Defense against Universal Adversarial Perturbations

Naveed Akhtar* Jian Liu* Ajmal Mian
*The authors contributed equally to this work.
Computer Science and Software Engineering
The University of Western Australia
naveed.akhtar@uwa.edu.au, jian.liu@research.uwa.edu.au, ajmal.mian@uwa.edu.au

Abstract

Recent advances in Deep Learning show the existence of image-agnostic quasi-imperceptible perturbations that when applied to 'any' image can fool a state-of-the-art network classifier to change its prediction about the image label. These 'Universal Adversarial Perturbations' pose a serious threat to the success of Deep Learning in practice. We present the first dedicated framework to effectively defend the networks against such perturbations. Our approach learns a Perturbation Rectifying Network (PRN) as 'pre-input' layers to a targeted model, such that the targeted model needs no modification. The PRN is learned from real and synthetic image-agnostic perturbations, where an efficient method to compute the latter is also proposed. A perturbation detector is separately trained on the Discrete Cosine Transform of the input-output difference of the PRN. A query image is first passed through the PRN and verified by the detector. If a perturbation is detected, the output of the PRN is used for label prediction instead of the actual image. A rigorous evaluation shows that our framework can defend the network classifiers against unseen adversarial perturbations in the real-world scenarios with up to 97.5% *success rate. The PRN also generalizes well in the sense that training for one targeted network defends another network with a comparable success rate.*

1. Introduction

Deep Neural Networks are at the heart of the current advancements in Computer Vision and Pattern Recognition, providing state-of-the-art performance on many challenging classification tasks [9], [12], [14], [16], [36], [37]. However, Moosavi-Dezfooli et al. [25] recently showed the possibility of fooling the deep networks to change their prediction about 'any' image that is slightly perturbed with the *Universal Adversarial Perturbations*. For a given network model, these image-agnostic (hence *universal*) perturbations can be computed rather easily [25], [26]. The perturbations remain

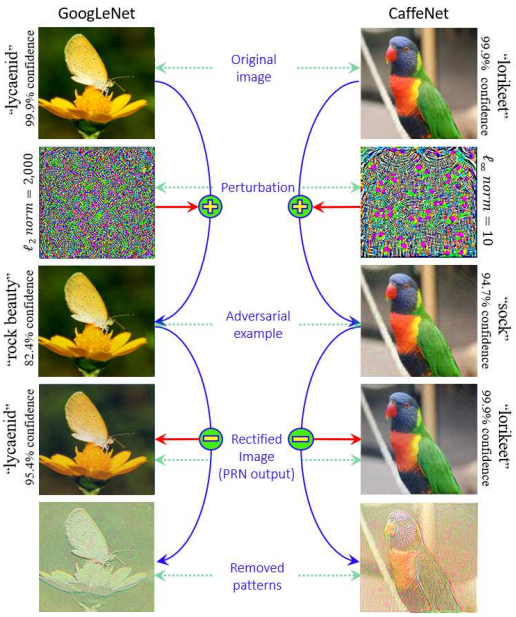

Figure 1. Adding quasi-imperceptible universal adversarial perturbations [25] can fool neural networks. The proposed framework rectifies the images to restore the network predictions. The patterns removed by rectification are separately analyzed to decide on the presence of adversarial perturbations in images. The shown 'perturbations' and 'removed patterns' are normalized on different scales for better visualization.

quasi-imperceptible (see Fig. 1), yet the *adversarial examples* generated by adding the perturbations to the images fool the networks with alarmingly high probabilities [25]. Furthermore, the fooling is able to generalize well across different network models.

Being image-agnostic, universal adversarial perturbations can be conveniently exploited to fool models on-the-fly on unseen images by using pre-computed perturbations. This even eradicates the need of on-board computational capacity that is needed for generating image-specific perturbations [7], [21]. This fact, along the cross-model generalization of universal perturbations make them particularly relevant to the practical cases where a model is deployed in

a possibly hostile environment. Thus, defense against these perturbations is a necessity for the success of Deep Learning in practice. The need for counter-measures against these perturbations becomes even more pronounced considering that the real-world scenes (e.g. sign boards on roads) modified by the adversarial perturbations can also behave as adversarial examples for the networks [17].

This work proposes the first dedicated defense against the universal adversarial perturbations [25]. The major contributions of this paper are as follows:

- We propose to learn a Perturbation Rectifying Network (PRN) that is trained as the 'pre-input' of a targeted network model. This allows our framework to provide defense to already deployed networks without the need of modifying them.
- We propose a method to efficiently compute synthetic image-agnostic adversarial perturbations to effectively train the PRN. The successful generation of these perturbations complements the theoretical findings of Moosavi-Dezfooli [26].
- We also propose a separate perturbation detector that is learned from the Discrete Cosine Transform of the image rectifications performed by the PRN for clean and perturbed examples.
- Rigorous evaluation is performed by defending the GoogLeNet [37], CaffeNet [16] and VGG-F network [4][1], demonstrating up to 97.5% success rate on unseen images possibly modified with unseen perturbations. Our experiments also show that the proposed PRN generalizes well across different network models.

2. Related work

The robustness of image classifiers against adversarial perturbations has gained significant attention in the last few years [6], [7], [29], [32], [34], [35], [40]. Deep neural networks became the center of attention in this area after Szegedy et al. [39] first demonstrated the existence of adversarial perturbations for such networks. See [1] for a recent review of literature in this direction. Szegedy et al. [39] computed adversarial examples for the networks by adding quasi-imperceptible perturbations to the images, where the perturbations were estimated by maximizing the network's prediction error. Although these perturbations were image-specific, it was shown that the same perturbed images were able to fool multiple network models. Szegedy et al. reported encouraging results for improving the model robustness against the adversarial attacks by using adversarial examples for training, a.k.a. *adversarial training*.

[1]The choice of the networks is based on the computational feasibility of generating the adversarial perturbations for the evaluation protocol in Section 5. However, our approach is generic in nature.

Goodfellow et al. [10] built on the findings in [39] and developed a 'fast gradient sign method' to efficiently generate adversarial examples that can be used for training the networks. They hypothesized that it is the linearity of the deep networks that makes them vulnerable to the adversarial perturbations. However, Tanay and Griffin [41] later constructed the image classes that do not suffer from the adversarial examples for the linear classifiers. Their arguments about the existence of the adversarial perturbations again point towards the over-fitting phenomena, that can be alleviated by regularization. Nevertheless, it remains unclear how a network should be regularized for robustness against adversarial examples.

Moosavi-Dezfooli [27] proposed the DeepFool algorithm to compute image-specific adversarial perturbations by assuming that the loss function of the network is linearizable around the current training sample. In contrast to the one-step perturbation estimation [10], their approach computes the perturbation in an iterative manner. They also reported that augmenting training data with adversarial examples significantly increases the robustness of networks against the adversarial perturbations. Baluja and Fischer [2] trained an Adversarial Transformation Network to generate adversarial examples against a target network. Liu et al. [19] analyzed the transferability of adversarial examples. They studied this property for both targeted and non-targeted examples, and proposed an ensemble based approach to generate the examples with better transferability.

The above-mentioned techniques mainly focus on generating adversarial examples, and address the defense against those examples with adversarial training. In-line with our take on the problem, few recent techniques also directly focus on the defense against the adversarial examples. For instance, Lu et al. [22] mitigate the issues resulting from the adversarial perturbations using foveation. Their main argument is that the neural networks (for ImageNet [33]) are robust to the foveation-induced scale and translation variations of the images, however, this property does not generalize to the perturbation transformations.

Papernot et al. [30] used distillation [13] to make the neural networks more robust against the adversarial perturbations. However, Carlini and Wagner [3] later introduced adversarial attacks that can not be defended by the distillation method. Kurakin et al. [18] specifically studied the adversarial training for making large models (e.g. Inception v3 [38]) robust to perturbations, and found that the training indeed provides robustness against the perturbations generated by the one-step methods [10]. However, Tramer et al. [42] found that this robustness weakens for the adversarial examples learned using different networks i.e. for the black-box attacks [19]. Hence, ensemble adversarial training was proposed in [42] that uses adversarial examples generated by multiple networks.

Dziugaite et al. [5] studied the effects of JPG compression on adversarial examples and found that the compression can sometimes revert network fooling. Nevertheless, it was concluded that JPG compression alone is insufficient as a defense against adversarial attacks. Prakash et al. [31] took advantage of localization of the perturbed pixels in their defense. Lu et al. [20] proposed SafetyNet for detecting and rejecting adversarial examples for the conventional network classifiers (e.g. VGG19 [11]) that capitalizes on the late stage ReLUs of the network to detect the perturbed examples. Similarly, a proposal of appending the deep neural networks with detector subnetworks was also presented by Metzen et al. [23]. In addition to the classification, adversarial examples and robustness of the deep networks against them have also been recently investigated for the tasks of semantic segmentation and object detection [8], [21], [43].

Whereas the central topic of all the above-mentioned literature is the perturbations computed for *individual* images, Moosavi-Dezfooli [25] were the first to show the existence of image-agnostic perturbations for neural networks. These perturbations were further analyzed in [26], whereas Metzen et al. [24] also showed their existence for semantic image segmentation. To date, no dedicated technique exists for defending the networks against the universal adversarial perturbations, which is the topic of this paper.

3. Problem formulation

Below, we present the notions of *universal adversarial perturbations* and the *defense* against them more formally. Let $\mathfrak{S}_c \in \mathbb{R}^{\mathrm{d}}$ denote the distribution of the (clean) natural images in a d-dimensional space, such that, a class label is associated with its every sample $\mathbf{I}_c \sim \mathfrak{S}_c$. Let $\mathcal{C}(.)$ be a classifier (a deep network) that maps an image to its class label, i.e. $\mathcal{C}(\mathbf{I}_c) : \mathbf{I}_c \rightarrow \ell \in \mathbb{R}$. The vector $\boldsymbol{\rho} \in \mathbb{R}^{\mathrm{d}}$ is a universal adversarial perturbation for the classifier, if it satisfies the following constraint:

$$\underset{\mathbf{I}_c \sim \mathfrak{S}_c}{\mathrm{P}} \Big(\mathcal{C}(\mathbf{I}_c) \neq \mathcal{C}(\mathbf{I}_c + \boldsymbol{\rho}) \Big) \geq \delta \quad \text{s.t.} \;\; ||\boldsymbol{\rho}||_p \leq \xi, \qquad (1)$$

where P(.) is the probability, $||.||_p$ denotes the ℓ_p-norm of a vector such that $p \in [1, \infty)$, $\delta \in (0, 1]$ denotes the *fooling ratio* and ξ is a pre-defined constant. In the text to follow, we alternatively refer to $\boldsymbol{\rho}$ as the *perturbation* for brevity.

In (1), the perturbations in question are *image-agnostic*, hence Moosavi-Dezfooli et al. [25] termed them *universal*[2]. According to the stated definition, the parameter ξ controls the norm of the perturbation. For the *quasi-imperceptible* perturbations, the value of this parameter should be very small as compared to the image norm $||\mathbf{I}_c||_p$. On the other hand, a larger δ is required for the perturbation to fool the classifier with a higher probability. In this work, we let $\delta \geq 0.8$ and consider the perturbations constrained by their ℓ_2 and ℓ_∞ norms. For the ℓ_2-norm, we let $\xi = 2,000$, and select $\xi = 10$ for the ℓ_∞-norm perturbations. For both types, these values are $\sim 4\%$ of the means of the respective image norms used in our experiments (in Section 5), which is the same as [25].

To defend $\mathcal{C}(.)$ against the perturbations, we seek two components of the defense mechanism. (1) A perturbation 'detector' $\mathcal{D}(\mathbf{I}_{\boldsymbol{\rho}/c}) : \mathbf{I}_{\boldsymbol{\rho}/c} \rightarrow [0, 1]$ and (2) a perturbation 'rectifier' $\mathcal{R}(\mathbf{I}_{\boldsymbol{\rho}}) : \mathbf{I}_{\boldsymbol{\rho}} \rightarrow \widehat{\mathbf{I}}$, where $\mathbf{I}_{\boldsymbol{\rho}} = \mathbf{I}_c + \boldsymbol{\rho}$. The detector determines whether an unseen image $\mathbf{I}_{\boldsymbol{\rho}/c}$ is perturbed or clean. The objective of the rectifier is to compute a transformation $\widehat{\mathbf{I}}$ of the perturbed image such that $\underset{\mathbf{I}_c \sim \mathfrak{S}_c}{\mathrm{P}} \Big(\mathcal{C}(\widehat{\mathbf{I}}) = \mathcal{C}(\mathbf{I}_c) \Big) \approx 1$. Notice that the rectifier does not seek to improve the prediction of $\mathcal{C}(.)$ on the rectified version of the image beyond the classifier's performance on the clean/original image. This ensures stable induction of $\mathcal{R}(.)$. Moreover, the formulation allows us to compute $\widehat{\mathbf{I}}$ such that $||\widehat{\mathbf{I}} - \mathbf{I}_c||_2 > 0$. We leverage this property to learn $\mathcal{R}(.)$ as the pre-input layers of $\mathcal{C}(.)$ in an end-to-end fashion.

4. Proposed approach

We draw on the insights from the literature reviewed in Section 2 to develop a framework for defending a (possibly) targeted network model against universal adversarial perturbations. Figure 2 shows the schematics of our approach to learn the 'rectifier' and the 'detector' components of the defense framework. We use the Perturbation Rectifying Network (PRN) as the 'rectifier', whereas a binary classifier is eventually trained to detect the adversarial perturbations in the images. The framework uses both real and synthetic perturbations for training. The constituents of the proposed framework are explained below.

4.1. Perturbation Rectifying Network (PRN)

At the core of our technique is the *Perturbation Rectifying Network* (PRN), that is trained as pre-input layers to the targeted network classifier. The PRN is attached to the first layer of the classification network and the joint network is trained to minimize the following cost:

$$\mathcal{J}(\boldsymbol{\theta}_p, \mathbf{b}_p) = \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(\ell_i^*, \ell_i), \qquad (2)$$

where ℓ_i^* and ℓ_i are the labels predicted by the joint network and the targeted network respectively, such that ℓ_i is necessarily computed for the clean image. For the N training examples, $\mathcal{L}(.)$ computes the loss, whereas $\boldsymbol{\theta}_p$ and $\mathbf{b}_p$ denote the PRN weight and bias parameters.

In Eq. (2) we define the cost over the parameters of PRN only, which ensures that the (already deployed) targeted net-

[2] A single perturbation that satisfies (1) for *any* classifier is referred as 'doubly universal' by Moosavi-Dezfooli et al. [25]. We focus on the singly universal perturbations in this work.

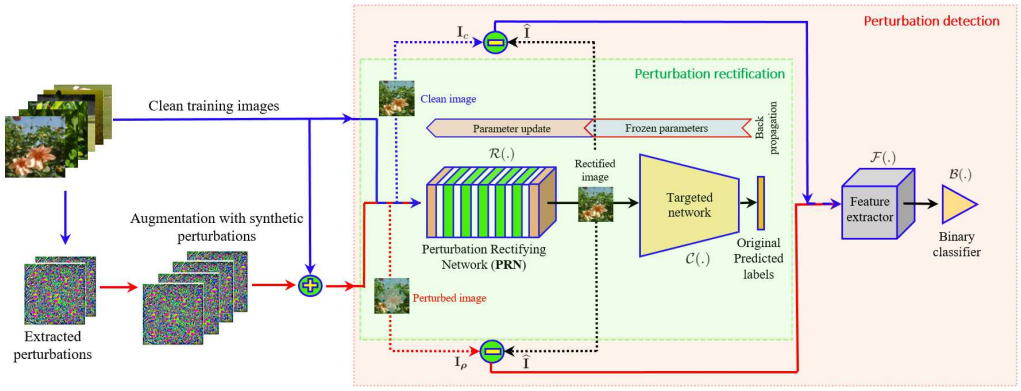

Figure 2. Training schematics: From the clean data, image-agnostic perturbations are computed and augmented with the synthetic perturbations. Both clean and perturbed images are fed to the Perturbation Rectifying Network (PRN). The PRN is learned by attaching it to the first layer of the targeted network such that the parameters of the targeted network are kept frozen during the PRN training. The perturbation detection mechanism extracts discriminative features from the difference between the inputs and outputs of the PRN and learns a binary classifier. To classify an unseen test image $\mathbf{I}_{\boldsymbol{\rho}/c}$, first $\mathcal{D}(\mathbf{I}_{\boldsymbol{\rho}/c}) = \mathcal{B}(\mathcal{F}(\mathbf{I}_{\boldsymbol{\rho}/c} - \mathcal{R}(\mathbf{I}_{\boldsymbol{\rho}/c})))$ is computed. If a perturbation is detected then $\mathcal{R}(\mathbf{I}_{\boldsymbol{\rho}/c})$ is used as the input to the classifier $\mathcal{C}(.)$ instead of the actual test image.

work does not require any modification for the defense being provided by our framework. This strategy is orthogonal to the existing defense techniques that either update the targeted model using adversarial training to make the networks more robust [18], [42]; or incorporate architectural changes to the targeted network, which may include adding a subnetwork to the model [23] or tapping into the activations of certain layers to detect the adversarial examples [20]. Our defense mechanism acts as an external wrapper for the targeted network such that the PRN (and the detector) trained to counter the adversarial attacks can be kept secretive in order refrain from potential counter-counter attacks[3]. This is a highly desirable property of defense frameworks in the real-world scenarios. Moosavi-Dezfooli [25] noted that the universal adversarial perturbations can still exist for a model even after their adversarial training. The proposed framework constitutionally caters for this problem.

We train the PRN using both clean and adversarial examples to ensure that the image transformation learned by our network is not biased towards the adversarial examples. For training, ℓ_i is computed separately with the targeted network for the clean version of the i^{th} training example. The PRN is implemented as 5-ResNet blocks [12] sandwiched by convolution layers. The $224 \times 224 \times 3$ input image is fed to Conv 3×3, stride = 1, feature maps = 64, 'same' convolution; followed by 5 ResNet blocks, where each block consists of two convolution layers with ReLU activations [28], resulting in 64 feature maps. The feature maps of the last ResNet block are processed by Conv 3×3, stride = 1, feature maps = 16, 'same' convolution; and then Conv 3×3, stride = 1, feature maps = 3, 'same' convolution.

[3]PRN+targeted network are end-to-end differentiable and the joint network can be susceptible to stronger attacks if PRN is not secretive. However, stronger perturbations are also more easily detectable by our detector.

We use the cross-entropy loss [9] for training the PRN with the help of ADAM optimizer [15]. The exponential decay rates for the first and the second moment estimates are set to 0.9 and 0.999 respectively. We set the initial learning rate to 0.01, and decay it by 10% after each 1K iterations. We used mini-batch size of 64, and trained the PRN for a given targeted network for at least 5 epochs.

4.2. Training data

The PRN is trained using clean images as well as their adversarial counterparts, constructed by adding perturbations to the clean images. We compute the latter by first generating a set of perturbations $\boldsymbol{\rho} \in \mathcal{P} \subseteq \mathbb{R}^{\text{d}}$ following Moosavi-Dezfooli et al. [25]. Their algorithm computes a universal perturbation in an iterative manner. In its inner loop (ran over the training images), the algorithm seeks a minimal norm vector [27] to fool the network on a given image. The current estimate of $\boldsymbol{\rho}$ is updated by adding to it the sought vector and back-projecting the resultant vector onto the ℓ_p ball of radius ξ. The outer loop ensures that the desired fooling ratio is achieved over the complete training set. Generally, the algorithm requires several passes on the training data to achieve an acceptable fooling ratio. We refer to [25] for further details on the algorithm.

A PRN trained with more adversarial patterns underlying the training images is expected to perform better. However, it becomes computationally infeasible to generate a large (e.g. > 100) number of perturbations using the above-mentioned algorithm. Therefore, we devise a mechanism to efficiently generate synthetic perturbations $\boldsymbol{\rho}_s \in \mathcal{P}_s \subseteq \mathbb{R}^{\text{d}}$ to augment the set of available perturbations for training the PRN. The synthetic perturbations are computed using the set $\mathcal{P}$ while capitalizing on the theoretical results of [26]. To generate the synthetic perturbations, we com-

Algorithm 1 ℓ_∞-norm synthetic perturbation generation

Input: Pre-generated perturbation samples $\mathcal{P} \subseteq \mathbb{R}^{\mathrm{d}}$, number of new samples to be generated η, threshold ξ.
Output: Synthetic perturbations $\mathcal{P}_s \subseteq \mathbb{R}^d$

1: set $\mathcal{P}_s = \{\}$; ℓ_2-threshold $= \mathbb{E}\left[\{||\boldsymbol{\rho}_{i\in\mathcal{P}}||_2\}_{i=1}^{|\mathcal{P}|}\right]$; $\mathcal{P}_n = \mathcal{P}$ with ℓ_2-normalized elements.
2: **while** $|\mathcal{P}_s| < \eta$ **do**
3: set $\boldsymbol{\rho}_s = \mathbf{0}$
4: **while** $||\boldsymbol{\rho}_s||_\infty < \xi$ **do**
5: $z \sim \text{unif}(0,1) \odot \xi$
6: $\boldsymbol{\rho}_s = \boldsymbol{\rho}_s + (z \odot \overset{\text{rand}}{\sim} \mathcal{P}_n)$
7: **end while**
8: **if** $||\boldsymbol{\rho}_s||_2 \geq \ell_2$-threshold **then**
9: $\mathcal{P}_s = \mathcal{P}_s \bigcup \boldsymbol{\rho}_s$
10: **end if**
11: **end while**
12: return

pute the vectors that satisfy the following conditions: (c1) $\boldsymbol{\rho}_s \in \boldsymbol{\Psi}_{\mathcal{P}}^+ : \boldsymbol{\Psi}_{\mathcal{P}}^+ =$ positive orthant of the subspace spanned by the elements of $\mathcal{P}$. (c2) $||\boldsymbol{\rho}_s||_2 \approx \mathbb{E}\left[||\boldsymbol{\rho}||_2, \forall \boldsymbol{\rho} \in \mathcal{P}\right]$ and (c3)[4] $||\boldsymbol{\rho}_s||_\infty \approx \xi$. The procedure for computing the synthetic perturbations that are constrained by their ℓ_∞-norm is summarized in Algorithm 1. We refer to the supplementary material of the paper for the algorithm to compute the ℓ_2-norm perturbations.

To generate a synthetic perturbation, Algorithm 1 searches for $\boldsymbol{\rho}_s$ in $\boldsymbol{\Psi}_{\mathcal{P}}^+$ by taking small random steps in the directions governed by the unit vectors of the elements of $\mathcal{P}$. The random walk continues until the ℓ_∞-norm of $\boldsymbol{\rho}_s$ remains smaller than ξ. The algorithm selects the found $\boldsymbol{\rho}_s$ as a valid perturbation if the ℓ_2-norm of the vector is comparable to the Expected value of the ℓ_2-norms of the vectors in $\mathcal{P}$. For generating the ℓ_2-norm perturbations, the corresponding algorithm given in the supplementary material terminates the random walk based on $||\boldsymbol{\rho}_s||_2$ in line-4, and directly selects the computed $\boldsymbol{\rho}_s$ as the desired perturbation. Analyzing the robustness of the deep networks against the universal adversarial perturbations, Moosavi-Dezfooli [26] showed the existence of shared directions (across different data points) along which a decision boundary induced by a network becomes highly positively curved. Along these vulnerable directions, small universal perturbations exist that can fool the network to change its predictions about the labels of the data points. Our algorithms search for the synthetic perturbations along those directions, whereas the knowledge of the desired directions is borrowed from $\mathcal{P}$.

Fig. 3 exemplifies the typical synthetic perturbations generated by our algorithms for the ℓ_2 and ℓ_∞ norms. It also shows the corresponding closest matches in the set $\mathcal{P}$ for the given perturbations. The fooling ratios for the synthetic perturbations is generally not as high as the original ones, nevertheless the values remain in an acceptable range. In our experiments (Section 5), augmenting the training data with the synthetic perturbations consistently helped in early convergence and better performance of the PRN. We note that the acceptable fooling ratios demonstrated by the synthetic perturbations in this work complement the theoretical findings in [26]. Once the set of synthetic perturbations $\mathcal{P}_s$ is computed, we construct $\mathcal{P}^* = \mathcal{P} \bigcup \mathcal{P}_s$ and use it to perturb the images in our training data.

[4]For the perturbations restricted by their ℓ_2-norm only, this condition is ignored. In that case, (c2) automatically ensures $||\boldsymbol{\rho}_s||_2 \approx \xi$.

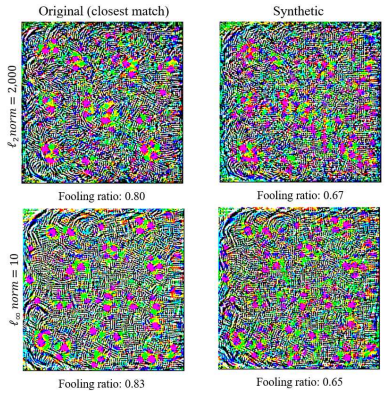

Figure 3. Illustration of synthetic perturbations computed for the CaffeNet [16]: The corresponding closest matches in set $\mathcal{P}$ are also shown. The dot product between the vectorized perturbations with their closest matches are 0.71 and 0.83 respectively for the ℓ_2 and ℓ_∞-norm perturbations.

4.3. Perturbation detection

While studying the JPG compression as a mechanism to mitigate the effects of the (image-specific) adversarial perturbations, Dziugaite et al. [5] also suggested the Discrete Cosine Transform (DCT) as a possible candidate to reduce the effectiveness of the perturbations. Our experiments, reported in supplementary material, show that the DCT based compression can also be exploited to reduce the network fooling ratios under the universal adversarial perturbations. However, it becomes difficult to decide on the required compression rate, especially when it is not known whether the image in question is actually perturbed or not. Unnecessary rectification often leads to degraded performance of the networks on the clean images.

Instead of using the DCT to remove the perturbations, we exploit it for perturbation detection in our approach. Using the training data that contains both clean and perturbed images, say $\mathbf{I}_{\boldsymbol{\rho}/c}^{\text{train}}$, we first compute $\mathcal{F}(\mathbf{I}_{\boldsymbol{\rho}/c}^{\text{train}} - \mathcal{R}(\mathbf{I}_{\boldsymbol{\rho}/c}^{\text{train}}))$ and then learn a binary classifier $\mathcal{B}(\mathcal{F}) \rightarrow [0,1]$ with the data labels denoting the input being 'clean' or 'perturbed'. We implement $\mathcal{F}(.)$ to compute the log-absolute values of the 2D-DCT coefficients of the gray-scaled image in the ar-

gument, whereas an SVM is learned as $\mathcal{B}(.)$. The function $\mathcal{D}(.) = \mathcal{B}(\mathcal{F}(.))$ forms the detector component of our defense framework. To classify a test image $\mathbf{I}_{\boldsymbol{\rho}/c}$, we first evaluate $\mathcal{D}(\mathbf{I}_{\boldsymbol{\rho}/c})$, and if a perturbation is detected then $\mathcal{C}(\mathcal{R}(\mathbf{I}_{\boldsymbol{\rho}/c}))$ is evaluated for classification instead of $\mathcal{C}(\mathbf{I}_{\boldsymbol{\rho}/c})$, where $\mathcal{C}(.)$ denotes the targeted network classifier.

5. Experiments

We evaluated the performance of our technique by defending CaffeNet [16], VGG-F network [4] and GoogLeNet [37] against universal adversarial perturbations. The choice of the networks is based on the computational feasibility of generating the perturbations for our experimental protocol. The same framework is applicable to other networks. Following Moosavi-Dezfooli [25], we used the ILSVRC 2012 [16] validation set of $50,000$ images to perform the experiments.

Setup: From the available images, we randomly selected $10,000$ samples to generate a total of 50 image-agnostic perturbations for each network, such that 25 of those perturbations were constrained to have ℓ_∞-norm equal to 10, whereas the ℓ_2-norm of the remaining 25 was restricted to $2,000$. The fooling ratio of all the perturbations was lower-bounded by 0.8. Moreover, the maximum dot product between any two perturbations of the same type (i.e. ℓ_2 or ℓ_∞) was upper bounded by 0.15. This ensured that the constructed perturbations were significantly different from each other, thereby removing any potential bias from our evaluation. From each set of the 25 perturbations, we randomly selected 20 perturbations to be used with the training data, and the remaining 5 were used with the testing data.

We extended the sets of the training perturbations using the method discussed in Section 4.2, such that there were total 250 perturbations in each extended set, henceforth denoted as $\mathcal{P}^*_\infty$ and $\mathcal{P}^*_2$. To generate the training data, we first randomly selected $40,000$ samples from the available images and performed 5 corner crops of dimensions $224 \times 224 \times 3$ to generate $200,000$ samples. For creating the adversarial examples with the ℓ_2-type perturbations, we used the set $\mathcal{P}^*_2$ and randomly added perturbations to the images with 0.5 probability. This resulted in $\sim 100,000$ samples each for the clean and the perturbed images, which were used to train the approach for the ℓ_2-norm perturbations for a given network. We repeated this procedure using the set $\mathcal{P}^*_\infty$ to separately train it for the ℓ_∞-type perturbations. Note that, for a given targeted network we performed the training twice to evaluate the performance of our technique for both types of perturbations.

For a thorough evaluation, two protocols were followed to generate the testing data. Both protocols used the unseen $10,000$ images that were perturbed with the 5 unseen testing perturbations. Notice that the evaluation has been kept doubly-blind to emulate the real-world scenario for a deployed network. For Protocol-A, we used the whole $10,000$ test images and randomly corrupted them with the 5 test perturbations with a 0.5 probability. For the Protocol-B, we chose the subset of the $10,000$ test images that were correctly classified by the targeted network in their clean form, and corrupted that subset with 0.5 probability using the 5 testing perturbations. The existence of both clean and perturbed images with equal probability in our test sets especially ensures a fair evaluation of the detector.

Evaluation metric: We used four different metrics for a comprehensive analysis of the performance of our technique. Let $\mathcal{I}_c$ and $\mathcal{I}_{\boldsymbol{\rho}}$ denote the sets containing clean and perturbed test images. Similarly, let $\widehat{\mathcal{I}}_{\boldsymbol{\rho}}$ and $\widehat{\mathcal{I}}_{\boldsymbol{\rho}/c}$ be the sets containing the test images rectified by PRN, such that all the images in $\widehat{\mathcal{I}}_{\boldsymbol{\rho}}$ were perturbed (before passing through the PRN) whereas the images in $\widehat{\mathcal{I}}_{\boldsymbol{\rho}/c}$ were similarly perturbed with 0.5 probability, as per our protocol. Let $\overset{*}{\mathcal{I}}$ be the set comprising the test images such that each image is rectified by the PRN only if it were classified as perturbed by the detector $\mathcal{D}$. Furthermore, let $acc(.)$ be the function computing the prediction accuracy of the target network on a given set of images. The formal definitions of the metrics that we used in our experiments are stated below:

1. PRN-gain (%) $= \frac{acc(\widehat{\mathcal{I}}_{\boldsymbol{\rho}}) - acc(\mathcal{I}_{\boldsymbol{\rho}})}{acc(\widehat{\mathcal{I}}_{\boldsymbol{\rho}})} \times 100.$
2. PRN-restoration (%) $= \frac{acc(\widehat{\mathcal{I}}_{\boldsymbol{\rho}/c})}{acc(\mathcal{I}_c)} \times 100.$
3. Detection rate (%) = Accuracy of $\mathcal{D}$.
4. Defense rate (%) $= \frac{acc(\overset{*}{\mathcal{I}})}{acc(\mathcal{I}_c)} \times 100.$

The names of the metric are in accordance with the semantic notions associated with them. Notice that the PRN-restoration is defined over the rectification of both clean and perturbed images. We do this to account for any loss in the classification accuracy of the targeted network incurred by the rectification of the clean images by the PRN. It was observed in our experiments that unnecessary rectification of the clean images can sometimes lead to a minor (1 - 2%) reduction in the classification accuracy of the targeted network. Hence, we used a more strict definition of the restoration by PRN for a more transparent evaluation. This definition is also in-line with our underlying assumption of the practical scenarios where we do not know *a prior* if the test image is clean or perturbed.

Same/Cross-norm evaluation: In Table 1, we summarize the results of our experiments for defending the GoogLeNet [37] against the perturbations. The table summarizes two kinds of experiments. For the first kind, we used the same types of perturbations for testing and training. For instance, we used the ℓ_2-type perturbations for learning the framework components (rectifier + detector) and then

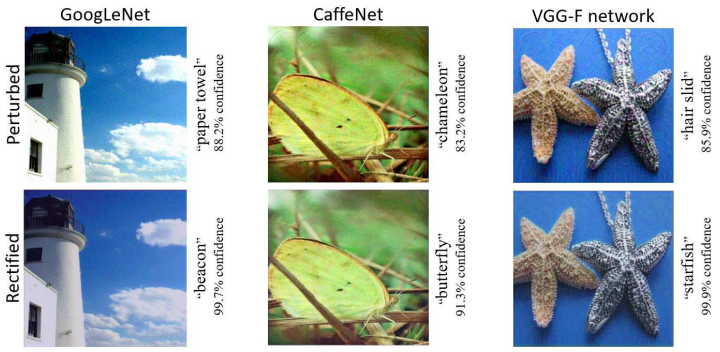

Figure 4. Representative examples to visualize the perturbed images and their rectified version computed by the PRN. The labels predicted by the networks along the prediction confidence are also given. The examples are provided for the ℓ_∞-type perturbations. Please refer to the supplementary material of the paper for more examples.

Table 1. Defense summary for the **GoogLeNet** [37]: The mentioned types of the perturbations (i.e. ℓ_2 or ℓ_∞) are for the testing data.

Metric	**Same test/train perturbation type**				**Different test/train perturbation type**			
	ℓ_2-type		ℓ_∞-type		ℓ_2-type		ℓ_∞-type	
	Prot-A	Prot-B	Prot-A	Prot-B	Prot-A	Prot-B	Prot-A	Prot-B
PRN-gain (%)	77.0	77.1	73.9	74.2	76.4	77.0	72.6	73.4
PRN-restoration (%)	97.0	92.4	95.6	91.3	97.1	92.7	93.8	89.3
Detection rate (%)	94.6	94.6	98.5	98.4	92.4	92.3	81.3	81.2
Defense rate (%)	97.4	94.8	96.4	93.7	97.5	94.9	94.3	91.6

also used the ℓ_2-type perturbations for testing. The results of these experiments are summarized in the left half of the table. We performed the 'same test/train perturbation type' experiments for both ℓ_2 and ℓ_∞ perturbations, for both testing protocols (denoted as Prot-A and Prot-B in the table). In the second kind of experiments, we trained our framework on one type of perturbation and tested for the other. The right half of the table summarizes the results of those experiments. The mentioned perturbation types in the table are for the testing data. The same conventions will be followed in the similar tables for the other two targeted networks below. Representative examples to visualize the perturbed and rectified images (by the PRN) are shown in Fig. 4. Please refer to the supplementary material for more illustrations.

From Table 1, we can see that in general, our framework is able to defend the GoogLeNet very successfully against the universal adversarial perturbations that are specifically targeted at this network. The Prot-A captures the performance of our framework when an attacker might have added a perturbation to an unseen image without knowing if the clean image would be correctly classified by the targeted network. The Prot-B represents the case where the perturbation is added to fool the network on an image that it had previously classified correctly. Note that the difference in the performance of our framework for Prot-A and Prot-B is related to the accuracy of the targeted network on clean images. For a network that is 100% accurate on clean images, the results under Prot-A and Prot-B would match exactly. The results would differ more for the less accurate classifiers, as also evident from the subsequent tables.

In Table 2, we summarize the performance of our framework for the CaffeNet [16]. Again, the results demonstrate a good defense against the perturbations. The final Defense-rate for the ℓ_2-type perturbation for Prot-A is 96.4%. Under the used metric definition and the experimental protocol, one interpretation of this value is as follows. With the defense wrapper provided by our framework, the performance of the CaffeNet is expected to be 96.4% of its original performance (in the perfect world of clean images), such that there is an equal chance of every query image to be perturbed or clean[5]. Considering that the fooling rate of the network was at least 80% on all the test perturbations used in our experiments, it is a good performance recovery.

In Table 3, the defense summary for the VGG-F network [4] is reported, which again shows a decent performance of our framework. Interestingly, for both CaffeNet and VGG-F, the existence of the ℓ_∞-type perturbations in the test images could be detected very accurately by our detector for the 'different test/train perturbation type'. However, it was not the case for the GoogLeNet. We found that for the ℓ_∞-type perturbations (with $\xi = 10$) the corresponding ℓ_2-norm of the perturbations was generally much lower for the GoogLeNet ($\sim 2,400$ on avg.) as compared to the CaffeNet and VGG-F ($\sim 2,850$ on avg.). This made the detection of the ℓ_∞-type perturbations more challenging for the GoogLeNet. The dissimilarity in these values indicate that there is a significant difference between the decision

[5]We emphasize that our evaluation protocols and metrics are carefully designed to analyze the performance in the real-world situations where it is not known apriori whether the query is perturbed or clean.

Table 2. Defense summary for the **CaffeNet [16]:** The mentioned types of the perturbations (i.e. ℓ_2 or ℓ_∞) are for the testing data.

Metric	Same test/train perturbation type				Different test/train perturbation type			
	ℓ_2-type		ℓ_∞-type		ℓ_2-type		ℓ_∞-type	
	Prot-A	Prot-B	Prot-A	Prot-B	Prot-A	Prot-B	Prot-A	Prot-B
PRN-gain (%)	67.2	69.0	78.4	79.1	65.3	66.8	77.3	77.7
PRN-restoration (%)	95.1	89.9	93.6	88.7	92.2	87.1	91.7	85.8
Detection rate (%)	98.1	98.0	97.8	97.9	84.2	84.0	97.9	98.0
Defense rate (%)	96.4	93.6	95.2	92.5	93.6	90.1	93.2	90.0

Table 3. Defense summary for the **VGG-F** network [4]: The mentioned types of the perturbations (i.e. ℓ_2 or ℓ_∞) are for the testing data.

Metric	Same test/train perturbation type				Different test/train perturbation type			
	ℓ_2-type		ℓ_∞-type		ℓ_2-type		ℓ_∞-type	
	Prot-A	Prot-B	Prot-A	Prot-B	Prot-A	Prot-B	Prot-A	Prot-B
PRN-gain (%)	72.1	73.3	84.1	84.3	68.3	69.2	84.7	84.8
PRN-restoration (%)	93.2	86.2	90.3	83.2	88.8	81.2	91.1	83.3
Detection rate (%)	92.5	92.5	98.6	98.6	92.5	92.5	98.1	98.1
Defense rate (%)	95.5	91.4	92.2	87.9	90.0	85.9	93.7	89.1

boundaries induced by the GoogLeNet and the other two networks, which is governed by the significant architectural differences of the networks.

Cross-architecture generalisation: With the above observation, it was anticipated that the cross-network defense performance of our framework would be better for the networks with the (relatively) similar architectures. This prediction was verified by the results of our experiments in Tables 4 and 5. These tables show the performance for ℓ_2 and ℓ_∞-type perturbations where we used the 'same test/train perturbation type'. The results are reported for protocol A. For the corresponding results under protocol B, we refer to the supplementary material. From these tables, we can conclude that our framework generalizes well across different networks, especially across the networks that have (relatively) similar architectures. We conjecture that the cross-network generalization is inherited by our framework from the cross-model generalization of the universal adversarial perturbations. Like our technique, any framework for the defense against these perturbations can be expected to exhibit similar characteristics.

Table 4. ℓ_2**-type cross-network defense** (Prot-A): Testing is done using the perturbations generated on the networks in the left-most column. The networks to generate the training perturbations are indicated in the second row.

PRN-restoration (%)			
	VGG-F	CaffeNet	GoogLeNet
VGG-F [4]	93.2	88.9	81.7
CaffeNet [16]	91.3	95.1	72.0
GoogLeNet [37]	84.7	85.9	97.0
Defense rate (%)			
	VGG-F	CaffeNet	GoogLeNet
VGG-F [4]	95.5	91.5	82.4
CaffeNet [16]	94.8	96.2	77.3
GoogLeNet [37]	88.3	87.3	97.4

Table 5. ℓ_∞**-type cross-network defense** summary (Prot-A).

PRN-restoration (%)			
	VGG-F	CaffeNet	GoogLeNet
VGG-F [4]	90.3	86.9	74.1
CaffeNet [16]	85.7	93.6	69.3
GoogLeNet [37]	85.9	83.3	95.6
Defense rate (%)			
	VGG-F	CaffeNet	GoogLeNet
VGG-F [4]	92.2	88.9	74.8
CaffeNet [16]	93.5	95.2	73.8
GoogLeNet [37]	88.4	85.4	96.4

6. Conclusion

We presented the first dedicated framework for the defense against *universal adversarial perturbations* [25] that not only detects the presence of these perturbations in the images but also rectifies the perturbed images so that the targeted classifier can reliably predict their labels. The proposed framework provides defense to a targeted model without the need of modifying it, which makes our technique highly desirable for the practical cases. Moreover, to prevent the potential counter-counter measures, it provides the flexibility of keeping its 'rectifier' and 'detector' components secretive. We implement the 'rectifier' as a Perturbation Rectifying Network (PRN), whereas the 'detector' is implemented as an SVM trained by exploiting the image transformations performed by the PRN. For an effective training, we also proposed a method to efficiently compute image-agnostic perturbations synthetically. The efficacy of our framework is demonstrated by a successful defense of CaffeNet [16], VGG-F network [4] and GoogLeNet [37] against the universal adversarial perturbations.

Acknowledgement This research was supported by ARC grant DP160101458. The Titan Xp used for this research was donated by NVIDIA Corporation.

References

[1] N. Akhtar and A. Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *arXiv preprint arXiv:1801.00553*, 2018.

[2] S. Baluja and I. Fischer. Adversarial transformation networks: Learning to generate adversarial examples. *arXiv preprint arXiv:1703.09387*, 2017.

[3] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *Security and Privacy (SP), 2017 IEEE Symposium on*, pages 39–57. IEEE, 2017.

[4] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. *arXiv preprint arXiv:1405.3531*, 2014.

[5] G. K. Dziugaite, Z. Ghahramani, and D. M. Roy. A study of the effect of jpg compression on adversarial images. *arXiv preprint arXiv:1608.00853*, 2016.

[6] A. Fawzi, O. Fawzi, and P. Frossard. Analysis of classifiers' robustness to adversarial perturbations. *arXiv preprint arXiv:1502.02590*, 2015.

[7] A. Fawzi, S.-M. Moosavi-Dezfooli, and P. Frossard. Robustness of classifiers: from adversarial to random noise. In *Advances in Neural Information Processing Systems*, pages 1632–1640, 2016.

[8] V. Fischer, M. C. Kumar, J. H. Metzen, and T. Brox. Adversarial examples for semantic image segmentation. *arXiv preprint arXiv:1703.01101*, 2017.

[9] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. 2016.

[10] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[11] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[12] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[13] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[14] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten. Densely connected convolutional networks. *arXiv preprint arXiv:1608.06993*, 2016.

[15] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[17] A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

[18] A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016.

[19] Y. Liu, X. Chen, C. Liu, and D. Song. Delving into transferable adversarial examples and black-box attacks. *arXiv preprint arXiv:1611.02770*, 2016.

[20] J. Lu, T. Issaranon, and D. Forsyth. Safetynet: Detecting and rejecting adversarial examples robustly. *arXiv preprint arXiv:1704.00103*, 2017.

[21] J. Lu, H. Sibai, E. Fabry, and D. Forsyth. No need to worry about adversarial examples in object detection in autonomous vehicles. *arXiv preprint arXiv:1707.03501*, 2017.

[22] Y. Luo, X. Boix, G. Roig, T. Poggio, and Q. Zhao. Foveation-based mechanisms alleviate adversarial examples. *arXiv preprint arXiv:1511.06292*, 2015.

[23] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff. On detecting adversarial perturbations. *arXiv preprint arXiv:1702.04267*, 2017.

[24] J. H. Metzen, M. C. Kumar, T. Brox, and V. Fischer. Universal adversarial perturbations against semantic image segmentation. *arXiv preprint arXiv:1704.05712*, 2017.

[25] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard. Universal adversarial perturbations. *CVPR*, 2017.

[26] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, P. Frossard, and S. Soatto. Analysis of universal adversarial perturbations. *arXiv preprint arXiv:1705.09554*, 2017.

[27] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2574–2582, 2016.

[28] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[29] A. Nguyen, J. Yosinski, and J. Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 427–436, 2015.

[30] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *Security and Privacy (SP), 2016 IEEE Symposium on*, pages 582–597. IEEE, 2016.

[31] A. Prakash, N. Moran, S. Garber, A. DiLillo, and J. Storer. Deflecting adversarial attacks with pixel deflection. *arXiv preprint arXiv:1801.08926*, 2018.

[32] A. Rozsa, E. M. Rudd, and T. E. Boult. Adversarial diversity and hard positive generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 25–32, 2016.

[33] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[34] S. Sabour, Y. Cao, F. Faghri, and D. J. Fleet. Adversarial manipulation of deep representations. *arXiv preprint arXiv:1511.05122*, 2015.

[35] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1528–1540. ACM, 2016.

[36] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[37] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[38] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

[39] 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.

[40] P. Tabacof and E. Valle. Exploring the space of adversarial images. In *Neural Networks (IJCNN), 2016 International Joint Conference on*, pages 426–433. IEEE, 2016.

[41] T. Tanay and L. Griffin. A boundary tilting persepective on the phenomenon of adversarial examples. *arXiv preprint arXiv:1608.07690*, 2016.

[42] F. Tramèr, A. Kurakin, N. Papernot, D. Boneh, and P. McDaniel. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017.

[43] C. Xie, J. Wang, Z. Zhang, Y. Zhou, L. Xie, and A. Yuille. Adversarial examples for semantic segmentation and object detection. *arXiv preprint arXiv:1703.08603*, 2017.