Detecting Adversarial Examples by Input Transformations, Defense Perturbations, and Voting

Federico Nesti, Alessandro Biondi, *Member, IEEE*, and Giorgio Buttazzo, *Fellow, IEEE*

***Abstract*—Over the past few years, convolutional neural networks (CNNs) have proved to reach superhuman performance in visual recognition tasks. However, CNNs can easily be fooled by adversarial examples (AEs), i.e., maliciously crafted images that force the networks to predict an incorrect output while being extremely similar to those for which a correct output is predicted. Regular AEs are not robust to input image transformations, which can then be used to detect whether an AE is presented to the network. Nevertheless, it is still possible to generate AEs that are robust to such transformations. This article extensively explores the detection of AEs via image transformations and proposes a novel methodology, called *defense perturbation*, to detect robust AEs with the same input transformations the AEs are robust to. Such a *defense perturbation* is shown to be an effective counter-measure to robust AEs. Furthermore, multinetwork AEs are introduced. This kind of AEs can be used to simultaneously fool multiple networks, which is critical in systems that use network redundancy, such as those based on architectures with majority voting over multiple CNNs. An extensive set of experiments based on state-of-the-art CNNs trained on the Imagenet dataset is finally reported.**

***Index Terms*—Adversarial defense, adversarial examples (AEs), convolutional neural network (CNN), deep neural network, input transformation, redundant neural networks.**

I. Introduction

DURING the past few years, convolutional neural networks (CNNs) have been used in many fields with outstanding, and sometimes superhuman, performance [1]–[3]. At the same time, a lot of research has been devoted to the robustness of such models, often focusing on adversarial examples (AEs) [4], [5].

AEs are maliciously crafted inputs (in this case, images) that have the power to fool a neural network by forcing its prediction toward an erroneous class, by slightly changing the intensity values of the pixels, keeping almost the same digital representation. From the perspective of a human observer, a typical AE has the same visual appearance as the original image.

Manuscript received July 27, 2020; revised January 27, 2021 and June 10, 2021; accepted August 11, 2021. *(Corresponding author: Federico Nesti.)*

The authors are with the Department of Excellence in Robotics and AI, Scuola Superiore Sant'Anna, 56127 Pisa, Italy (e-mail: f.nesti@santannapisa.it).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2021.3105238.

Digital Object Identifier 10.1109/TNNLS.2021.3105238

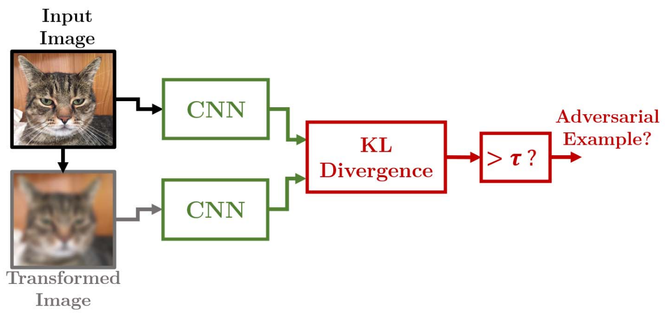


Fig. 1. BASELINE architecture used to detect standard AEs.

AEs are a serious concern for the safety and security of systems based on artificial intelligence (AI). For instance, they could be used to attack a neural network for object recognition in an autonomous (or even semiautonomous) vehicle, possibly causing a catastrophic consequence [6]–[9].

These facts motivate the search for an effective defense against AEs. For instance, Guo *et al.* [10] showed that standard[1] AEs are not robust to input transformations, such as translation, rotation, and other input changes. These findings suggest that standard AEs can be detected by measuring how the network prediction changes when an input image is replaced with the same one processed with a given transformation. The two network predictions can then be compared using the Kullback–Leibler (KL) divergence [11], so that an input image is considered to be adversarial if the two predictions are "distant" from each other and nonadversarial if the two predictions are "close" to each other. A binary classification can then be obtained by applying a threshold to the output of the KL module. This approach can easily be implemented using the architecture illustrated in Fig. 1, referred to as BASELINE detection architecture.

Input transformations are attractive because they are simple, require a limited computational cost (thus can be performed at run time), and do not need any training procedure. Nevertheless, they suffer from two major problems: 1) they might not have a good detection performance (also due to the accuracy degradation they may cause) and 2) it has been shown [12]

[1]In this article, standard AEs refer to those AEs crafted without considering image transformations, using the classical formulation presented in Section II by (1).

that it is still possible to construct AEs that are *robust* to input transformations.

A. This Article

This work addresses the two problems mentioned above. To evaluate how different input transformations affect the performance of the detection system and the accuracy of the network, an extensive experimental campaign is reported both for adversarial and nonadversarial images. To the best of our knowledge, a similar experimental study has never been presented in the literature.

To cope with the second problem, different new methods and architectures are proposed to include a novel counter-measure for detecting robust AEs. The counter-measure is based on the assumption that the defender is aware that the attacker knows how to craft robust AEs and also has knowledge about the transformations that are used. To counteract this kind of attacks, a new defense method, called *defense perturbation*, is introduced to "convert" robust AEs into nonrobust ones. Such a defense perturbation is generated from robust AEs, similar to how universal adversarial perturbations [13] are generated.

As a further evolution, this work considers a multinetwork architecture composed of three different state-of-the-art CNNs for image recognition (trained on ImageNet [14]), which are combined by means of a majority voting algorithm (two out of three). For instance, this approach was proposed by Biondi *et al.* [15] as a solution for adopting neural networks in safety-critical control systems. Although there exist methods to transfer AEs between different models and architectures [16], multinetwork AEs are introduced as a new kind of AEs that are capable of fooling multiple networks simultaneously without applying any network-specific perturbation.

Contribution and Paper Structure: In summary, this article makes the following novel contributions.

1) It reports an extensive experimental evaluation on the detection capabilities of input transformations.
2) It presents a methodology for setting up a counter-measure against robust AEs.
3) It introduces multinetwork AEs to systematically fool multiple networks at once.
4) The proposed methods are finally combined to design an effective architecture for detecting robust AEs.

The rest of this article is organized as follows. Section II introduces the problem and the notation; Section III provides a brief overview of background and related work; Section IV describes the proposed approaches; Section V presents the experimental results; and Section VI states the conclusions.

II. System and Threat Model

This article considers CNNs for image recognition. Let $\mathcal{X} = [0,1]^{h\times w\times c}$ be the image space (of dimensions w, h, and c, namely, the width, height, and the number of channels of the image, respectively). A CNN behaves as a function $f(\cdot)\colon\mathcal{X} \to [0,1]^n$ that takes as input an image of fixed dimensions and outputs a discrete probability distribution vector with dimension n equal to the number of classes considered for the classification problem. The class predicted by a neural network classifier is represented by the function $\hat{f}(\cdot) = \arg\max f(\cdot)$.

An adversarial perturbation can be modeled as a tensor $r \in [-\epsilon,\epsilon]^{h\times w\times c}$, where $\epsilon \in (0,1)$ (typically small) is a parameter typically named adversarial strength. Given a source image $x \in \mathcal{X}$, an AE is then an image $x + r \in \mathcal{X}$ such that if $\hat{f}(x) = t$, where t is the correct target class of the image x, then $\hat{f}(x+r) = t_{\text{adv}}$, where $t_{\text{adv}} \neq t$ is the adversarial target class.

AEs can be characterized by: 1) the knowledge level of the attacker; 2) the target specificity; and 3) the similarity metric used to minimize the distance of an AE from the source image. The AEs considered in this article are as follows.

1) *White-Box:* The attacker has perfect knowledge of the structure of the network and its parameters (this is the strongest type of attack).
2) *Targeted:* They are crafted to force the prediction of the network to a specific class.
3) Generated using the L_2 **norm**, i.e., the Euclidean norm, to express the distance of an AE from the source image.

An AE of this type can be crafted by optimizing

$$\min_r \left[\mathcal{L}(f(x+r), t_{\text{adv}}) + k\|r\|_2^2\right] \tag{1}$$

where $\mathcal{L}$ in the above equation is a loss function expressing the distance between the target t_{adv} and the output distribution of the network, and k is a constant that reflects how much the magnitude of the perturbation is weighted in minimization. Typically, the loss function $\mathcal{L}$ is a cross-entropy, but it can also assume other more complex forms (e.g., as in the Carlini–Wagner (CW) attack [17], described in Section III). The optimization is iteratively performed with a stochastic gradient descent approach for a certain number of epochs: this optimization strategy is used throughout the whole article for the generation of AEs. When searching for the minimum-perturbation AE, a further optimization is usually performed to find the optimal k. This fine-grained optimization is out of the scope of this work, and hence k is fixed to 0.01 for all the AEs.

The AEs crafted with the procedure described above are sensitive to input transformations and are referred to as standard AEs. Conversely, robust AEs are generated with a slightly modified optimization process based on the architecture illustrated in Fig. 2, which is named ROBUST_ADV_GEN. Let $\{g_j(x;\theta_j),\ j = 1,\ldots,N\}$ be a set of N transformations of the input image (e.g., translation, rotation), each of which depends on a parameter θ_j. AEs that are robust to each of these transformations can then be generated by minimizing

$$\min_r \left[\mathcal{L}(f(x+r), t_{\text{adv}}) + \sum_{j=1}^{N} \mathcal{L}(f(g_j(x+r;\theta_j)), t_{\text{adv}}) + k\|r\|_2^2\right]. \tag{2}$$

Most image transformations depend on a parameter θ_j that varies in a known range. At each iteration step of the optimization procedure, the parameters of the transformations are uniformly sampled from their corresponding ranges, hence

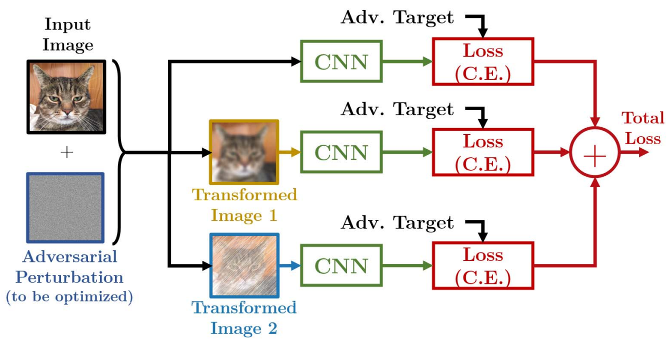


Fig. 2. ROBUST_ADV_GEN architecture used to craft robust AEs.

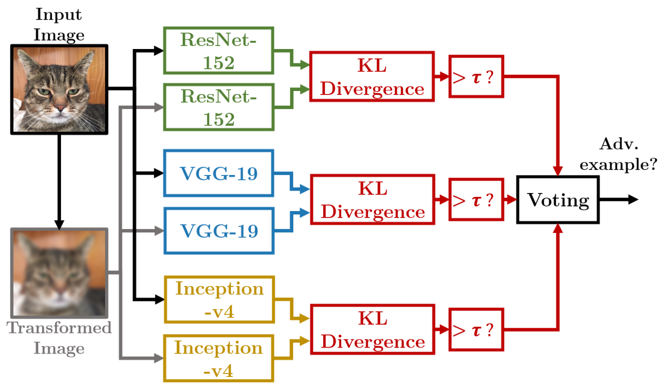


Fig. 3. VOTING_BASELINE architecture used in this work.

generating AEs that are robust to a wide spectrum of configurations of image transformations. Details on the generation of robust AEs used in this work are reported in Section V.

Since a deep neural network is naturally prone to classification errors, especially when considering gray cases (i.e., previously unseen, possibly harmful inputs), an ensemble of networks can be used [15] to mitigate the errors of a single network, by combining the outputs with a voting algorithm. The resulting architecture is illustrated in Fig. 3 and consists of applying a voting algorithm (such as majority voting) over M different CNNs. This architecture is referred to as VOTING_BASELINE. This article only considers $M = 3$ networks, which are described in detail in Section V. Standard AEs, as those described above, result adversarial for a single network. With this kind of voting architecture, an AE that fools a single network is a less dangerous threat, since the voting algorithm will cover for that mistake with the predictions from the other two networks.

Note that single-network AEs may also fool other networks. However, to assess the detection capabilities of such system, it is more convenient to craft images that are adversarial for the three networks simultaneously. Such AEs are also adversarial for each single network and can be used to evaluate the detection performance of input transformations for each single network. For this reason, multinetwork AEs are introduced. Given a set of M classifiers, each denoted by $f_i(\cdot)$, $i = 1, \dots, M$, and a set of N transformations of the input image, denoted as $\{g_j(x; \theta_j), j = 1, \dots, N\}$, multinetwork robust AEs can be crafted by minimizing the following extension of (2):

$$\min_r \left[\sum_{i=1}^{M} \mathcal{L}(f_i(x+r), t_{\text{adv}}) + \sum_{j=1}^{N} \sum_{i=1}^{M} \mathcal{L}(f_i(g_j(x+r;\theta)), t_{\text{adv}}) + k\|r\|_2^2 \right]. \quad (3)$$

If generated in this way, an image results to be adversarial not only for all the M networks but also for all the N transformations.

These AEs have been used for experimental evaluation and proved to be more difficult to detect than single-networks AEs, as they are adversarial for each of the selected networks. The methodology presented above extends a series of state-of-the-art attack methods, which are reviewed next.

III. Background and Related Work

The literature concerning AEs has been growing exponentially over the past years, and several different attack and defense methods have been proposed. This section reviews the most common attacks and defenses, highlighting how this work is positioned within the published literature.

A. Attacks

1) L-BFGS Attack: Szegedy *et al.* [4] first introduced AEs against deep neural networks. The approach is the one described in (1). The loss function they used is a cross-entropy.

2) CW L2 Attack: Carlini and Wagner [17] proposed a set of more complex loss functions (and different threat models), among which the most relevant is $\mathcal{L} = \max(0, \max\{Z(x+r)_k : k \neq t_{\text{adv}}\} - Z(x+r)_{t_{\text{adv}}})$, where $Z(x)$ is the output of the logits layer (i.e., before the softmax layer), and $Z(x)_k$ is the logit value corresponding to class k. The effect of using this loss is that during the optimization process, this term falls to zero as soon as the predicted class matches the adversarial target, without considering the confidence associated with that prediction. This allows minimizing the adversarial perturbation only as soon as the first term drops to zero.

3) FGSM Attack: The fast gradient sign method was introduced by Goodfellow *et al.* [18], and it is a noniterative, untargeted method for adversarial examples' generation. It is sufficient to compute the gradient with respect to the image of a certain loss function $\nabla_x \mathcal{L}(f(x), t)$ (usually cross-entropy). The adversarial perturbation is then generated as $\epsilon\, \text{sign}(\nabla_x \mathcal{L}(f(x), t))$. Although this type of AE is one of the most popular for its simplicity and fast generation, the focus of this article is on AEs that are able to simultaneously fool multiple networks, and targeted AEs are more suited for this purpose. Hence, this kind of attack is not considered in this work since, for an accurate assessment of the detection capabilities of this multinetwork system, it would require an accurate filtering of the perturbations that resulted adversarial for the ensemble of the three networks.

4) Robust AEs: AEs can be made robust (in expectation) to a certain transformation distribution T [12] by randomizing the parameters of the transformation during optimization of adversarial perturbation. This kind of AEs are, by construction, not easily detectable with the BASELINE architecture (Fig. 1).

5) Universal Adversarial Perturbation: This kind of attack crafts an image-agnostic adversarial perturbation [13], which is generated to result adversarial for a set of different images belonging to different classes. A universal perturbation has the property of making regular images become AEs, even when considering images that were not used for the optimization of the perturbation. Although this kind of attack is not considered in this work because it is weaker than the ones presented above, it is worth citing it since it inspired the defense perturbation presented in this article, as described in Section IV.

6) Other Attacks: The literature presents many other different attacks that are not considered in this article for space limitations. Among these, the most relevant to us are DeepFool [19] and JMSA [20].

B. Defenses

The defenses proposed in the literature can be divided into three categories, briefly described in the following.

1) Modifying Data: The defenses that fall in this category are the ones that modify the input data at run time to defend from AEs. Possible approaches are based on data compression and filtering [10], [11], [21] or data randomization [22]. The underlying idea is similar to the one presented in this work. However, the objective of this work is to detect AEs rather than predicting the correct class; moreover, it presents an extensive experimental evaluation of the most common image transformations.

2) Modifying Model: This type of defenses includes the ones that act on the classifier to prevent attacks. For instance, regularization (i.e., adding a penalty term to the loss function during training to improve generalization) is a widely diffused method. Adversarial training [4] was the first defense to be proposed. It consists of enlarging the original dataset with a set of AEs, which is used to retrain the network. It has been criticized because it just shifts the problem to find new AEs. Among others, it is worth citing defensive distillation [23], which trains a second, simpler network over soft targets (i.e., the output values of the original network) and deep contractive networks [24], which improve the defense performance of denoising convolutional autoencoders.

3) Auxiliary Tools: These approaches make use of an external tool to defend or detect AEs. Among these, Defense-GAN [25] and MagNet [26] present good performance.

Many other works are present in the literature. The interested reader may refer to the reviews presented by Yuan *et al.* [27] and Xu *et al.* [28].

C. This Work

Although many of the defenses listed above show good performance, the work presented in this article focuses on data modification techniques, since they are simple and computationally cheap to detect AEs, without changing or retraining the neural network. The proposed detection system can also be seen as an external tool aimed at detecting AEs, but without considering complex models (such as generative adversarial networks or additional neural networks) as previous works do.

Previous work used input transformations [10], [11], [21], [22] but, to the best of our records, none of them presented an extensive experimental evaluation to determine which transformations are the most effective in terms of detection rate. Furthermore, still to our records, no counter-measure for robust AEs has been presented before. This latter aspect is crucial, since any defense based on differentiable input transformations (as in the work of Xie *et al.* [22]) can be completely fooled by robust AEs.

This article presents a novel method to detect robust AEs with input transformations, which, to our records, was never proposed in the literature. An experimental comparison with similar state-of-the-art methods is performed in Section V.

IV. Detection Algorithms

The objective of this work is to detect different kinds of AEs and evaluate the detection performance of four different architectures. The AEs considered in this work can be classified into: 1) standard, i.e., those generated by the attacks reviewed at the beginning of Section III), and 2) robust ones, i.e., those that cannot be detected by input transformations. Orthogonally, they can also be classified as: 1) single-network, i.e., those generated to attack just a single CNN, and 2) multinetwork, i.e., those that are capable of attacking multiple CNNs subject to majority voting. Note that this taxonomy allows defining four classes of AEs.

The BASELINE and VOTING_BASELINE architectures (Figs. 1 and 3) are suited for the detection of standard AEs only, as they fail in detecting robust ones. The former is suited for single-network AEs, while the latter for multinetwork ones.

Other two architectures are proposed to detect both standard and robust AEs by leveraging a defense perturbation that is assumed not be known by the attacker. The ENHANCED architecture is designed to detect single-network AEs, while the VOTING_ENHANCED architecture is designed to detect multinetwork AEs by combining voting and defense perturbations.

A. Baseline Detection Architectures

The BASELINE architecture has been considered to evaluate the detection performance of a set of input transformations. The input image $x \in \mathcal{X}$ (that might be an AE) is transformed by means of the chosen input transformation, producing x'. Then both the images are fed into the network, and the distance between the two output probability distributions is computed using the KL divergence. Since the KL divergence is not symmetric, it is not a proper distance. Hence, the actual distance is computed as the maximum KL of the two possible combinations

$$D(f(x), f(x')) = \max\{\mathrm{KL}(f(x), f(x')), \mathrm{KL}(f(x'), f(x))\}. \quad (4)$$

The computed distance is then thresholded to classify the image as adversarial or not. Note that this is different from the approach proposed by Kantaros *et al.* [11], where the minimum of the two KL divergences is used. This is because all our experiments showed that the detection algorithm obtains a significantly better classification accuracy using the maximum, rather than the minimum. This can be explained by considering that the maximum is a more conservative estimate of the distance and has more impact on AEs, as they are characterized by more asymmetrical KL divergences than regular images.

The VOTING_BASELINE architecture is also considered to assess the performance of a multinetwork ensemble in detecting AEs. In particular, three pretrained state-of-the-art CNNs are used, as shown in Fig. 3. The three networks are subject to majority voting, which is a common algorithm for fault detection and exclusion and in general for fault-tolerant systems. Here, it is used to decide whether an input image is adversarial for the majority of the networks, meaning that as long as there are two networks detecting potential AEs, the voting algorithm classifies the input as adversarial. The detection performance of such an architecture is discussed in Section V.

These two architectures are based on image transformations only and both fail in detecting robust AEs. Section IV-B introduces new architectures to overcome this limitation.

B. Enhanced Detection Architectures: Counter-Measures Against Robust AEs

AEs have great fooling power. During the experimental campaign conducted for this work, it was possible to find AEs that were robust to any kind of differentiable input transformation and even robust to any possible combination of four consecutive transformations, chosen between three different transformations. Also, randomization does not help: during the experiments, it was possible to find AEs that are robust to additive Gaussian noise.

Since AEs have such great fooling power, the key idea of this work is to use the same crafting procedure to generate a defense perturbation, which is basically a mask of pixels added to the input image at run time. This perturbation is optimized to make robust AEs sensitive again to input transformations.

The idea behind the generation of the defense perturbation is inspired by the one used to compute universal adversarial perturbations [13]. This generation process is aimed at producing an image-agnostic pixel mask that is capable of removing the robustness to specific transformations, once it is added to a robust AE. The defense perturbation is generated using the architecture shown in Fig. 4, which is named DEFENSE_GEN, and takes as input a dataset constructed as follows. Given a set of L original images and a set of N input transformations $\{g_j(x, \theta_j), j = 1, \dots, N\}$, a robust AE is generated for each of the L images and for each of the N input transformations. All images, i.e., both the original and the adversarial ones, are then added to a dataset. For each of the images $\tilde{x}$ in such a dataset, the following optimization procedure is performed for $k_{\max}$ steps:

$$\min_d \left[\mathcal{L}(f(\tilde{x} + d), \tilde{t}) + \sum_{j=1}^{N} \mathcal{L}(f(g_j(\tilde{x} + d; \theta_j)), t) + k\|d\|_2^2 \right] \tag{5}$$

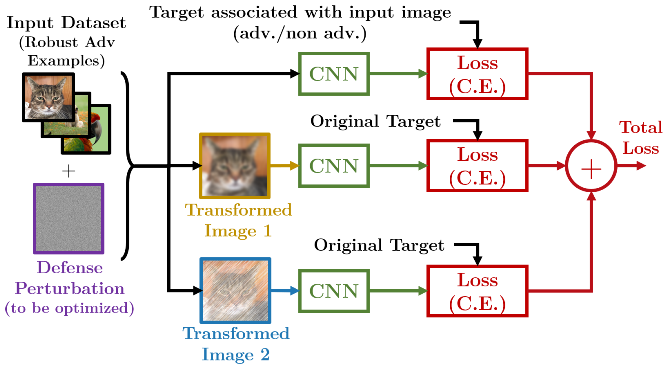


Fig. 4. DEFENSE_GEN architecture used to craft the defense perturbation.

where t is the original (nonadversarial) target and $\tilde{t}$ is the target associated with image $\tilde{x}$. If $\tilde{x}$ is an AE, $\tilde{t}$ is the adversarial target, whereas if $\tilde{x}$ is a regular image, $\tilde{t}$ is the original target. This iterative procedure outputs a mask d that, when added to a robust AE $\tilde{x}$, produces a new image that is no longer robust to input transformations, and therefore can be used to detect AEs, just as in the standard case.

This property comes from the fact that the generation process of the defense perturbation follows the same logic for generating robust AEs, but reversed. The ROBUST_ADV_GEN [(2)] optimizes a perturbation, added to the original image, which pushes the prediction of the network toward the same label for both the nontransformed and the transformed images. Conversely, DEFENSE_GEN [(5)] does the opposite: it pushes the prediction of the transformed image toward the original label (as it actually is with standard AEs), and the nontransformed image toward the corresponding label. Since the input images are both AEs and non-AEs, the corresponding label would be the adversarial label for AEs, and the original label for non-AEs. In this way, the defense perturbation learns the "opposite average robust perturbation" that, added to a robust AE, removes its robustness.

If the optimization described in (2) is used to generate an AE for each network and for each transformation, then the minimization of (5) is aimed at finding the perturbation that is able to do the opposite, which is inhibiting the patterns that make the AEs robust to input transformations.

The downside of this method is that the generation of a dataset of robust AEs and the training of a defense perturbation are computationally expensive. Another problem of this method is that it is suited only for robust AEs, while it has been experimentally found that it has the property of making standard AEs robust. This happens because the role of the defense perturbation is to counteract the adversarial perturbation that makes an AE robust to input transformations. When adding the defense perturbation to a standard AE, it introduces a perturbation that transforms standard AEs into robust ones, hence having a negative effect for the detection architecture.

Nevertheless, standard AEs are sensitive to any kind of perturbation, meaning that just the application of the mask is sufficient to discriminate AEs. Given these observations, the detection of both standard and robust AEs can be performed using the ENHANCED architecture illustrated in Fig. 5,

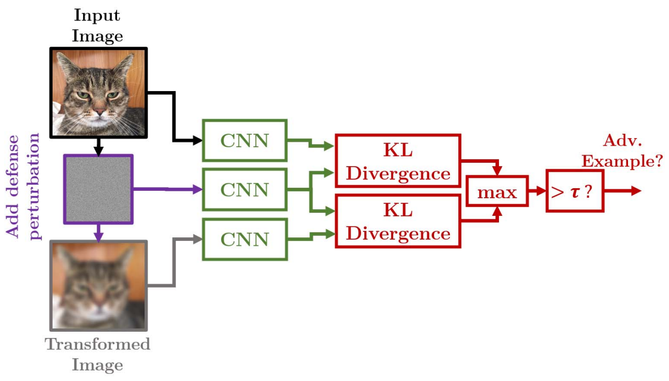


Fig. 5. ENHANCED architecture for detecting robust AEs.

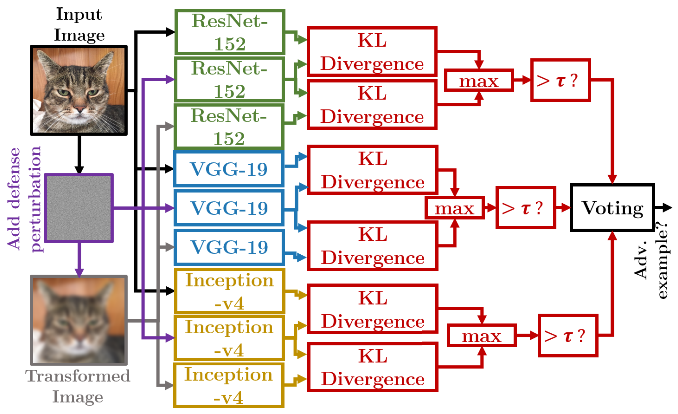


Fig. 6. VOTING_ENHANCED detection architecture for detecting robust AEs.

where the mask is used as a first transformation (to detect standard AEs and to make robust AEs sensitive to input transformations), and a second transformation is used to detect robust AEs. Each of the two KL divergences is computed by (4), taking the maximum between the two results. With this counter-measure, robust AEs can be detected with simple input transformations at run time and three inference operations with the same network.

Note that since the defense perturbation d is a differentiable transformation, an attacker could generate AEs that are robust to that specific mask. Being the mask not known in advance, the attacker might follow the same procedure described in this section to generate another defense mask, which has the same defensive properties, and then use it to craft AEs that are robust to that mask. However, as one may expect, experiments show that different datasets used as input to the above procedure lead to different defense perturbation masks. Hence, the attacker should also know the exact data distribution used to generate the defense perturbation, which can easily be kept secret.

A further evolution of the detection architecture is finally proposed by combining the ENHANCED architecture with majority voting, hence obtaining the architecture illustrated in Fig. 6, which is named VOTING_ENHANCED. Under this latter architecture with voting over M networks, the defense perturbation is generated as for the ENHANCED architecture but optimizing

$$\min_d \left[\sum_{i=1}^{M} \mathcal{L}(f_i(\tilde{x}+d), \tilde{t}) + \sum_{j=1}^{N} \sum_{i=1}^{M} \mathcal{L}(f_i(g_j(\tilde{x}+d;\theta_j)), t) + k\|d\|_2^2 \right]. \quad (6)$$

The defense perturbation generated in this way results to be effective for multinetwork robust AEs.

V. Experimental Results

This section presents the results obtained from an extensive experimental evaluation carried out to test the performance of the approaches proposed in this article to detect AEs. After describing the experimental setting, the input transformations used for the evaluations are introduced, together with each characteristic parameter and their range. Then, the effect of each transformation on the accuracy of the networks is reported. The performance of the different detection methods is presented by first considering the BASELINE and VOTING_BASELINE architectures, and then the ENHANCED and VOTING_ENHANCED architectures, which include the defense perturbation.

A. Experimental Setting

Three CNNs were selected for the experiments: 1) VGG-19 [29]; 2) Resnet-v2-152 [30]; and 3) Inception-v4 [31]. These networks were chosen as they are among the most used, top-performing CNNs for visual recognition. They were pretrained on the ImageNet dataset and downloaded from the tf-slim library [32].

All the experiments presented in this article were performed on an Nvidia DGX station, composed of 4 Tesla-v100 GPUs, with 32 GB of RAM each. The code for the experiments was written in Python 3 using Tensorflow 1.15 (configured to use GPUs). The ImageNet dataset was downloaded from Kaggle. Under this setting, the generation of a single multinet robust AE took 250 s.

B. Input Transformations

The image transformations considered in this work are several and can be divided into three different groups. Each transformation comes with a parameter that varies in a certain range.

1) Topological Transformations: They include the basic affine transformations that can be expressed in the form of a warping matrix $T \in \mathrm{IR}^{3\times3}$. Each of such transformations can be defined as $v' = Tv$, where $v = [u, v, 1]^T$ and $v' = [u', v', 1]^T$ represent the homogeneous coordinates of a pixel of the original and the transformed images, respectively. The transformations of this type considered in this work are as follows.

1) *Translation:* Horizontal and vertical translation parameters are combined into a single diagonal translation to simplify the parametric search. Range: [−45 px, +45 px].
2) *Rotation:* Range: [−25°, 25°].

3) *Horizontal Shear:* It is expressed by the transformation $T = \begin{bmatrix} 1 & s_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$, where s_x is the shear parameter with range $[-0.175, 0.175]$.
4) *Scale:* It is expressed by the transformation $T = \text{diag}([s, s, 1])$, where s is the scale parameter with range $[0.875, 1.175]$.
5) *Mirror:* (no parameter).

2) Appearance Transformations: They include those transformations that change the appearance of the image, with no topological changes. The ones considered in this work are as follows.

1) *Average Blur:* Range: [2×2 kernel, 6×6 kernel]).
2) *Brightness Change:* It adds the value of the parameter to the intensity of the image, pixelwise. Range: $[-35, +35]$.
3) *Contrast Change:* It scales the pixelwise intensity of the image by the parameter value. Range: $[0.875, 1.125]$.

3) Special: They include two uncategorized transformations.

1) *Bit Depth Reduction:* It changes the number of bits used to represent the intensity of the pixels. Range: [4 bits, 7 bits].
2) *Gaussian Noise:* It adds a Gaussian noise to each pixel of the input image (no parameters[2]).

It is worth noting that all the transformations used are differentiable with respect to the input image. This is crucial, since this work assumes white-box robust AEs. Other more complex nondifferentiable transformations were used in the literature, but they are out of the scope of this work, since a black-box approach should be used to craft AEs that are robust to those transformations.

C. Performance Metrics

AEs' detection is a binary classification problem. A classical way to evaluate the performance of balanced binary classifiers with a decision threshold is through the receiver operating characteristic (ROC), which expresses, for different values of the threshold, the fraction of true positives (TPs), i.e., AEs detected as AEs, against the fraction of false positives (FPs), i.e., non-AEs detected as AEs.

The baseline performance, i.e., the performance of a random classifier that detects as many TPs as FPs for any value of the threshold, is represented in an ROC plot by the linear pattern with slope 1 [passing through $(0, 0)$ and $(1, 1)$], while the best performance is reached when the classifier achieves TP rate 1 and FP rate 0. A more compact way to represent the performance is through the area under curve (AUC) of an ROC graph: perfect classification performance has AUC of 1, while baseline performance has AUC equal to 0.5.

When using this performance indicator, the information about exact TP/FP ratio is lost. However, this is not important when dealing with detection systems with AUC close to 1,

[2]The additive Gaussian noise is with 0 mean and standard deviation set equal to the adversarial strength $\epsilon = 0.05$ of the AEs, to obtain a fair comparison between attacker and defender capabilities.

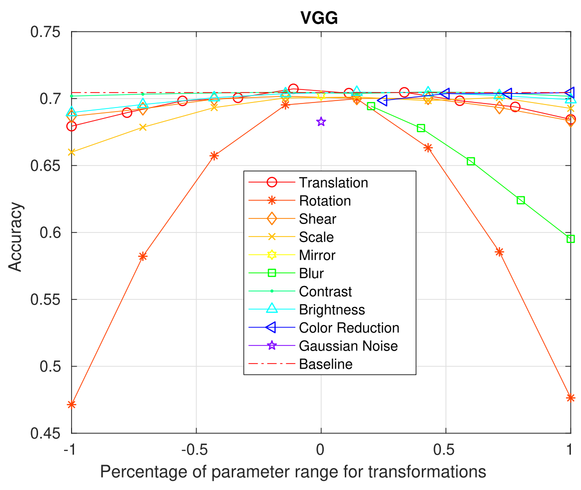


Fig. 7. Accuracy of VGG-19 when using input transformations.

since they will produce very similar-looking ROC graphs, all passing close to the point (1, 0). Conversely, when the classification performance is poor, the ROC curve has much more "freedom" and could assume several different shapes with the same AUC result. Since the focus of this work is to achieve a high-performance AE detection, there is no interest in studying the actual ROC curve of poorly performing detection systems. Solutions with AUC $\geq$0.95 can be considered satisfying for our purposes.

The results are presented for multiple transformations. In all the following plots, the performance value (accuracy or AUC of ROC) is on the y-axis, while the x-axis refers to the normalized value of the parameter used to control the transformations. Note that some transformations are controlled by a parameter that varies in a symmetric range (e.g., see the case of translation), while others are not. The ranges of the former are of the form $[\gamma - \alpha, \gamma + \alpha]$ and each parameter $\theta \in [\gamma - \alpha, \gamma + \alpha]$ is reported on the x-axis as $(\theta - \gamma)/\alpha$, i.e., obtaining a normalized representation that varies from -1 to $+1$. Conversely, nonsymmetric ranges are of the form $[\alpha, \beta]$, with $\alpha < \beta$, and each parameter $\theta \in [\alpha, \beta]$ is reported on the x-axis as θ/β, hence obtaining a normalized value that varies from α/β to 1 only. Mirror and Gaussian noise are parameter-free transformations and their performance is represented by a single point with value 0 on the x-axis.

D. Accuracy Drop

A major drawback of using input image transformations for AEs' detection is that they cause an accuracy drop for the classifier. The baseline accuracy is evaluated without input transformation on a validation subset of ImageNet composed of ten images for each class, for a total of 10k images. For each transformation, the accuracy is computed for discretized values within the ranges of each parameter and reported in Fig. 7 for VGG-19. The results obtained with the other networks show a very similar pattern for all the transformations, hence they are not reported for space limitations.

E. AEs Datasets

Several datasets of AEs have been produced to test the performance of the proposed approaches and generate the defense perturbations. Each dataset consists of multinetwork AEs. For standard AEs, two datasets have been generated, one with the L-BFGS attack and one for the CW attack, each containing 1000 samples, one for each class of the ImageNet dataset. For robust AEs, four different datasets have been generated, each serving a different purpose.

1) *M_{test}:* It is generated to test the performance of the ENHANCED and VOTING_ENHANCED architectures. It includes adversarial samples generated starting from nine samples of the first 100 classes and two for the other 900 classes, for a total of 2700 AEs. This dataset is much larger than the others because it is crucial to evaluate whether the proposed counter-measure is effective in general, meaning that it has to generalize both for a wide distribution of AEs belonging to the same class and for each class.
2) *$M_{def\text{-}gen}$:* It is the one used to generate the defense perturbation, referred to as D_{def} in the following, used in the ENHANCED and VOTING_ENHANCED architectures, as detailed in Section IV-B.
3) *$M_{att\text{-}gen}$:* This dataset is used to simulate the case in which an attacker tries to attack the ENHANCED and VOTING_ENHANCED architectures. In this case, since the attacker is assumed not to know D_{def}, he/she has to first generate its own defense perturbation (referred to as D_{att}) before attempting at generating AEs that are robust to enhanced detection architectures. The $M_{\text{att-gen}}$ dataset serves the purpose of generating D_{att} with the approach detailed in Section IV-B.
4) *M_{att}:* It is the dataset generated to be robust to both input transformations and defense perturbation D_{att}. It is used to experimentally confirm that it is not possible to generate AEs that are robust to the ENHANCED and VOTING_ENHANCED architectures, provided that the attacker does not know the exact AEs used to generate the defense perturbation D_{def}.

All the multinetwork robust AEs are generated by considering: 1) one transformation for each class discussed in Section V-B, namely, translation, blur, and Gaussian noise and 2) the cross-entropy loss function (as for the L-BFGS attack).

These three transformations proved to be sufficient to make the AE robust to the entire set of considered transformations, hence making possible to generate robust AEs with a limited computational effort.

All the generated datasets are summarized in Table I. The actual size of the dataset is twice the one showed in the table, since it also includes the original images.

Except for the L-BFGS dataset, which was optimized over 250 epochs, all the other datasets were optimized over 500 epochs. For every dataset the Adam optimizer was used, with a learning rate of 0.1, and a fixed adversarial strength $\epsilon = 0.05$ was chosen.

TABLE I
AEs' Datasets Generated. DP1 Refers to the Defense Perturbation Used for Detection, Whereas DP2 Refers to the Defense Perturbation Generated to Craft Defense-Robust AEs

Dataset	**Robust to...**	**Samples**	Notes
Standard (L-BFGS)	-	1000	1 per class
Standard CW (CW)	-	1000	1 per class
Multi-robust (M_{test})	Translation Blur Gaussian Noise	2700	9 for the first 100 classes 2 for the others Used for tests
Multi-robust ($M_{\text{def-gen}}$)	Translation Blur Gaussian Noise	1000	1 per class Used for D_{def} generation
Multi-robust ($M_{\text{att-gen}}$)	Translation Blur Gaussian Noise	1000	1 per class Used for D_{att} generation (to craft dataset M_{att})
Defense-robust (M_{att})	Translation Blur Gaussian Noise D_{att}	1000	1 per class

F. Detection of AEs With Only Input Transformations

This section discusses the performance of the VOTING_BASELINE architecture (Fig. 3). When dealing with standard AEs, this architecture exhibits a very good performance, as it can be seen from Fig. 8 (L-BFGS attack) and Fig. 9 (CW attack). The AEs generated with the CW attack resulted to be harder to detect and, for some transformation, the performance degrades quicker for higher parameter values. Clearly, some of the transformations are less suitable than others for this kind of detection. Contrast and brightness change, color reduction, and rotation show poorer performance with respect to other transformations (e.g., other topological transformations, blur, and Gaussian noise).

Very different results are obtained when presenting robust AEs to the VOTING_BASELINE architecture. These results are summarized in Fig. 10. As expected, the detection performance is much worse. What is interesting is that even if the AEs were chosen to be robust to three transformations, the performance of all the other transformations drops, especially when dealing with small parametric values. This can be explained with the fact that transformations with small parameter values are similar to each other, especially when dealing with transformations of the same category (e.g., scale and shear have comparable detection performance with respect to translation).

Some transformations resulted to be even worse than a random detection algorithm, meaning that even a coin toss shows better AE detection performance. The experiment shows that when dealing with robust AEs, differentiable input transformations cannot be used in this naive detection system. Not even noise (Gaussian in this case) is a safe choice: even though it showed better performance with respect to other transformations, its AUC drops to about 0.82, with a ROC curve that saturates (i.e., 100% of TP) for about 50% of FP.

Further tests were conducted to explore the effect of cascaded transformations: the usage of a series of randomly selected transformations can help when dealing with this kind of robust AEs, but it is also still possible to craft

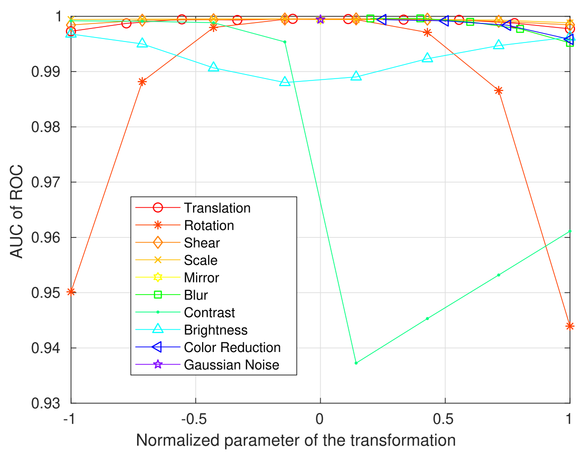


Fig. 8. Performance of the VOTING_BASELINE architecture in detecting standard AEs (L-BFGS) for each input transformation.

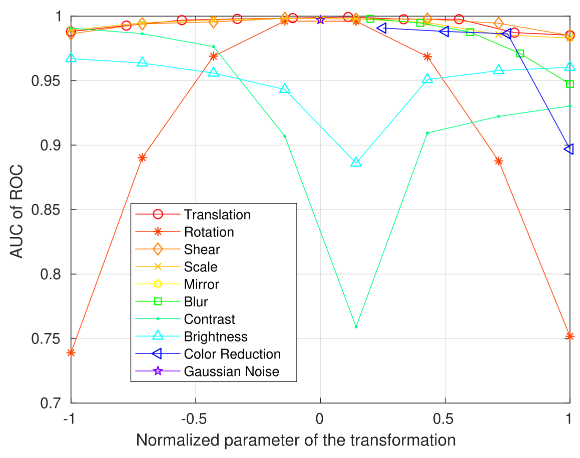


Fig. 9. Performance of the VOTING_BASELINE architecture in detecting standard AEs (CW) for each input transformation.

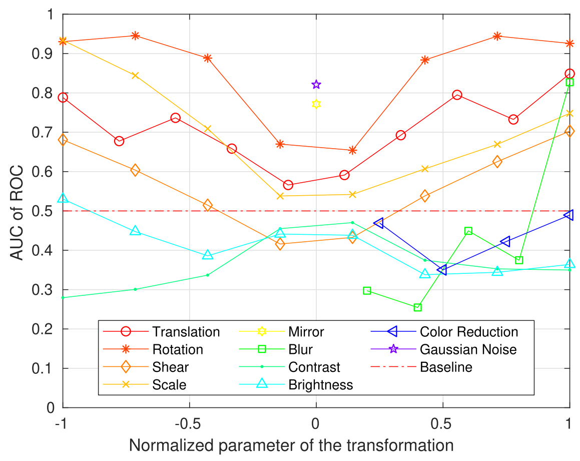


Fig. 10. Performance of the VOTING_BASELINE architecture in detecting robust AEs (translation, blur, Gaussian noise) for each input transformation.

AEs that are robust to any possible combination of cascaded transformations. In our experiment, this has been verified via cascades of four randomly selected transformations (picked from the three transformations used before, i.e., translation, blur, and Gaussian noise). These results were omitted for space limitations.

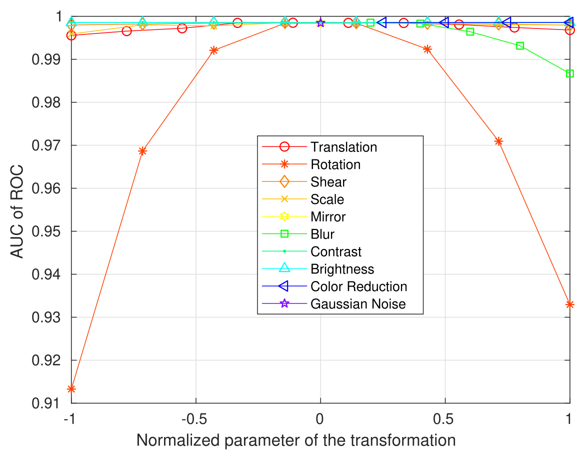


Fig. 11. Performance of the VOTING_ENHANCED architecture in detecting standard AEs (L-BFGS) for each input transformation.

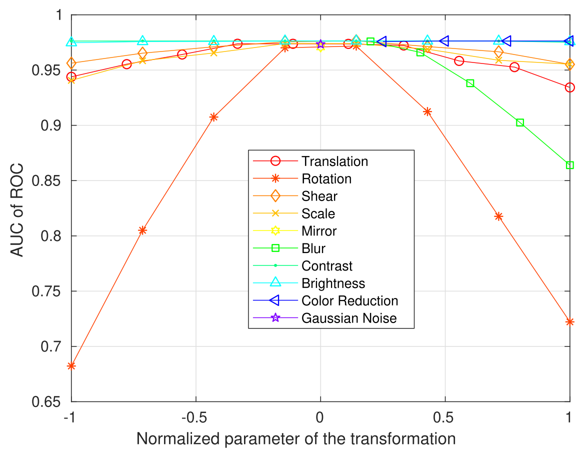


Fig. 12. Performance of the VOTING_ENHANCED architecture in detecting standard AEs (CW) for each input transformation.

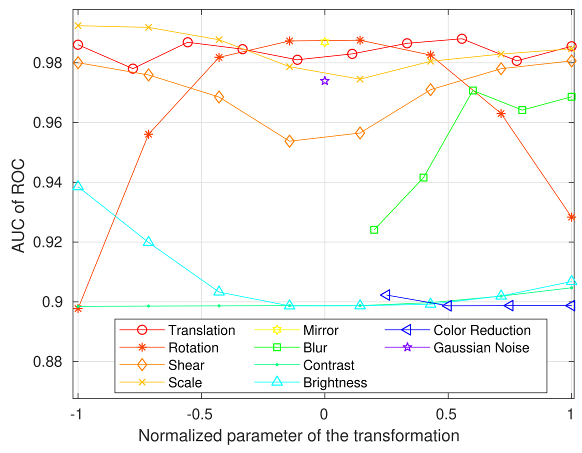


Fig. 13. Performance of the VOTING_ENHANCED architecture in detecting robust AEs (set M_{test}) for each input transformation.

G. Detection of AEs Using the Defense Perturbation

The results of Section V-F showed that an attacker unaware of the specific input transformation used for detection can

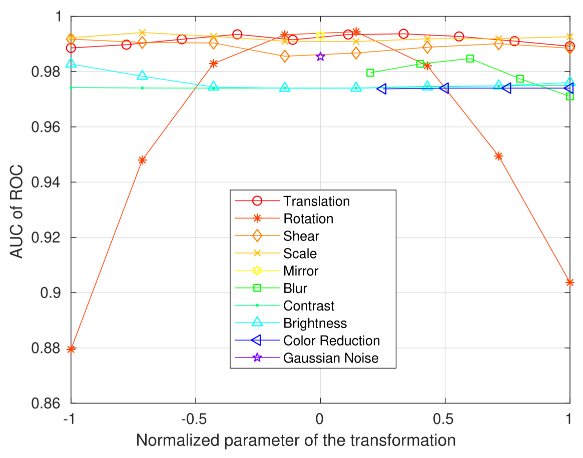


Fig. 14. Performance of the VOTING_ENHANCED architecture in detecting robust AEs (set M_{att}) for each input transformation.

still craft robust AEs that jeopardize the detection system. This section reports the results achieved by applying the defense perturbation with the VOTING_ENHANCED architecture illustrated in Fig. 6.

As explained in Section IV, the defense perturbation is effective not only for detecting standard AEs but also for making robust AEs sensitive to input transformations again. The AEs used to craft the defense perturbation are generated from images different than those used to test the performance of the detection system. The same three transformations were chosen (translation, blur, and Gaussian noise).

The performance of the VOTING_ENHANCED architecture in detecting standard AEs with a defense perturbation is shown in Fig. 11 for the L-BFGS attack, and in Fig. 12 for the CW attack, while the results for robust AEs are shown in Fig. 13. Interestingly, as observed for the results shown in Fig. 10, although only three transformations were used to craft the defense perturbation, all the other transformations exhibited a performance boost from the use of the defense perturbation.

Another set of experiments has been performed to assess the robustness of the defense perturbation. As mentioned in Section IV-B, the attacker does not have access to the defense perturbation D_{def} nor to the specific dataset $M_{\text{def-gen}}$ used to generate it. Anyway, he/she can still try to generate another defense perturbation D_{att} from a different (but presumably similar) dataset $M_{\text{att-gen}}$ with the purpose of generating a set M_{att} of AEs that are robust to D_{def}.

The results of the detection of M_{att} are reported in Fig. 14, which shows that the AEs prepared by the attacker can still be detected with D_{def} in the VOTING_ENHANCED architecture. This makes the VOTING_ENHANCED architecture an effective counter-measure for robust AEs, making them sensitive to input perturbations, while maintaining good detection performance on standard AEs.

H. Effect of Voting

This section evaluates the effect of voting used in the VOTING_ENHANCED architecture and shows how, in most cases, it increases the detection accuracy with respect to the ENHANCED architecture, which considers a single network.

To present the results in a compact way, the effect of voting is reported as the minimum improvement of the AUC computed over the parametric range, for each transformation and for each network. The change is reported as a percentage of the total area (i.e., 1) for an easier reading. Positive values of the AUC change are to be considered as improvements of the VOTING_ENHANCED architecture with respect to the ENHANCED architecture (i.e., with a single network). Conversely, negative values represent worse AUC. The minimum change in performance over the parametric range is chosen because it indicates the worst case improvement due to voting.

The results of this test are summarized in Table II. It is worth observing that voting helps in most of the cases, except for rotation, where the performance of the resulting voting system is heavily affected by the difference in performance between the networks.

I. Comparison With State-of-the-Art Defenses

To the best of our records, the detection method presented in this article is the first attempt to defend CNNs against attacks from robust AEs in a white-box setting with input transformations. Previous work that used input transformations to detect white-box AEs did not consider robust AEs. This is the case of two works: Prakash *et al.* [21] and Xie *et al.* [22], which used randomization and pixel deflection, respectively. Randomization[3] consists of concatenating two input transformations, namely, rescaling and padding; pixel deflection is an image transformation that swaps nearby pixel values. We compared two versions: the white-box version (naive) and the full defense with Wavelet Denoiser.[4]

Furthermore, our comparison also considered VisionGuard [11], which uses an architecture similar to BASELINE, with JPEG compression as input transformation (which is not differentiable).

For a fair comparison, the evaluation was carried out on a single network (Inception-v4), as the existing methods did not consider multinetwork architectures. The transformation used in our method is Gaussian noise, as it proved to be the best-performing among the evaluated ones. The evaluation metric is a tuple representing the TP and FP rates. This choice is motivated by the fact that pixel deflection and randomization are not merely detection methods but defenses (i.e., they modify the input image to correctly classify it). The TP and FP rates for the two detection methods (ours and VisionGuard), are computed by just selecting the threshold that provides the best rate (instead of computing the AUC of the ROC graph).

The results are summarized in Table III. As it can be noted from the table, our method exhibits comparable performance with respect to the other methods for the detection of standard AEs (L-BFGS and CW columns). On the contrary, none of the other methods is capable of properly detecting robust AEs

[3]2017 NIPS adversarial defense competition runner-up—code available at https://github.com/anishathalye/obfuscated-gradients/tree/master/randomization

[4]The code for this part was taken from https://github.com/anishathalye/pixel-deflection

TABLE II

MINIMUM CHANGE IN AUC OF ROC (PERCENTUAL OF TOTAL AUC = 1) INTRODUCED BY THE VOTING_ENHANCED ARCHITECTURE WITH RESPECT THE ENHANCED ARCHITECTURE. RESULTS ARE SHOWN FOR EACH TRANSFORMATION AND FOR EACH DATASET OF AEs; THE ROWS IN EACH CELL REPRESENT THE MINIMUM CHANGE IN AUC WITH RESPECT TO EACH NETWORK (VGG-19, RESNET-V2-152, AND INCEPTION-V4, RESPECTIVELY)

	L-BFGS	CW	M_{test}	M_{att}
Translation	+0.71	+4.90	+2.36	+2.05
	+0.96	+2.74	+2.13	+2.37
	+0.18	+0.50	+0.47	-0.01
Rotation	+0.78	+4.94	+3.41	+2.46
	+1.07	-2.15	+2.23	+2.52
	-3.53	-6.28	-2.92	-4.88
Shear	+0.80	+5.13	+3.65	+2.94
	+0.83	+2.59	+2.71	+2.37
	+0.20	+0.78	+1.05	-0.04
Scale	+0.76	+4.95	+2.27	+2.37
	+0.95	+2.90	+2.08	+2.25
	+0.22	+0.41	+0.48	+0.12
Mirror	+0.75	+4.81	+2.28	+2.10
	+1.07	+3.23	+1.74	+1.85
	+0.19	+0.82	+2.03	+0.38
Blur	+0.85	+5.74	+3.23	+4.43
	+0.88	+2.56	+5.03	+2.97
	+0.26	+1.54	+0.18	-0.20
Contrast	+0.85	+5.78	+0.24	+3.98
	+0.78	+2.23	+5.88	+3.10
	+0.19	+1.30	+1.59	-0.50
Brightness	+0.85	+5.73	+0.25	+2.81
	+0.78	+2.23	+5.02	+3.10
	+0.19	+1.28	+1.76	-0.41
Color Reduction	+0.84	+5.72	+0.26	+3.97
	+0.78	+2.23	+5.78	+3.10
	+0.20	+1.36	+1.63	-0.41
Gaussian Noise	+0.84	+5.68	+4.79	+4.14
	+0.84	+2.37	+3.48	+2.8
	+0.22	+1.74	-0.47	-0.49

(M_{test} column), while our method provides a high detection performance. Note that the AEs in M_{test} have not been generated to be robust to pixel deflection (with Wavelet denoiser) and VisionGuard (i.e., they have been generated to be robust to translation, blur, and Gaussian noise), which are even based on nondifferentiable functions. This suggests that there exists a sort of transferability effect for robust AEs: future work will investigate in this direction.

J. Discussion and Future Work

In the light of the above experimental results, some aspects are worth to be discussed.

1) Best Transformation: Among the transformations that were tested, some performed better than others. For example, rotation shows poor performance for all kinds of adversarial datasets. Others, such as contrast and brightness changes, and

TABLE III

COMPARISON WITH STATE-OF-THE-ART METHODS USING INPUT TRANSFORMATIONS AGAINST THE SAME THREAT MODEL. THE TWO VALUES IN EACH CELL DENOTE THE TP AND FP RATES, RESPECTIVELY

Method	L-BFGS	CW	M_{test}
Randomization [22]	0.983 0.085	0.986 0.088	0.010 0.070
Pixel Deflection [21] (naive)	0.948 0.083	0.982 0.082	0.004 0.071
Pixel Deflection [21] (w/ Wavelet denoiser)	0.985 0.125	0.986 0.123	0.048 0.111
VisionGuard [11] (JPEG92)	**0.997** 0.060	**0.992** 0.078	0.043 0.149
Ours (Gaussian noise)	0.978 0.088	0.954 0.119	**0.944** 0.080

bit depth color reduction, show poor performance in detecting robust AEs, even though the AEs were not specifically robust to those transformations. Blur is not among the best-performing either, while scale, translation, shear, mirror, and Gaussian noise consistently show good performance.

In general, by the results of Section V-I, it emerges that to use our method with the ENHANCED architecture, one has to accept slightly lower performance in detecting standard AEs to be effective against all types of AEs. This happens because it has been experimentally found that the defense perturbation makes standard AEs robust to the input transformations considered in this work.

2) Results Cannot be Generalized to Any CNN: It is not possible to state that translation will perform well as an AEs' detector for any CNN, trained on any dataset. Performance will likely depend not only on the architecture and the dataset used but also on the data augmentation techniques used during training. Further experimental evaluations should be performed in this direction.

Nevertheless, the analysis presented in this article showed that networks with different architecture, but trained on the same dataset,[5] present similar results for this kind of AEs' detection system: the best and the worst performing transformations are the same, and in general the patterns of the detection performance of the transformations as functions of their parameters are similar.

3) Robust AEs Are Robust to Nondifferentiable Transformations Too: The reason why robust AEs are not detected with randomization or naive pixel deflection is evident: all those kinds of differentiable transformations are fooled by robust AEs because of the similarity of the transformations with respect to the ones used to craft them. However, the reasons why VisionGuard and pixel deflection with wavelet denoiser are not able to detect robust AEs is still unclear and it is surely worth investigating. The input transformations used by these works are JPEG compression and wavelet denoiser, which are nondifferentiable input transformations and, therefore, cannot be used to generate robust AEs, at least as defined in this work. Our hypothesis (to be validated by future experiments) is that there exists some kind of transferability of AEs not only

[5]ImageNet was chosen to provide real-world images; synthetic datasets as MNIST will surely show different results.

between different architectures of neural networks but also in robustness between transformations.

4) Open Issues: The following aspects require further investigation to be generalized.

1) Due to space limits, not all the most common kinds of AEs were considered in the evaluation. Although the results of this experimental evaluation cannot be generalized to the entire spectrum of AEs, this work can be considered as a starting point to exhaustively test the detection potentiality of input transformations. The loss function used to optimize the robust AEs is the cross-entropy loss. Other losses would certainly lead to different results. However, the process for generating the defense perturbation is general enough to account for robust AEs crafted with different loss functions, which will be explored in a future work.
2) This article considered a detection system using threshold-based binary classification. By varying the threshold, it is possible to plot ROC graphs that help understand which transformations are best suited for this kind of detection. However, at run time a certain threshold must be chosen (one for each network). The thresholds should take into account all the possible kinds of AEs that one wants to detect, to average the performance over the entire possible range of different AEs.
3) The reason why nondifferentiable input transformations are not able to detect robust AEs is still not clear and should be investigated in detail.

VI. Conclusion

This article introduced a methodology to detect AEs for CNNs. The method exploits the detection power of input image transformations for standard AEs, which have been extensively tested to discover those transformations that are more suitable for this kind of detection problem.

Although robust AEs can significantly degrade the performance of simple detection architectures (BASELINE and VOTING_BASELINE), this article presented a counter-measure against robust AEs based on the generation of a defense perturbation. This perturbation allows making robust AEs sensitive again to input transformations and it can be used to achieve very good detection performance for both standard and robust AEs. Majority voting for multi-CNN systems has also been introduced to further improve the detection performance.

Future work will investigate extensions of the approaches presented in this article to understand whether different kinds of attacks can fool the proposed detection systems. Also, further tests will be conducted to clarify the role of data distribution used to generate the defense perturbation, to better comprehend whether it is possible for an attacker to fool the proposed detection systems without knowing the exact data distribution used to craft the defense.

References

[1] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, "A survey of the recent architectures of deep convolutional neural networks," *Artif. Intell. Rev.*, vol. 53, no. 8, pp. 5455–5516, Apr. 2020, doi: 10.1007/s10462-020-09825-6.

[2] D. Mishkin, N. Sergievskiy, and J. Matas, "Systematic evaluation of convolution neural network advances on the imagenet," *Comput. Vis. Image Understand.*, vol. 161, pp. 11–19, Aug. 2017.

[3] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Feb. 2015, doi: 10.1038/nature14236.

[4] C. Szegedy *et al.*, "Intriguing properties of neural networks," 2013, *arXiv:1312.6199*. [Online]. Available: http://arxiv.org/abs/1312.6199

[5] J. Zhang and C. Li, "Adversarial examples: Opportunities and challenges," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 7, pp. 2578–2593, Jul. 2020.

[6] M. Bojarski *et al.*, "End to end learning for self-driving cars," 2016, *arXiv:1604.07316*. [Online]. Available: http://arxiv.org/abs/1604.07316

[7] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," Jul. 2016, *arXiv:1607.02533*. [Online]. Available: http://arxiv.org/abs/1607.02533

[8] B. Nassi, Y. Mirsky, D. Nassi, R. Ben-Netanel, O. Drokin, and Y. Elovici, "Phantom of the ADAS: Securing advanced driver-assistance systems from split-second phantom attacks," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*. New York, NY, USA: Association for Computing Machinery, 2020, pp. 293–308, doi: 10.1145/3372297.3423359.

[9] X. Huang *et al.*, "A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability," 2018, *arXiv:1812.08342*. [Online]. Available: http://arxiv.org/abs/1812.08342

[10] C. Guo, M. Rana, M. Cisse, and L. van der Maaten, "Countering adversarial images using input transformations," Oct. 2017, *arXiv:1711.00117*. [Online]. Available: http://arxiv.org/abs/1711.00117

[11] Y. Kantaros *et al.*, "VisionGuard: Runtime detection of adversarial inputs to perception systems," Feb. 2020, *arXiv:2002.09792*. [Online]. Available: http://arxiv.org/abs/2002.09792

[12] A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok, "Synthesizing robust adversarial examples," Jul. 2017, *arXiv:1707.07397*. [Online]. Available: http://arxiv.org/abs/1707.07397

[13] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," Oct. 2016, *arXiv:1610.08401*. [Online]. Available: http://arxiv.org/abs/1610.08401

[14] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.

[15] A. Biondi, F. Nesti, G. Cicero, D. Casini, and G. Buttazzo, "A safe, secure, and predictable software architecture for deep learning in safety-critical systems," *IEEE Embedded Syst. Lett.*, vol. 12, no. 3, pp. 78–82, Sep. 2020.

[16] C. Xie *et al.*, "Improving transferability of adversarial examples with input diversity," Mar. 2018, *arXiv:1803.06978*. [Online]. Available: http://arxiv.org/abs/1803.06978

[17] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," Aug. 2016, *arXiv:1608.04644*. [Online]. Available: http://arxiv.org/abs/1608.04644

[18] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," Dec. 2014, *arXiv:1412.6572*. [Online]. Available: http://arxiv.org/abs/1412.6572

[19] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "DeepFool: A simple and accurate method to fool deep neural networks," Nov. 2015, *arXiv:1511.04599*. [Online]. Available: http://arxiv.org/abs/1511.04599

[20] R. Wiyatno and A. Xu, "Maximal jacobian-based saliency map attack," Aug. 2018, *arXiv:1808.07945*. [Online]. Available: http://arxiv.org/abs/1808.07945

[21] A. Prakash, N. Moran, S. Garber, A. DiLillo, and J. Storer, "Deflecting adversarial attacks with pixel deflection," Jan. 2018, *arXiv:1801.08926*. [Online]. Available: http://arxiv.org/abs/1801.08926

[22] C. Xie, J. Wang, Z. Zhang, Z. Ren, and A. Yuille, "Mitigating adversarial effects through randomization," Nov. 2017, *arXiv:1711.01991*. [Online]. Available: http://arxiv.org/abs/1711.01991

[23] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," Nov. 2015, *arXiv:1511.04508*. [Online]. Available: http://arxiv.org/abs/1511.04508

[24] S. Gu and L. Rigazio, "Towards deep neural network architectures robust to adversarial examples," Dec. 2014, *arXiv:1412.5068*. [Online]. Available: http://arxiv.org/abs/1412.5068

[25] P. Samangouei, M. Kabkab, and R. Chellappa, "Defense-GAN: Protecting classifiers against adversarial attacks using generative models," May 2018, *arXiv:1805.06605*. [Online]. Available: http://arxiv.org/abs/1805.06605

[26] D. Meng and H. Chen, "MagNet: A two-pronged defense against adversarial examples," May 2017, *arXiv:1705.09064*. [Online]. Available: http://arxiv.org/abs/1705.09064

[27] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: Attacks and defenses for deep learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 9, pp. 2805–2824, Sep. 2019.

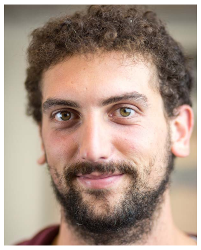
[28] H. Xu *et al.*, "Adversarial attacks and defenses in images, graphs and text: A review," 2019, *arXiv:1909.08072*. [Online]. Available: http://arxiv.org/abs/1909.08072

[29] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," Sep. 2014, *arXiv:1409.1556*. [Online]. Available: http://arxiv.org/abs/1409.1556

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," Mar. 2016, *arXiv:1603.05027*. [Online]. Available: http://arxiv.org/abs/1603.05027

[31] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, inception-ResNet and the impact of residual connections on learning," Feb. 2016, *arXiv:1602.07261*. [Online]. Available: http://arxiv.org/abs/1602.07261

[32] S. Guadarrama and N. Silberman. (2016). *TensorFlow-Slim: A Lightweight Library for Defining, Training and Evaluating Complex Models in TensorFlow*. [Online]. Available: https://github.com/google-research/tf-slim and https://github.com/google-research/tf-slim



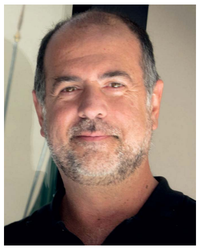
Federico Nesti received the M.Sc. degree *(cum laude)* in robotics and automation engineering from the University of Pisa, Pisa, Italy, in 2018. He is currently pursuing the Ph.D. degree with the Real-Time Systems (ReTiS) Laboratory, Scuola Superiore Sant'Anna, Pisa. He wrote a thesis as a visiting student at TU Delft, Delft, The Netherlands.

He coauthored papers with the members of the U-PHOS project, a university space hands-on opportunity for which he designed electronics and controls.

Alessandro Biondi (Member, IEEE) graduated *(cum laude)* in computer engineering from the University of Pisa, Pisa, Italy, in 2013, within the excellence program, and received the Ph.D. degree in computer engineering from the Scuola Superiore Sant'Anna, Pisa, in 2017, under the supervision of Prof. Giorgio Buttazzo and Prof. Marco Di Natale.

He is an Assistant Professor with the Real-Time Systems (ReTiS) Laboratory, Scuola Superiore Sant'Anna. In 2016, he was a Visiting Scholar at the Max Planck Institute for Software Systems, Saarbrücken, Germany. His research interests include design and implementation of real-time operating systems and hypervisors, schedulability analysis, cyber-physical systems, synchronization protocols, and safe and secure machine learning.

Dr. Biondi was a recipient of the six Best Paper Awards, one Outstanding Paper Award, the ACM SIGBED Early Career Award 2019, and the EDAA Dissertation Award 2017.



Giorgio Buttazzo (Fellow, IEEE) graduated in electronic engineering from the University of Pisa, Pisa, Italy, in 1985, and received the M.S. degree in computer science from the University of Pennsylvania, Philadelphia, PA, USA, in 1987, and the Ph.D. degree in computer engineering from the Scuola Superiore Sant'Anna, Pisa, in 1991.

He is a Full Professor of computer engineering at the Scuola Superiore Sant'Anna, Pisa, Italy. He has authored seven books in real-time systems and more than 300 articles in the field of real-time systems, robotics, and neural networks.

Dr. Buttazzo received 11 best paper awards. He is the Editor-in-Chief of *Real-Time Systems* and an Associate Editor of the *ACM Transactions on Cyber-Physical Systems*.