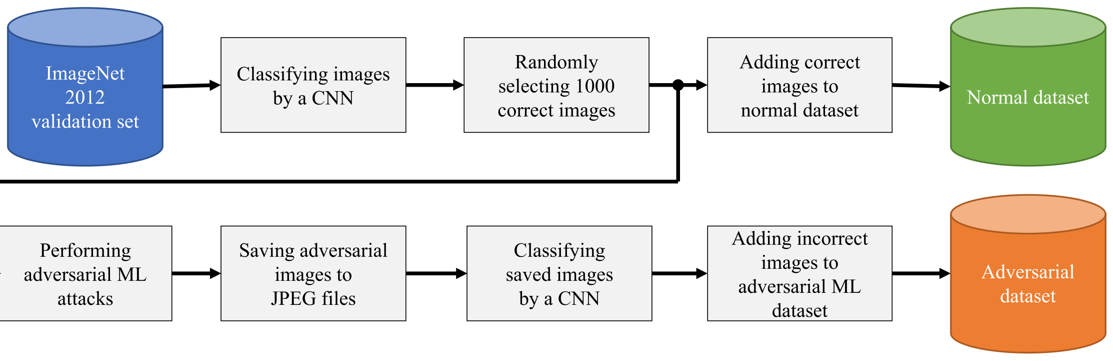

ImageNet 2012 validation set
Classifying images by a CNN
Randomly selecting 1000 correct images
Adding correct images to normal dataset
Normal dataset
Performing adversarial ML attacks
Saving adversarial images to JPEG files
Classifying saved images by a CNN
Adding incorrect images to adversarial ML dataset
Adversarial dataset

Fig. 2 Overview of dataset creation procedure using a CNN. The same procedure was used with VGG-16, VGG-19, ResNet-18, and ResNet-50 networks to together create a full normal dataset (for normal images) and a full adversarial dataset (for adversarial images).

classification is defined to be when one of the top-5 predicted labels is the true label.

- All images in the normal dataset must be correctly classified by the targeted CNNs (100% accuracy). All images in the adversarial dataset must be misclassified by the targeted CNNs (0% accuracy). To make the attacks realistic, the images in the adversarial dataset are saved as JPEG files since some adversarial noise is mitigated when saving.
- Since some labels fall into the same category (for example, different breeds of dogs), the adversarial attacks should produce adversarial images with a label belonging to a different category than the original one.

In the next sections, we describe in detail the dataset construction and the adversarial attacks used. Finally, we describe the analysis we performed on the newly created datasets to determine their quality and to gain some insight about them.

2.2 Dataset Construction

We used the images from the validation set of the ImageNet database [26] (which has ground-truth labels) to generate the adversarial images used in our experiments. There were 1,000 labels in total. For the object-recognition CNNs, we used the VGG-16 and VGG-19 networks [27] and the ResNet-18 and ResNet-50 networks [28] pre-trained on the ImageNet database, implemented using the PyTorch framework [29]. We used the Pillow library[†] for image processing and the FoolBox library (version 1.8.0) [30] for adversarial image generation. Twelve commonly used methods (Table 1) were used to perform targeted and non-targeted adversarial attacks.

As shown in Fig. 2, we used each CNN to classify five million images from the ImageNet 2012 validation set. Classification was considered correct when the ground-truth label was one of the predicted top-5 labels. We then randomly selected 1,000 images per CNN from the correctly classified images; therefore, there were 4,000 normal images in total. We limited the number because the time required to craft adversarial images from them is quite long. These 4,000 images were added to the normal dataset. We then performed the 12 adversarial attacks listed in Table 1 on the 4,000 images as described in Sect. 2.3. The Pillow library was used to save them as JPEG files with a QF of 100 to preserve most of the adversarial noise. We then loaded the saved images and used the same CNN to classify them again to ensure that they were still misclassified (the predicted top-5 results did not contain the ground-truth label). We obtained 22,326 misclassified adversarial images in total and added them to the adversarial dataset.

We distributed the obtained normal and adversarial images into training (train), development (dev), and evaluation (eval) sets as detailed in Table 2. The train set was used for training, the dev set was used to select the model, and the eval set was used to test the detectors. We ensured that the normal images and their adversarial versions in the train, dev, and eval sets did not overlap so that the detectors would not remember the training images. There were three attack settings: targeted, non-targeted, and combination. The targeted and non-targeted attack settings were used to test the generalization of the adversarial image detectors while the combination setting was used mainly for hyper-parameter

[†]https://pillow.readthedocs.io/en/stable

Table 1 Number of successful adversarial images created from 1,000 input images using FoolBox library [30] on VGG-16, VGG-19, ResNet-18, and ResNet-50 networks.

Method	VGG-16	VGG-19	ResNet-18	ResNet-50
Targeted attacks:				
L-BFGS [1]	618	623	348	229
BIM [2]	693	711	531	431
PGD [2]	651	678	484	348
L1-iter [30]	661	591	680	513
L2-iter [30]	820	746	722	583
Non-targeted attacks:				
Gradient [30]	654	548	590	517
FGSM [31]	509	450	451	379
Deep Fool [32]	707	671	658	604
Newton [33]	571	505	558	425
ADef [34]	1	4	1	1
JSMA [35]	102	86	114	71
Carlini-Wagner [36]	427	361	401	299

selection for the proposed detection method. This combination attack setting was also used for all experiments on adversarial image correction.

2.3 Crafting Adversarial Images

To craft the adversarial images used for the targeted attacks, we used the limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) method proposed by Szegedy *et al.* [1], the basic iterative method (BIM) using L-infinity, the projected gradient descent (PGD) method described by Kurakin *et al.* [2], and the L1- and L2- versions of BIM (L1-iter and L2-iter) implemented in FoolBox [30]. For the indexes of the image labels of the ImageNet database, nearby labels often fall into the same group. For example, 239 is "Bernese mountain dog," 245 is "French bulldog," and 250 is "Siberian husky." All of them are dog breeds. To maximize the effect of the adversarial attacks, the target label should be in a different category than the original label. The target label for the adversarial attack image was thus shifted 100 steps right from the predicted top-1 label for the normal image. It is important to note that we used the top-1 predicted label, not the ground-truth label annotated in the database to perform adversarial attacks in order to make them more realistic (which is in accordance with the viewpoint of an attacker attacking a large-scale system without any knowledge of the ground-truth labels). Since there were 1,000 labels in total, modular operation was used to ensure the shifted label index was in the range $[0, 1000)$. The attack objective was to achieve a target class probability of 99%.

For non-targeted attacks, we used the basic gradient attack method implemented in FoolBox [30], the fast gradient signed method (FGSM) proposed by Goodfellow *et al.* [31], the Deep Fool method proposed by Moosavi-Dezfooli *et al.* [32], the Newton method proposed by Jang *et al.* [33], the ADef method proposed by Alaifari *et al.* [34], the Jacobian-based saliency map attack (JSMA) method proposed by Papernot *et al.* [35], and the method proposed by Carlini and Wagner [36]. Since these methods are non-targeted attacks, the attack objective was to change the predicted top-5 labels so that they differed from the original predicted top-1 labels.

Table 2 Details of normal and adversarial image datasets, which were divided into training, development, and evaluation sets for three attack settings.

Attack Setting/Datasets	Normal	Adversarial	Total	Ratio
Targeted attacks:				
Train	2,800	8,392	11,192	1:3.00
Dev	600	1,623	2,223	1:2.71
Eval	600	1,646	2,246	1:2.74
Non-targeted attacks:				
Train	2,800	7,718	10,518	1:2.75
Dev	600	1,469	2,069	1:2.45
Eval	600	1,478	2,078	1:2.46
Combination attacks:				
Train	2,800	16,110	18,910	1:5.75
Dev	600	3,092	3,692	1:5.15
Eval	600	3,124	3,724	1:5.21

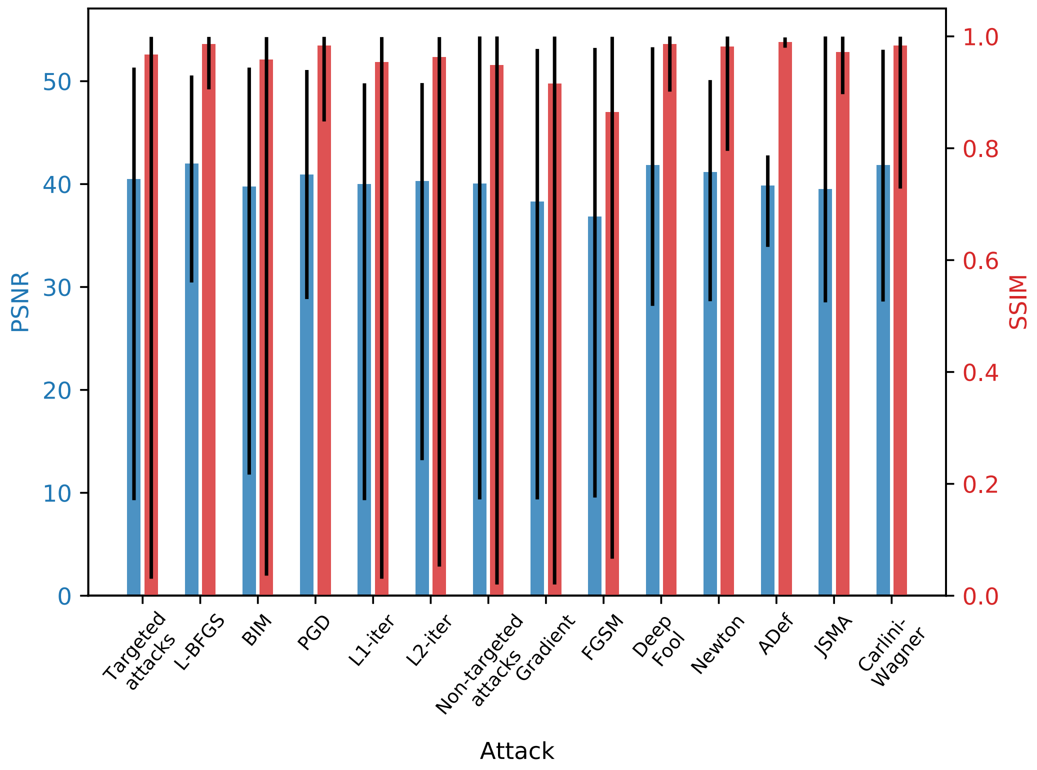


Fig. 3 Average PSNR (blue) and SSIM (red) between adversarial images used for targeted and non-targeted attacks and corresponding original images. Error bars indicate min and max values.

2.4 Dataset Analysis

As shown by the results in Table 1, the VGG networks were generally more vulnerable to targeted attacks than the ResNet networks, resulting in more misclassified images. One possible explanation is that the skip connections used by ResNet networks make them more robust than the VGG networks. The modified BIM attack using the L2 distance (L2-iter) was the most effective attack overall. The non-targeted attacks were more difficult to carry out since they needed to change the top-5 labels so that they did not include the current top-1 labels. Among the attack methods, Deep Fool was the most successful while the ADef attack was the least successful overall, producing only one or four adversarial images for each network. The JSMA method also had limited success, with around 100 adversarial images for each network.

Beside the success rate of attacks, we also evaluated the quality of the obtained adversarial images by calculating the peak-signal-to-noise ratio (PSNR) and the structural similarity index measure (SSIM) [37] between them and their corresponding original images. The results are shown in Fig. 3. Although there were some extreme cases, the average PSNR was about 40 dB while the average SSIM was greater than 0.9. This means that the quality of the adversarial images was high and that the perceived quality of the original images was preserved. The L-BFGS, Deep Fool, and Carlini-Wagner attacks produced adversarial images with the highest quality whereas the Gradient and FGSM attacks produced ones with the lowest quality. The L-BFGS, PGD, Deep Fool, ADef, JSMA, and Carlini-Wanger attacks produced adversarial images with stable and high SSIM.

3. Effects of Image Processing Operations on Normal and Adversarial Images

Some image processing operations like JPEG compression, spatial smoothing, and scalar quantization have recently been found to have noticeable effects on adversarial noise [17], [18], [23]. We expand on this and hypothesize that using multiple common image processing operations (already implemented in many image processing libraries) and multiple parameter values will provide much more useful information than using operations with fixed values as in previous work. Furthermore, the effects of multiple parameter values on adversarial images differ from those on natural images depending on the differences in the values. Although common image processing operations like JPEG compression, Gaussian blurring, rotation, and scaling do not substantially affect the usability of the processed images, they have certain non-linear effects on adversarial noise. JPEG compression reduces the number of bits needed for storage. Gaussian blurring removes high-frequency components and thus acts as a low-pass filter. Rotation, which requires the use of an interpolation algorithm to adjust the pixels, removes noise[†]. Scaling up also removes noise while scaling down, which reduces the entropy of an image (the degree of disorder, which is used to characterize the texture of an image), reduces the amount of noise. These effects on adversarial noise can be used to distinguish adversarial images from normal images as well as to correct adversarial images. It should thus be possible to identify operation-dependent characteristics from the outputs of an object-recognition CNN as the strength of an operation is gradually increased.

To confirm this hypothesis, we applied the four image processing operations to images from our two datasets and classified the resulting images using the VGG-16, VGG-19, ResNet-18, and ResNet-50 networks. The parameter values were as follows:

- JPEG compression with QF $\in$ {100, 95, 90, 85, 80, 75, 70, 65, 60, 55, 50, 45, 40, 35, 30, 25}.
- Gaussian blurring with kernel size $\in$ {2, 3, 4, 5}.
- Clockwise image rotation with angle $\in$ {1°, 2°, 3°, 4°, 5°, 6°, 7°, 8°} and without reversing back. Nearest-neighbor interpolation was used.
- Image scaling with scale $\in$ {0.75, 0.8, 0.85, 0.9, 0.95, 1.05, 1.1, 1.15, 1.2, 1.25} and without reversing back. Nearest-neighbor interpolation was used.

The image operations were done using Pillow version 6.1.0. Some of the results for ResNet-50 are shown in Table 3. Similar effects were also observed for VGG-16, VGG-19, and ResNet-18. The combined result of all networks is visualized in Fig. 4. JPEG compression with a QF of 100 changed the top-5 labels for both the normal and adversarial images, and the effect on adversarial images was clearer. Reducing image quality by increasing the compression ratio greatly reduced the number of misclassified adversarial images and slightly increased that of misclassified normal images. A large increase in the ratio increased the misclassification rate for normal images. The results for scaling and rotation were similar to those for compression while those for Gaussian blurring differed slightly. JPEG compression is thus the best candidate to eliminate adversarial noise while scaling is the best for preserving the correctness of normal images. One result in particular should be noted: the number of misclassified normal images after applying a 3×3 Gaussian blur kernel was higher than after applying the other operations, which is not good for our objectives, *i.e.*, detection and correction of adversarial images. In summary, these results support our hypothesis that the classification labels of the adversarial images change when the values of the parameters change while the classification labels of normal images remain mostly unchanged. The next two sections describe the methods we devised for detecting and correcting adversarial examples by exploiting variations in the outputs of a DNN.

[†]We used nearest-neighbor interpolation in our experiments.

Table 3 Number of top-5 **misclassified** images for ResNet-50 network from both normal and adversarial datasets before and after applying image processing operations.

Attack Setting/Datasets	**Original**	**JPEG Compression**					**Scaling**						**Gaussian Blurring**	**Rotation**	
		100	**80**	**60**	**40**	**20**	**0.75**	**0.85**	**0.95**	**1.05**	**1.15**	**1.25**	**3×3**	**2°**	**5°**
Normal images	0	1	18	33	46	92	43	23	18	13	26	29	259	32	57
Targeted attack:															
L-BFGS [1]	229	221	27	31	41	67	45	27	26	25	38	36	43	40	58
BIM [2]	431	423	47	49	59	88	51	38	40	42	57	50	65	60	79
PGD [2]	348	339	28	40	47	82	44	29	28	27	42	34	58	48	69
L1-iter [30]	513	507	51	51	56	81	55	48	51	55	67	59	73	79	84
L2-iter [30]	583	575	44	46	55	88	56	42	46	46	64	52	67	74	89
Non-targeted attack:															
Gradient [30]	517	515	112	89	93	114	94	77	91	107	124	115	108	126	155
FGSM [31]	379	373	128	129	134	139	100	71	92	120	155	127	95	138	158
Deep Fool [32]	604	594	32	21	27	55	33	24	34	37	34	30	59	59	50
Newton [33]	425	411	60	38	36	61	34	27	41	49	47	38	82	76	68
ADef [34]	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
JSMA [35]	71	72	29	18	29	38	35	23	23	20	25	20	33	35	44
Carlini-Wagner [36]	299	289	28	24	31	60	29	20	25	18	33	27	38	43	49

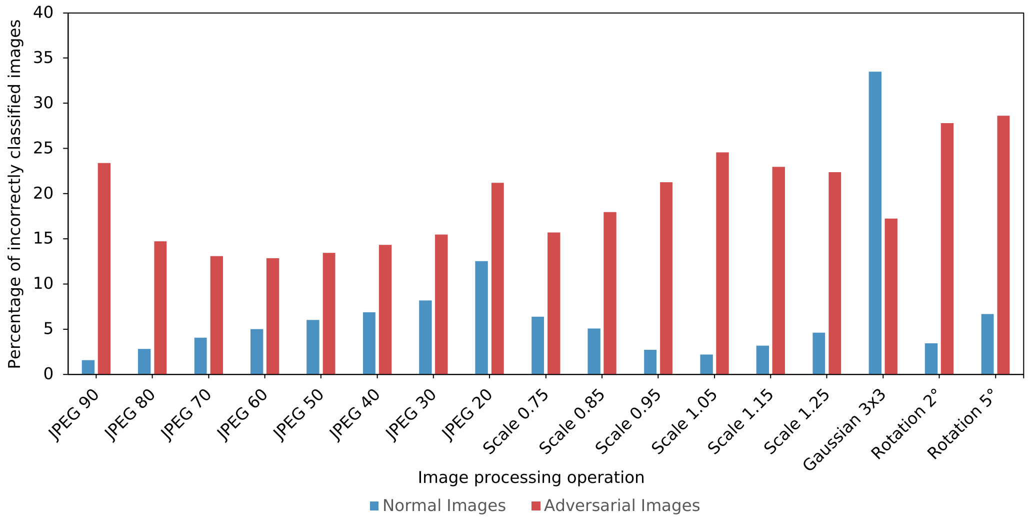


Fig. 4 Percentages of incorrectly classified normal and adversarial images after applying four image processing operations. Note that before applying these operations, the accuracy on normal images was 100% and on adversarial images was 0%. (Due to our dataset design, unqualified images were eliminated.)

4. Detecting Adversarial Images

As demonstrated in the previous section, the four image processing operations had different effects on normal and adversarial images. Moreover, changing the parameter values changed the DNN outputs. We utilize these effects and two statistical-based features, a counting feature (described in Sect. 4.1) and a differences feature (described in Sect. 4.2), to detect adversarial images. With each of these two features, we use four traditional machine learning classifiers (**traditional classifiers**, described in Sect. 4.3), and a CNN-based classifier (**stats-CNN classifier**, described in Sect. 4.4). For comparison, we used a feature squeezing method [17] as a baseline since it and our detection method are conceptually similar. To make our results more convincing, we also used as baselines two CNN-based classifiers that take raw images as input (**raw-image CNN classifiers**) (described in Sect. 4.4).

As statistical-based features of our detection method, we define three variables:

- $L = (a, b, c, d, e)$: top-5 label for image I (normal or adversarial) predicted using a CNN **before** applying image processing operation i (*e.g.*, JPEG compression with a QF of 80 or 5° clockwise rotation).
- $L_i = (a_i, b_i, c_i, d_i, e_i)$: top-5 label for an image **after** applying image processing operation i.
- n: total number of image processing operations (38 in our experiments).

4.1 Counting Feature

We define $C(a)$ as the number of occurrences of label $a \in L = (a, b, c, d, e)$ at the first position in an ordered top-5

Table 4 Accuracy (in %) of each classifier using counting feature (count) and differences feature (diff.) on eval set.

Classifier	JPEG Compression		Scaling		Gaussian Blurring		Rotation		JPEG + Scaling		All	
	Count	Diff.	Count	Diff.	Count	Diff.	Count	Diff.	Count	Diff.	Count	Diff.
SVM (SVC) [38]	90.98	91.92	92.56	92.32	87.35	87.35	89.90	89.39	92.86	**94.04**	93.39	**94.20**
Random forest [39]	90.57	89.82	91.08	91.27	86.87	86.04	88.24	88.59	92.16	91.94	92.35	92.35
LDA [40]	89.98	90.36	91.97	91.62	85.77	87.35	89.15	89.29	92.86	92.67	92.21	92.86
MLP [41]	91.25	90.41	92.32	92.35	86.95	87.38	89.66	89.02	**93.56**	92.19	**93.37**	92.29

label set $\{L_i | i = 1, \dots, n\}$.

$C(a)$ is defined as

$$C(a) = \sum_{i=1}^{n} \delta(a, a_i), \tag{1}$$

with

$$\delta(a, a_i) = \begin{cases} 1, & \text{if } a = a_i \\ 0, & \text{if } a \neq a_i. \end{cases} \tag{2}$$

The same equation is used for b, c, d, and e at the second, third, fourth, and fifth positions, respectively. Therefore, the features of each image are $\{C(a), C(b), C(c), C(d), C(e)\}$.

The following is a simple example of the counting feature using three image processing operations. Given $L = (100, 105, 198, 213, 479)$, we have

- $L_1 = (100, 97, 198, 220, 221)$
- $L_2 = (101, 80, 201, 119, 212)$
- $L_3 = (100, 89, 213, 117, 304)$

after applying the image processing operations. The resulting counting features are $\{2, 0, 1, 0, 0\}$.

4.2 Differences Feature

We define $\Delta(a, a_i)$ as the binary differential function used to measure the difference between two labels a and a_i:

$$\Delta(a, a_i) = 1 - \delta(a, a_i). \tag{3}$$

The differences feature derived from input image I can be expressed as the set

$$\{(\Delta(a, a_i), \Delta(b, b_i), \Delta(c, c_i), \Delta(d, d_i), \Delta(e, e_i)) | i = 1, \dots, n\}.$$

The differences features obtained using this example are $\{0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1\}$.

4.3 Feature Selection and Hyper-Parameter Tuning

Using the two statistical-based features introduced above (counting and differences), we evaluated the performance of detectors using one of the four traditional machine learning classifiers: the C-support vector classification (SVC) version of the support vector machine (SVM) classifier [38], the random forest classifier [39] with 100 trees and a maximum depth of 2, the linear discriminant analysis (LDA) classifier [40], and the multiple layer perception (MLP) classifier [41]. All of them were implemented in the scikit-learn library version 0.21.3[†]. For simplicity, we call them **traditional classifiers**. It is important to note that we use the terms "detector" and "classifier" interchangeably. In this feature selection and hyper-parameter tuning step, we trained them on only the dataset for the combination attack setting.

As shown in Table 4, the differences feature and the counting one generally produced similar accuracies. Among the individual image processing operations, the scaling one achieved the highest accuracy for all detectors. The combination of JPEG compression and scaling and the combination of all operations resulted in higher accuracy than using any of them individually. However, these combinations also increased the feature size, and more classification operations were required for the object detection CNNs (VGG-Nets and ResNets) to produce those features. For the counting feature, using the MLP-based detector on the features from JPEG compression and the scaling operation resulted in the highest accuracy (93.56%) while for the differences feature, using the SVM-based detector on all features from all image processing operations resulted in the highest accuracy (94.20%).

4.4 CNN-Based Classifiers

In addition to the traditional machine learning algorithms described in Sect. 4.3, we used two simple feed-forward CNNs as classifiers, one that takes the counting feature as input and one that takes the differences feature. The two CNNs used all the features extracted using the four image processing algorithms: JPEG compression, scaling, Gaussian blurring, and rotation. Each CNN had five layers of 1-D convolution (each convolution layer was followed by a batch normalization layer and a rectified linear unit), and two fully connected layers at the end with a dropout rate of 50%. For simplicity, we call these two networks **stats-CNN classifiers**.

We also used two common CNNs that take raw images as input: ResNet-50 [28] and XceptionNet [42]. The XceptionNet one is commonly used for forgery detection [43]. We modified their last fully connected layer so that the output was binary. These two CNN-based classifiers take images as input and output the probabilities that those images are adversarial. Since there was limited training data, beside training them from scratch, we also fine-tuned their Im-

[†]https://scikit-learn.org/stable

ageNet pre-trained versions. For simplicity, we call these two networks **raw-image CNN classifiers**.

For both the stats- and raw-image CNN classifiers, we used a learning rate of 5×10^{-4} for all cases. Each network was trained for 150 epochs, and the checkpoint with the highest accuracy on the dev set was selected for evaluation.

4.5 Evaluation

We selected the two best traditional classifiers, the SVM (SVC) one using all features and the MLP one using the JPEG compression and scaling features, to compare with the stats-CNN classifiers, the raw-image CNN classifiers, and the feature squeezing method [17]. We tested them on two scenarios: (1) seen attacks in which all classifiers were trained and tested on the dataset for the combination setting and (2) unseen attacks in which all classifiers were trained on the dataset for the targeted attack setting and were tested on the dataset for the non-targeted attack setting and vice versa. The dataset details are listed in Table 2. Since the feature squeezing method only requires training on normal images, we used the "Best Joint Detection (5-bit, 2x2, 11-3-4)" setting [17] (designed for the ImageNet database) with a threshold of 1.2128 pre-trained for all scenarios. Since the output of the feature squeezing method is binary, EERs could not be calculated. For the other methods, the accuracies were calculated using a threshold of 0.5. The experiment results are shown in Table 5.

Overall, the raw-image CNN classifiers outperformed the statistical-based ones (including the traditional classifiers and the stats-CNN classifier) and the feature squeezing method in both the seen and unseen scenarios. Between the two raw-image CNN classifiers, surprisingly, the XceptionNet one trained from scratch achieved better performance and outperformed its fine-tuned version with accuracies greater than 99% and EERs less than 1%. On the other hand, the ResNet classifier with fine-tuning outperformed its trained version. The performances of the traditional classifiers and the stats-CNN classifiers were similar, indicating that their scores are the upper limits for the statistical features. Although having limited results and limited generalizability compared with the raw-image CNN classifiers, our statistical-based classifiers nevertheless have reasonable discriminative ability and overall outperformed the feature squeezing method. These results support our hypothesis that using various parameter values is better than using fixed ones. In addition, it is important to note that CNN-based classifiers, especially raw-image ones, are potentially vulnerable to second-level adversarial attacks. That is, attackers can disguise their adversarial images by adding secondary adversarial noise to alter the output of the adversarial image detector from adversarial to normal. In this case, our statistical-based detection method is more robust.

Another interesting result is that the detectors trained on the dataset for the non-targeted attack setting had better generalizability than those trained on the dataset for the targeted attack one. This can be interpreted to mean that the adversarial noise created by non-targeted attacks is more general than that created by targeted attacks. Therefore, when designing an adversarial machine learning dataset, it is crucial to include a sufficient amount of data for non-targeted attacks.

Table 5 Accuracy and EER (in %) of each detector on eval sets. First two detectors were best statistical-based classifiers selected on basis of results presented in previous section. Next four were CNN-based detectors used as baselines.

Attack Setting/Detector	Accuracy	EER
Seen attacks (Combination):		
MLP - counting feature	93.56	10.00
SVM - differences feature	94.20	9.67
CNN - counting feature	93.29	10.21
CNN - differences feature	94.09	9.50
Feature squeezing [17]	83.97	-
ResNet-50 (training)	83.89	49.09
ResNet-50 (fine-tuning)	99.49	0.66
XceptionNet (training)	**99.95**	**0.15**
XceptionNet (fine-tuning)	99.60	0.49
Targeted attacks → Non-targeted attacks:		
MLP - counting feature	68.62	18.81
SVM - differences feature	71.03	15.16
CNN - counting feature	60.39	21.31
CNN - differences feature	66.79	17.73
Feature squeezing [17]	72.71	-
ResNet-50 (training)	71.13	50.50
ResNet-50 (fine-tuning)	95.72	3.17
XceptionNet (training)	**99.42**	**0.60**
XceptionNet (fine-tuning)	83.69	12.95
Non-targeted attacks → Targeted attacks:		
MLP - counting feature	93.63	3.77
SVM - differences feature	93.99	3.76
CNN - counting feature	93.32	3.04
CNN - differences feature	93.32	4.02
Feature squeezing [17]	84.42	-
ResNet-50 (training)	99.42	0.61
ResNet-50 (fine-tuning)	99.02	0.97
XceptionNet (training)	**100.00**	**0.00**
XceptionNet (fine-tuning)	99.78	0.52

5. Correcting Adversarial Images

Given that the image processing operations substantially reduced the number of misclassified adversarial images while only slightly affecting the normal images, we utilized them to correct adversarial images. We developed a correction method with two levels of correction:

- **Label correction**: Restore the original labels of adversarial images. At this level, only the output labels are of interest; for example, only the true label of a manipulated stop sign is of interest to a self-driving car.
- **Image correction**: Mitigate adversarial noise in adversarial images to restore their original labels and make them usable for other tasks. At this level, the quality and usability of the corrected images as well as the labels are of interest. Image correction is therefore more complicated than label correction.

Our correction method can be used independently or in combination with an adversarial image detector. Since it is not always clear whether an image is normal or adversarial, a correction method should work well on both normal and adversarial images. Details of the label correction and image correction algorithms are described in the next two sections.

5.1 Label Correction Algorithm

We define $\mathbf{S_L} = \{L_i | i = 1, \dots, n\}$ as the set of top-5 labels acquired by applying n image processing operations to image I. The frequencies of every label in $\mathbf{S_L}$ are calculated, and the five labels with the highest frequencies are identified as the corrected top-5 ones.

5.2 Image Correction Algorithm

In the context of image classification, an image correction method must satisfy the following conditions:

- Adversarial noise must be mitigated so that the corrected images can be correctly classified by the DNN.
- The usability of the adversarial images must be recovered and that of the normal images must be preserved.
- The quality of both the normal and adversarial images must be preserved as much as possible.

Since JPEG compression has good performance on both normal and adversarial images, we use it as the core image processing operation to eliminate adversarial noise. There are two ways of using JPEG compression for image correction:

1. *Baseline:* Use a **fixed QF** for JPEG compression. The downside with this approach is that there are trade-offs between performance on normal and adversarial images and between performance and the quality of the corrected images.
2. *Proposed:* Use a **heuristic algorithm** to determine the optimal QF for JPEG compression. With this approach, the corrected labels calculated with the label correction algorithm are used to calculate the best QFs for producing those labels. This is discussed in detail below.

We define f_{label} as the corrector function described in Sect. 5.1 and I as the image to be corrected. The proposed heuristic algorithm has three steps:

- ***Step 1***: Calculate the corrected top-5 labels: $L_{corr} = f_{label}(I)$.
- ***Step 2***: For each label l in L_{corr}, select the highest QF from the sorted list of QFs $\{100, 95, 90, \dots, 40\}$ that produces l and put it into S_{QF}.
- ***Step 3***: Select a QF from S_{QF} and use it to compress I to obtain the corrected image.

The strategies that can be used for selecting a QF from S_{QF} in step 3 are visualized in Fig. 5. The candidates are *max*, *median*, 2nd*min*, and *min* of S_{QF}. The higher the QF, the better the quality of the corrected image. If the input image is natural, $max(S_{QF})$ is obviously the best choice. However, if the input image is adversarial, $min(S_{QF})$ seem to be the safest choice. The performances achieved with these choices are presented and discussed in the next section.

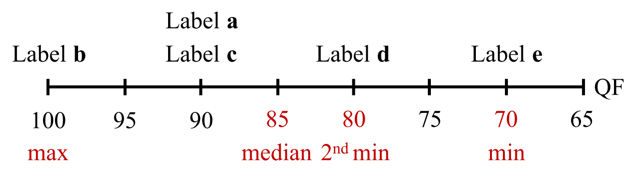


Fig. 5 Visualization of corrected top-5 labels calculated in step 1 of heuristic algorithm and their corresponding QFs calculated in step 2. The *max*, *median*, 2nd*min*, and *min* of S_{QF} labels on the axis represent the candidate QFs to be selected in step 3.

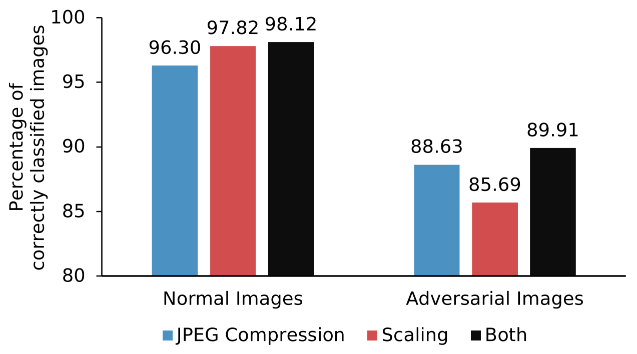


Fig. 6 Percentage of correct classification results after applying label correction method to both normal and adversarial images.

5.3 Evaluation

Since there is no learning required for the correction algorithms, we tested them on the entire normal dataset and adversarial image dataset. We tested label correction first and then image correction.

5.3.1 Label Correction

We used JPEG compression, scaling, and their combination for label correction. As shown in Fig. 6, JPEG compression had better performance than scaling for the adversarial images, and their combination produced the best overall performance. Only 1.88% of the normal images was misclassified after "correction" while 89.91% of the adversarial images were corrected. If this correction was performed after detection of adversarial images, about 98.12% of the false positive inputs (normal images misclassified as adversarial ones) would also be corrected, therefore boosting the overall performance.

5.3.2 Image Correction

We tested the fixed QF baseline approach with several QF values from 40 to 90 and the proposed heuristic algorithm with S_{QF} values of *max*, *median*, 2nd*min*, and *min*. In addition, we built a convolutional denoising autoencoder [44] to check whether it is useful for adversarial noise removal. To avoid data overlapping, we trained it on the test set of the

Table 6 Percentage of correct classification of corrected images (both normal and adversarial).

Correction Method	Normal Images	Adversarial Images
QF 90	98.40	77.38
QF 80	**97.17**	**86.24**
QF 70	**95.93**	**88.07**
QF 60	**94.97**	**88.46**
QF 50	93.97	88.02
QF 40	93.12	87.28
Heuristic - *max*	99.65	25.81
Heuristic - *median*	99.55	54.03
Heuristic - 2^{nd}*min*	98.98	72.96
Heuristic - min	**97.52**	**88.98**
Denoising autoencoder	57.77	49.81

ImageNet database using Gaussian noise with $\sigma = 0.1$.

As shown in Table 6, the denoising autoencoder had poor performances on both normal and adversarial images. One explanation is that denoised images have different distributions than normal images; therefore, the DNNs could not correctly classify most of them. For the fixed QF baseline approach, there was a trade-off between performance on normal and adversarial images. The QFs between 60 and 80 are safe choices for achieving balanced performance between the two types of images. With the heuristic algorithm, using $max(S_{QF})$ preserved most of the normal images, but it was the worst at mitigating adversarial noise. Although it sits at the mid-point of the S_{QF} values, using *median* resulted in poor correction of the adversarial images. Using the two remaining values resulted in substantial improvement in mitigating adversarial noise. Using *min* achieved the best balance in performance between normal and adversarial images: 97.52% correct classification for normal images and 88.98% for adversarial images.

Examples of normal images, their corresponding adversarial images, and their corrected versions are shown in Fig. 7. There were no perceived substantial differences between the different versions, meaning that adversarial noise is difficult to notice and that the correction algorithm did not substantially degrade the quality of both the normal and adversarial images. The average PSNR and SSIM between the corrected adversarial images and their original versions in the entire datasets are shown in Fig. 8. The heuristic - *min* approach had higher PSNR and SSIM than the fixed QF approach. The explanation for this is illustrated in Fig. 9, which is a histogram of the QFs selected using the heuristic - *min* approach for both normal and adversarial images. For the normal images, in most cases, a QF of 100 was selected, so their quality was almost completely preserved. For the adversarial images, more than two-thirds of the selected QFs were from 75 to 95, so their quality was also good. Since there are some extreme cases in the adversarial dataset, the correction algorithm could not fully recover their quality, resulting in low values of PSNR and SSIM.

5.4 Considerations

Our experiments demonstrated the effectiveness of using image processing operations for both label correction and image correction. For label correction, using multiple operations with multiple parameter values helped maximize the chance of restoring the original labels. It also provides label information for image correction. The proposed heuristic - *min* algorithm effectively determines which QF is the best for image correction so that image quality is preserved as much as possible, which is better than using JPEG compression with a fixed QF. The correction algorithms can be used independently, for instance, in data cleansing, or in combination with an adversarial image detector to provide multiple outputs: (1) whether the input image is adversarial and (2) if yes, what its true label is.

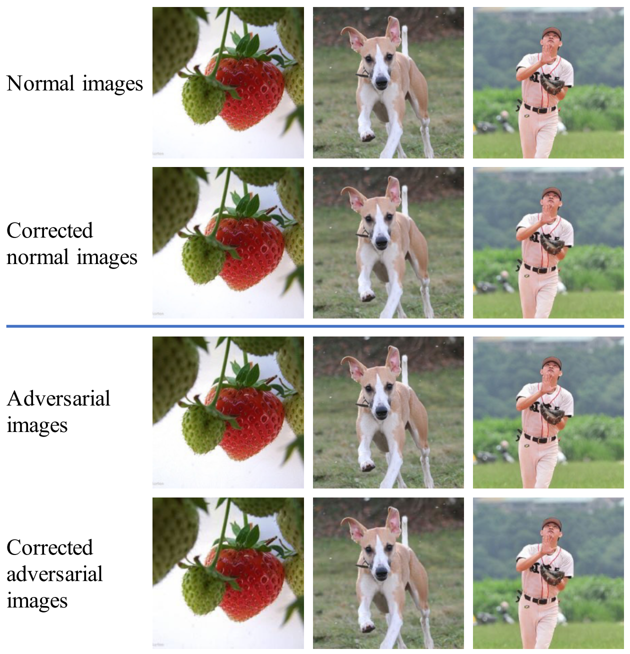


Fig. 7 From top to bottom: Examples of normal images, their corrected versions, their adversarial versions, and the corrected versions of the adversarial images (corrected using heuristic - *min* algorithm).

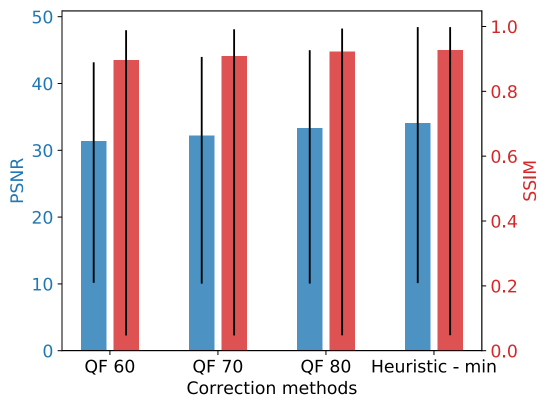


Fig. 8 Average PSNR (blue) and SSIM (red) between corrected adversarial images and corresponding original images. Error bars indicate min and max values.

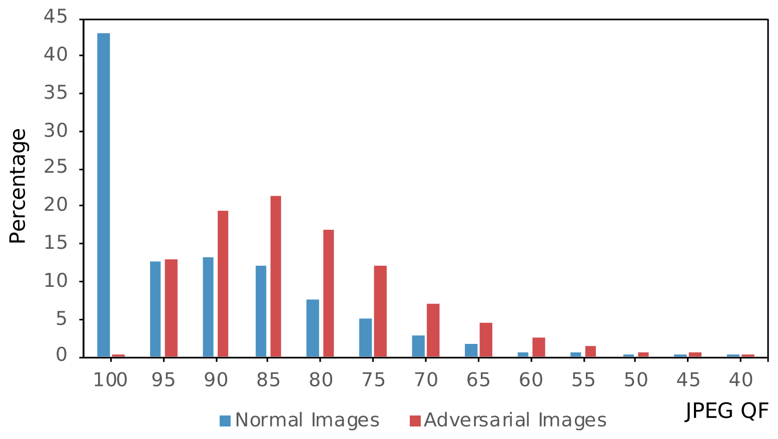


Fig. 9 Histogram of JPEG QF calculated using heuristic - *min* algorithm on both normal and adversarial images.

6. Discussion and Future Work

Our results demonstrated that image processing operations like JPEG compression, scaling, Gaussian blurring, and rotation have different effects on normal and adversarial images depending on the parameter values. Although these operations have negligible effects on normal images, they are effective in mitigating adversarial noise, which is useful for both detection and correction of adversarial images. Different from adversarial training, the proposed detection and correction methods can be performed as a pre-filter to process the data before being processed by a DNN. They are thus applicable to all pre-trained DNNs without the need for re-training. One disadvantage of using multiple image processing operations with multiple parameter values is the computation cost when several processed images need to be classified.

Our results also demonstrated that using statistical features based on image processing operations and using feature squeezing are not as effective as using features automatically extracted by a CNN from raw images, especially the XceptionNet one, which can be trained with a small amount of data. However, the traditional classifiers using handcrafted features are more robust than the CNN-based ones when the attackers add adversarial noise to fool the adversarial image detector (due to the fact that most of the currently implemented image processing operations are non-differentiable). We also found that the adversarial noise created by non-targeted attacks is more general than that created by targeted attacks; therefore, it is important to include non-targeted adversarial images in the training data.

Future work includes testing using more adversarial attacks with multiple noise strengths on larger and more diverse databases. It also includes evaluating additional image processing operations and reducing the computational expense of detection and correction. Another important task is dealing with robust adversarial examples, which is a challenging problem.

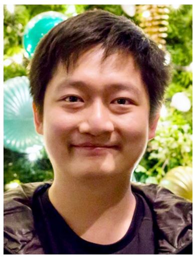
Acknowledgments

This research was supported by JSPS KAKENHI Grants JP16H06302, JP17H04687, JP18H04120, JP18H04112, JP18KT0051, and JP19K22846 and by JST CREST Grants JPMJCR18A6 and JPMJCR20D3, Japan.

References

[1] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, “Intriguing properties of neural networks,” arXiv preprint arXiv:1312.6199, 2013.
[2] A. Kurakin, I. Goodfellow, and S. Bengio, “Adversarial examples in the physical world,” ILCR-W, 2017.
[3] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, F. Tramer, A. Prakash, T. Kohno, and D. Song, “Physical adversarial examples for object detectors,” WOOT, 2018.
[4] C. Sitawarin, A.N. Bhagoji, A. Mosenia, M. Chiang, and P. Mittal, “Darts: Deceiving autonomous cars with toxic signs,” arXiv preprint arXiv:1802.06430, 2018.
[5] A. Boloor, X. He, C. Gill, Y. Vorobeychik, and X. Zhang, “Simple physical adversarial examples against end-to-end autonomous driving models,” arXiv preprint arXiv:1903.05157, 2019.
[6] L. Schönherr, K. Kohls, S. Zeiler, T. Holz, and D. Kolossa, “Adversarial attacks against automatic speech recognition systems via psychoacoustic hiding,” NDSS, 2019.
[7] H. Xu, Y. Ma, H. Liu, D. Deb, H. Liu, J. Tang, and A. Jain, “Adversarial attacks and defenses in images, graphs and text: A review,” arXiv preprint arXiv:1909.08072, 2019.
[8] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, “Universal adversarial perturbations,” CVPR, pp.1765–1773, 2017.
[9] C. Xie, Z. Zhang, Y. Zhou, S. Bai, J. Wang, Z. Ren, and A.L. Yuille, “Improving transferability of adversarial examples with input diversity,” CVPR, pp.2730–2739, 2019.
[10] H. Zhang, T. Zheng, J. Gao, C. Miao, L. Su, Y. Li, and K. Ren, “Data poisoning attack against knowledge graph embedding,” IJCAI, 2019.
[11] K. Grosse, P. Manoharan, N. Papernot, M. Backes, and P. McDaniel, “On the (statistical) detection of adversarial examples,” arXiv preprint arXiv:1702.06280, 2017.
[12] X. Li and F. Li, “Adversarial examples detection in deep networks with convolutional filter statistics,” CVPR, pp.5764–5772, 2017.
[13] Z. Gong, W. Wang, and W.S. Ku, “Adversarial and clean data are not twins,” arXiv preprint arXiv:1704.04960, 2017.

[14] J.H. Metzen, T. Genewein, V. Fischer, and B. Bischoff, "On detecting adversarial perturbations," ICLR, 2017.
[15] K. Lee, K. Lee, H. Lee, and J. Shin, "A simple unified framework for detecting out-of-distribution samples and adversarial attacks," NIPS, pp.7167–7177, 2018.
[16] X. Ma, B. Li, Y. Wang, S.M. Erfani, S. Wijewickrema, G. Schoenebeck, D. Song, M.E. Houle, and J. Bailey, "Characterizing adversarial subspaces using local intrinsic dimensionality," ICLR, 2018.
[17] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," NDSS, 2018.
[18] B. Liang, H. Li, M. Su, X. Li, W. Shi, and X. Wang, "Detecting adversarial image examples in deep neural networks with adaptive noise reduction," IEEE Transactions on Dependable and Secure Computing, vol.18, no.1, pp.72–85, 2018.
[19] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," SP, pp.582–597, IEEE, 2016.
[20] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," ICLR, pp.274–283, 2018.
[21] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," ICLR, 2018.
[22] T. Pang, C. Du, Y. Dong, and J. Zhu, "Towards robust detection of adversarial examples," NeurIPS, pp.4579–4589, 2018.
[23] C. Guo, M. Rana, M. Cisse, and L. Van Der Maaten, "Countering adversarial images using input transformations," ICLR, 2018.
[24] A. Prakash, N. Moran, S. Garber, A. DiLillo, and J. Storer, "Protecting jpeg images against adversarial attacks," Data Compression Conference, pp.137–146, IEEE, 2018.
[25] O. Taran, S. Rezaeifar, T. Holotyak, and S. Voloshynovskiy, "Machine learning through cryptographic glasses: combating adversarial attacks by key-based diversified aggregation," EURASIP journal on information security, vol.2020, pp.1–18, 2020.
[26] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A.C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," IJCV, vol.115, no.3, pp.211–252, 2015.
[27] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," ICLR, 2015.
[28] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," CVPR, pp.770–778, 2016.
[29] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," NIPS-W, 2017.
[30] J. Rauber, W. Brendel, and M. Bethge, "Foolbox: A python toolbox to benchmark the robustness of machine learning models," arXiv preprint arXiv:1707.04131, 2017.
[31] I.J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," arXiv preprint arXiv:1412.6572, 2014.
[32] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," CVPR, pp.2574–2582, 2016.
[33] U. Jang, X. Wu, and S. Jha, "Objective metrics and gradient descent algorithms for adversarial examples in machine learning," ACSAC, pp.262–277, ACM, 2017.
[34] R. Alaifari, G.S. Alberti, and T. Gauksson, "ADef: an iterative algorithm to construct adversarial deformations," ILCR, 2019.
[35] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z.B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," EuroS&P, pp.372–387, IEEE, 2016.
[36] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," SP, pp.39–57, IEEE, 2017.
[37] A. Hore and D. Ziou, "Image quality metrics: Psnr vs. ssim," ICPR, pp.2366–2369, IEEE, 2010.
[38] C. Cortes and V. Vapnik, "Support-vector networks," Machine learning, vol.20, no.3, pp.273–297, 1995.
[39] T.K. Ho, "Random decision forests," ICDAR, pp.278–282, IEEE, 1995.
[40] R.O. Duda, P.E. Hart, D.G. Stork, et al., Pattern classification, Wiley, New York, 1973.
[41] D.W. Ruck, S.K. Rogers, M. Kabrisky, M.E. Oxley, and B.W. Suter, "The multilayer perceptron as an approximation to a bayes optimal discriminant function," IEEE Transactions on Neural Networks, vol.1, no.4, pp.296–298, 1990.
[42] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," CVPR, pp.1251–1258, 2017.
[43] A. Rossler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Niessner, "FaceForensics++: Learning to detect manipulated facial images," ICCV, pp.1–11, 2019.
[44] I. Goodfellow, Y. Bengio, and A. Courville, Deep learning, MIT press, Cambridge, Massachusetts, 2016.



Huy H. Nguyen received a B.S. degree in Information Technology from VNUHCM - University of Science, Ho Chi Minh City, Vietnam, in 2013. He is currently pursuing a Ph.D. degree in computer science at the Graduate University for Advanced Studies (SOKENDAI), Kanagawa, Japan. His current research interests include security and privacy in biometrics and machine learning.

Minoru Kuribayashi received B.E., M.E., and D.E. degrees from Kobe University, Japan, in 1999, 2001, and 2004. He was a research associate from 2002 to 2007 and an assistant professor from 2007 to 2015 at Kobe University. Since 2015, he has been an associate professor in the Graduate School of Natural Science and Technology, Okayama University. His research interests include multimedia security, digital watermarking, cryptography, and coding theory. He serves as an associate editor for JISA and IEICE and as a vice chair of the APSIPA Multimedia Security and Forensics Technical Committee. He is a member of the Information Forensics and Security Technical Committee of the IEEE Signal Processing Society. He received the Young Professionals Award from the IEEE Kansai Section in 2014 and the Best Paper Award at IWDW 2015 and 2019. He is a senior member of the IEEE and IEICE.

Junichi Yamagishi received a Ph.D. from the Tokyo Institute of Technology in 2006. He was a senior research fellow in the Centre for Speech Technology Research (CSTR) at the University of Edinburgh, U.K., from 2006 to 2013. He is currently a professor at the National Institute of Informatics in Japan. He was awarded the Itakura Prize by the Acoustic Society of Japan, the Kiyasu Special Industrial Achievement Award by the Information Processing Society of Japan, and the Young Scientists' Prize by the Ministry of Education, Science and Technology, the JSPS prize, and the DOCOMO prize in 2010, 2013, 2014, 2016, and 2018, respectively. He served as a co-organizer for the bi-annual ASVspoof special sessions at INTERSPEECH 2013-9, a co-organizer for the bi-annual Voice conversion challenge at INTERSPEECH 2016 and Odyssey 2018, an organizing committee member for the 10th ISCA Speech Synthesis Workshop 2019, and a technical program committee member for IEEE ASRU 2019. He also served as a member of the IEEE Speech and Language Technical Committee, as an associate editor of the IEEE/ACM TASLP, and as a lead guest editor for the IEEE JSTSP SI on Spoofing and Countermeasures for Automatic Speaker Verification. He is currently a PI of the JST-CREST and ANR supported VoicePersona project. He also serves as the chair of the ISCA SynSIG and as a senior area editor of the IEEE/ACM TASLP.

Isao Echizen received B.S., M.S., and D.E. degrees from the Tokyo Institute of Technology, Japan, in 1995, 1997, and 2003, respectively. He joined Hitachi, Ltd. in 1997 and until 2007 was a research engineer in the company's systems development laboratory. He is currently a director and a professor of the Information and Society Research Division, the National Institute of Informatics (NII), and a professor in the Department of Information and Communication Engineering, Graduate School of Information Science and Technology, The University of Tokyo, Japan. He was a visiting professor at Tsuda University, Japan, a visiting professor at the University of Freiburg, Germany, and a visiting professor at the University of Halle-Wittenberg, Germany. He is currently engaged in research on multimedia security and multimedia forensics. He currently serves as a research director of the CREST FakeMedia project, Japan Science and Technology Agency (JST). He received the Best Paper Award from the IPSJ in 2005 and 2014, the Fujio Frontier Award and the Image Electronics Technology Award in 2010, the One of the Best Papers Award from the Information Security and Privacy Conference in 2011, the IPSJ Nagao Special Researcher Award in 2011, the DOCOMO Mobile Science Award in 2014, the Information Security Cultural Award in 2016, and the IEEE Workshop on Information Forensics and Security Best Paper Award in 2017. He was a member of the Information Forensics and Security Technical Committee and the IEEE Signal Processing Society. He is the Japanese representative on IFIP TC11 (Security and Privacy Protection in Information Processing Systems), a member-at-large of the APSIPA Board of Governors, and an editorial board member of the IEEE Transactions on Dependable and Secure Computing and the EURASIP Journal on Image and Video Processing.