Prepint of paper to appear in IEEE Signal Processing Mag. Nov 2017

On the Robustness of Deep Neural Networks

Manuel Günther, Andras Rozsa, and Terrance E. Boult*

Vision and Security Technology Lab, University of Colorado Colorado Springs

{mgunther,arozsa,tboult}@vast.uccs.edu

May 27, 2017

1 Vulnerability of Deep Neural Networks

Deep Neural Networks (DNNs) have become the quasi-standard in many machine learning tasks since they obtain state-of-the-art results and outperform more traditional machine learning models for problems in vision, image and speech processing, and many other tasks. One reason that DNNs have proven very useful is that they are one of the few algorithms that can make principled use of huge databases for training. However, despite their brilliant recognition capabilities, little is known about how DNNs achieve their accuracies, and even less is known about their robustness and their limitations. After Szegedy *et al.* [19] showed that deep networks have some "intriguing" properties, several researchers started actively exploring the limitations, and showed how easily the networks could be attacked or fooled – demonstrating essential limits of the robustness of DNNs. This paper analyzes these two categories of attacks: adversarial images [19, 3], which are imperceptible perturbations to an input to turn it into another class, and fooling images [9], which look like none of the

*This research is based upon work funded in part by NSF IIS-1320956 and in part by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via IARPA R&D Contract No. 2014-14071600012. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.

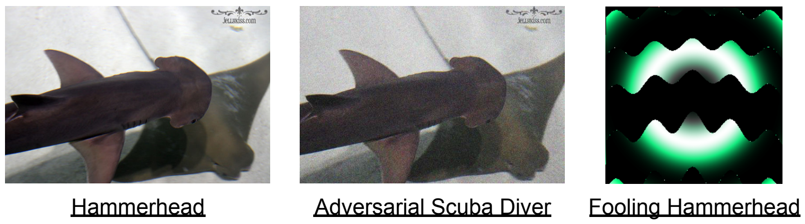


Figure 1: Adversarial and Fooling Images. *Examples show an original image classified as hammerhead, an associated adversarial image, and an associated fooling image. The adversarial image has added noise after which the network incorrectly classifies it as a scuba diver. The fooling image looks nothing like normal images, but the network classifies it as hammerhead shark with very high confidence. This paper discussed issues related to such adversarial and fooling images including how to improve robustness by mitigating their impact.*

classes, see Fig. 1.

While a deep network will always have a most-likely class, one might hope that for an unknown input, all classes would have a low probability and that thresholding on uncertainty would reject unknown classes. Recent papers have shown how to produce "fooling" [9], "rubbish" [3] and adversarial images [19, 3, 8] that are visually far from the labeled class, but produce high-confidence scores for that label. Thresholding on uncertainty is not sufficient to determine what is unknown, illustrating that there are fundamental issues that need to be understood about the robustness of deep networks. Thus, fooling and adversarial images are directly an issue of network robustness; we briefly review these concepts and then summarize the contributions of this paper.

1.1 Fooling Images

Nguyen *et al.* [9] used images containing random and non-random patterns, which are far from any class that the network has learned to predict. By applying small non-random perturbations to these images, the DNN will predict, whatever you want it to predict. Particularly, given a starting image, a DNN that was trained to predict classes, and the desired output class, a perturbation is computed using gradient ascent [9], which will increase the

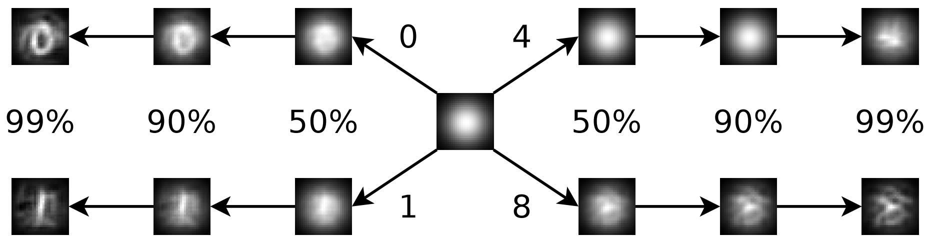


Figure 2: Fooling Images. *Fooling images for hand-written digits were generated on a LeNet pretrained on the MNIST image training set. Starting from the pattern in the center, fooling images for different labels and three different desired classification confidences (F50, F90 and F99) are produced.*

probability of the desired class. This procedure is iteratively executed until the classification probability reached the desired confidence level.

Fig. 2 shows some examples of fooling images. Starting from a pattern in the center the image is evolved to increase the DNN's classification as one of the digits: 0, 1, 4 and 8. The images with 50 % confidence, which we call F50, do not look much alike the desired class. For labels 0 and 1, images that are classified with higher confidences start to resemble the desired digit – at least for this starting image. For labels 4 and 8, the digit is not well pronounced even though they have moved relatively far from the starting image. Note that Nguyen *et al.* [9] present other fooling images for the MNIST dataset on their web page.

1.2 Adversarial Images

Although fooling images show weaknesses of deep networks, a different type of threat was introduced by so-called adversarial examples. Similarly to fooling images, adversarial examples are based on perturbations added to images that change the classification. The difference is that for adversarial examples the starting image is a correctly classified and the perturbations are – ideally – imperceptible. Because they are imperceptible, some researchers claim that adversarial images pose a "security threat", an issue we examine later.

After Szegedy *et al.* [19] introduced the concept of adversarial images, several techniques were engineered to create those. The first method that could reliably and efficiently produce

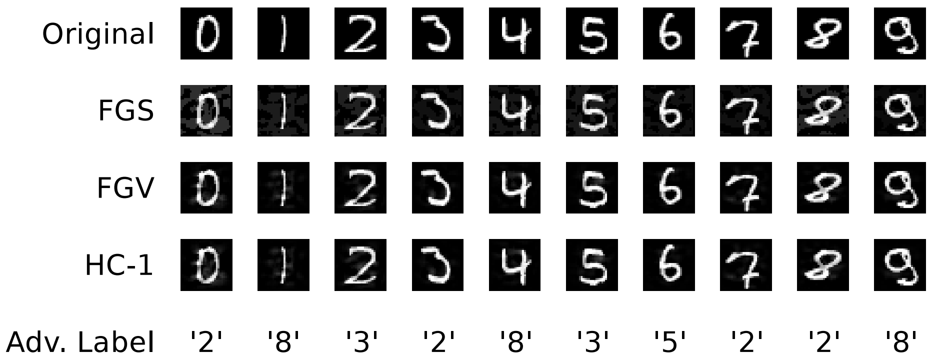


Figure 3: ADVERSARIAL IMAGES. *Adversarial images for hand-written digits were generated on a LeNet pretrained on the MNIST image training set. The first row shows the originating images, the three rows below show the FGS, FGV and HC-1 adversarial images, while their classifications are given below. Though each adversarial image generation type can generate different classification results, images were selected such that they are classified identically (per column).*

adversarial examples was introduced by Goodfellow *et al.* [3]. Their Fast Gradient Sign (FGS) method used the gradient of loss with respect to the input image in order to create perturbations. Given a correctly classified input image x and its gradient ∇x, the adversarial image is $x_{\text{FGS}} = x + w \cdot \text{sign}(\nabla x)$ where w is such that the classification of x_{FGS} just switches to another class. Note that in some papers, a fixed w is used that may produce larger perturbations making them more portable and stable, but also more visible.

Rozsa *et al.* [13] adapted FGS by dropping the *sign*, yielding the Fast Gradient Value (FGV) method: $x_{\text{FGV}} = x + w \cdot \nabla x$. They also introduced the so-called Hot/Cold (HC) approach, generating adversarial perturbations by simultaneously increasing the probability of the target (hot) class while reducing the probability of the original (cold) class. Herein we consider targeting either the closest class, HC-1, or all other classes (HC).

The tutorial code includes methods for generating all three types of adversarial images: FGS, FGV and HC-1. Some of the exemplary images shown in Fig. 3 look noisier and slightly different from their originals. Adversarial images, however, only post a security threat if they are visually indistinguishable from the originals and clearly show the original class, while DNNs are predicting another label. To quantify the quality of the adversarial image, several metrics are used. Goodfellow *et al.* [3], for example, used the L_1, L_2 and L_∞ norms of the perturbation as a quality measure. However, none of these measures are close to human perception. For example, L_∞ only measures the maximum pixel perturbation, independent of if this perturbation was in the object, an edge or in the background. To overcome this inconsistency, Rozsa *et al.* [13] introduces the Perceptual Adversarial Similarity Score (PASS), which is based on human perception. PASS values close to 1 indicate imperceptible perturbations, while smaller PASS values suggest stronger modifications.

While the above works and others intentionally generate adversarial images to defeat deep networks, recent work [12] shows that "adversarial inputs" frequently occur naturally – many errors by networks are images that are a small perturbation from being correct.

1.3 Contributions

Since we still do not know why fooling and adversarial images exist in the first place, an important question is if we can find ways to limit the success of adversarial images. In Sec. 2 we review three different approaches that were proposed in the literature. First, we test how adversarial training [3, 13] can reduce the success rate of adversarial image generation, and we discuss the limitations of such training. Second, experiments by Graese *et al.* [4] used simple image preprocessing techniques to remove adversarial properties. Third, we discuss the work on Open Set Deep Network (OSDN) introduced by Bendale and Boult [1], review their findings on ImageNet data and do a new testing with MNIST to evaluate if we can correct their labels or at least predict the presence of adversarial or fooling images.

Returning to potential causes, an idea independently developed by the authors of [14] and our team is that adversarial images were the result of aliasing. In Sec. 3, we perform experiments that reject this hypothesis by showing that a network that cannot be affected by aliasing is still susceptible to adversarial image generation.

Throughout this article, we review material from the literature, but to help the reader we use tutorial-like code and focus our evaluations on hand-written digit classification using the MNIST dataset, and LeNet, a not-so-deep neural network, which is well-investigated in the literature, while being small enough to easily run experiments on a laptop. The MNIST dataset consists of 70000 images of hand-written digits, which are cropped to be in the center of 28×28 pixel gray-scale images. Examples are displayed in the first row of Fig. 3. LeNet consists of two convolutional layers (conv1 and conv2) followed by two fully-connected layers (ip1 and ip2). The classification is performed using a softmax layer, which turns the ten outputs of layer ip2 into classification probabilities. This example is part of many deep network tutorials, and we expand on that with tutorial-like code for adversarial images, fooling images and Open Set Deep Networks. This code is based on the Caffe framework [5] and implemented using its Python interface. The code requires some small modifications to the original Caffe code – refer to our installation instructions. Our code can be found at `http://bitbucket.org/vastlab/code.sp17`.

2 Techniques to Manage Adversarial and Fooling Images

Recently, researchers have stated that adversarial examples can "seriously undermine the security of the system supported by the DCNN" [11] because the incorrect classification could potentially lead to an incorrect action with consequences. For example, if the digits signifying an amount on a check were crafted to be adversarial, the amount of money transferred between accounts could be altered. Though a "defense" is presented [11], it was later shown [2] to be fundamentally flawed and, hence, provides no real defense. Other researchers have been exploring techniques, which provide some real defense, three of which we explore deeper in this section.

2.1 Does Adversarial Training Improve Robustness?

One simple idea of generating more stable networks is by data augmentation. When a DNN is fooled by small modifications to the input, augmenting the training with these kinds of images should be able to reduce these properties. Indeed, Szegedy *et al.* [19] and Goodfellow *et al.* [3] showed that training a network explicitly or implicitly using adversarial images improves the stability towards those adversarial images. However, they performed experiments only on a single type of adversarial images, i.e., FGS, and they used images that were not adversarial in that the perturbations were clearly visible. From their experiments, it is unclear if such an adversarial training will improve the stability for other adversarial image generation techniques such as FGV or HC.

In our work [13], we explored training with a greater range of adversarial types, showing that increased diversity improved both network accuracy and robustness. That paper showed that going a bit farther than the minimum adversarial, say by 5%, improved accuracy more than just using minimally adversarial images, including showing that such training improved deep networks on ImageNet. However, Rozsa *et al.* [13] just mixed various adversarial image types together and did not study how training with one type impacted robustness with respect to others.

Using the MNIST dataset, we do adversarial training experiments, but this time we evaluate several techniques to compute adversarial images for training, and test robustness across types. First, we train a basic LeNet model on the MNIST training set for 100k iterations. Afterward, we use that network to extract one adversarial image with FGS and for FGV, as well as nine adversarial images for each training set image with HC. For each of the adversarial image types, we fine-tuned the basic LeNet using only the adversarial images of that type. The fine-tuning for FGS and FGV used 20k iterations, while we chose 50k iterations for the HC adversarial images – as we have more of those. The tutorial code includes the networks and the images.

Finally, we evaluated the three fine-tuned LeNets plus our basic LeNet. We tried to extract adversarial images on the MNIST test set using the four networks – it is not sufficient to test adversarial images generated on the basic network since fine-tuning modifies the network's weights such that many of original adversarial images are no longer adversarial. We computed, how often we could find an adversarial perturbation that would change the classification of the fine-tuned network, and computed the average PASS value of the successful cases. The results are presented in Tab. 1. We can first see that fine-tuning

		FGS		**FGV**		**HC**	
Model	**Accuracy**	Rate	PASS	Rate	PASS	Rate	PASS
LeNet	99.09 %	85.37 %	0.43	85.37 %	0.78	99.92 %	0.73
LeNet+FGS	99.23 %	80.69 %	0.48	80.69 %	0.76	100.00 %	0.72
LeNet+FGV	99.18 %	60.52 %	0.42	60.52 %	0.77	99.95 %	0.71
LeNet+HC	99.22 %	71.92 %	0.45	71.92 %	0.77	99.99 %	0.71

Table 1: Training with Adversarial Images. *We compare the original LeNet model with the models fine-tuned with FGS, FGV and HC adversarial images. The second column presents classification accuracy of the networks. The remaining columns provide success rate and average PASS value for adversarial images generated with FGS, FGV, and HC on the MNIST test set.*

with any type of adversarial images improves the (already very high) classification accuracy a little. This is not a surprise, as data augmentation is known to increase classification accuracies. More interestingly, for FGS and FGV the success rate of adversarial images does drop moderately, the highest drop, i.e., from 85 % to 60 % is seen when trained with FGV adversarial images. On the other hand, HC adversarial images could be generated in almost all cases, and adversarial training did not reduce this success rate. In all cases, the quality of adversarial images did not change substantially, indicated by an almost stable average PASS value per technique. Hence, we can conclude that adversarial training can slightly reduce the number of FGS and FGV adversarial images, but HC adversarial images can not be combated with adversarial training, even when training with them.

2.2 Fighting Adversarial Images with Image Preprocessing

Most real world applications of deep learning use inputs from a camera or scanned images, where the input images will never be perfectly captured, but contain slight transformations, such as shifting or blurring, and perturbations, such as noise. Hence, images generally undergo preprocessing before being input to the neural network. It is reasonable to ask if a simple image preprocessing could be sufficient to mitigate the effect of adversarial images.

Graese *et al.* [4] investigated how images would be transformed in a real application, i.e., when (adversarial) images are printed on paper and recaptured for further processing.

Technique	**None**	**Noise**	**Blur**	**Shift**	**Combined**	**Crops**
FGS	0.00 %	28.73 %	60.70 %	67.86 %	64.46 %	90.10 %
FGV	0.00 %	60.51 %	66.63 %	68.58 %	68.26 %	89.47 %
HC-1	0.00 %	59.50 %	66.29 %	68.12 %	67.16 %	89.76 %
Accuracy	99.09 %	99.08 %	98.39 %	98.24 %	97.35 %	99.02 %

Table 2: Fighting Adversarial Images with Preprocessing. *The success of correcting the classification of adversarial images with several image preprocessing techniques is shown. For comparison, images with no perturbation are reported, too. Adversarial images were created with three different techniques: FGS, FGV and HC-1. The last row presents the classification accuracy on the original images that were preprocessed.*

To mimic the image acquisition process, they performed several preprocessing techniques such as shifting or blurring the image slightly, adding random noise to the image before classification, or scaling and re-cropping the image. Graese *et al.* [4] also explored a preprocessing specific to text-like problems, i.e., binarization and found that it was the most effective technique for destroying the tested adversarial image types. However, binarization is rather problem specific and, hence, not included in this analysis.

In our experiments, we use the same basic LeNet and the same adversarial images from the last section, where we apply three types of modifications. First, the image is shifted with one pixel in random direction (left, right, top, bottom) and the removed row or column is added to the other side of the image. Second, the image is blurred with a Gaussian blur kernel with one pixel standard deviation. Third, random normal-distributed noise with zero mean and a standard deviation of one gray level is added to each pixel in the image. Finally, a combination of all processing steps is applied (first shift, then blur, and finally noise). After processing, all pixels are converted to integral values in [0,255].

The results of these different preprocessing steps are presented in Tab. 2. To see if the preprocessing has any impact on the classification accuracy, we classified all test images that were preprocessed with the presented techniques. When shifting and blurring the image, the classification accuracy drops moderately, i.e., from 99.09 % with the original unprocessed images to 98.39 % using blurred images and 98.29 % with shifted images. Interestingly, adding small noise does not affect the classification accuracy at all. The highest drop in performance is obtained when combining all preprocessing techniques.

Further, we tried to classify the adversarial images after shifting, blurring and adding noise. Tab. 2 reports the percentage of adversarial images that were classified correctly after applying preprocessing. Due to the very noisy nature of FGS adversarial images, adding noise did not help too much, only 28 % of those images were classified correctly. FGV and HC-1 adversarial images usually contain less visible perturbations and, hence, adding noise works considerably better. However, other preprocessing techniques such as blurring and shifting removed the adversarial properties of around two thirds of the adversarial images. Combining the techniques did not improve mitigation further.

One technique that is often applied to improve classification accuracy is to take several crops of an image and average the results of these crops into the final classification. Previously, this preprocessing was not added to address adversarial robustness, but rather to improve network performance. However, we argue that these are related, as we show that using crops helps destroy adversarial properties. The performance improvement is likely because it impacts natural adversarials [12], where the networks misclassifies the center crop. While Graese *et al.* [4] rescaled the images before cropping, which is not common practice, we simply padded the images with one black pixel on each side, resulting in a 30×30 pixel image. As common in literature, we take five crops of the image, i.e., top-left, top-right, bottom-left, bottom-right and the center crop, where the latter corresponds to the original image. In literature, images are often also mirrored, but for the digit classification this makes no sense. Hence, we run the five crops trough our network and compute the average of the ip2 features, and classify the image based on this average. The results are given in the last column of Tab. 2. As we can see, using this simple technique already 90 % of the adversarial images are no longer adversarial. Interestingly, this comes at the price of very slightly reduced accuracy, which is almost identical to the accuracy without preprocessing.

As a result, we can see that very small perturbations to the input images can reveal their adversarial properties in about two thirds of the cases. We can assume that adding larger shifts, a stronger blurring and more noise will further increase the number of adversarial images that will be corrected. However, these perturbations may slightly decrease classification accuracy, and stronger perturbations will decrease accuracy even more. Fortunately, as Graese *et al.* [4] demonstrated this decrease in accuracy can be prevented when the original network is trained or fine-tuned using the same perturbations as applied to the test set images.

Graese *et al.* [4] suggest that normal image acquisition would destroy most adversarial images. In recent work, Kurakin *et al.* [6] showed that some adversarial images from ImageNet can printed and recaptured, persisting as adversarial images. However, their approach was rather artificial by printing the images in higher resolution than the original images were. Even for those images, the rate of destroying adversarial properties of high-quality adversarials with invisible noise was around 50–98 %. While this suggests that many adversarial images might survive, its important to recall that the original network accuracy has only approximately 80 % top-1 accuracy, even 20 % of the adversarials surviving is not inherently more of a risk than standard classification errors. We also note, the paper does not discuss multiple crops. Hence, it is not clear if Kurakin *et al.* [6] truly contracts Graese *et al.* [4], but it does suggest that more work is needed to address the potential security concern in real settings. For now, the security concern some researchers are raising may largely be an artifact of not applying normal image preprocessing combined with state of the art deep network techniques, and are certainly a lower risk than some authors suggest.

2.3 Open Set Deep Networks for Adversarial and Fooling Images.

In the previous section, we showed how to detect and correct adversarial images. Fooling images, however, pose a different problem, as there is no notion of *correcting* the images, as they do not represent a class that is known by the classifier. Unfortunately, in general classifiers are *closed set*, and will classify any input image as one of the classes it was trained on. Rather, the classifier should be *open set* and able to *reject* the image as *unknown*.

The first approach extending DNNs to open set recognition was presented by Bendale and Boult [1]. To reject an image as unknown, the representation (activation vector) at a given layer of the DNN was averaged over all images of a class. They call this the Mean Activation Vector (MAV). They then compute an Weibull-based calibration which, together with the MAV, served as a coarse representation of the networks known about a class. When an input image is classified as a given class, the Open Set Deep Network (OSDN) also checks if the representation was close enough to the MAV, and if it was not, the input image can be rejected as *unknown*, or the various class probabilities can be recomputed using *openmax*. The OSDN [1] was a proof of concept using AlexNet on ImageNet, testing with some fooling, adversarial and open-set images, i.e., normal images from unknown classes. Their paper presented the first adversarial image detector and showed that blurring and processing with the OSDN could correct the classification, see Fig. 1. However, their primary focus was on open set including fooling images. They showed the network was fairly good at detecting fooling images. We replicate and expand on their experiments, adapting and running it on the MNIST dataset with a formal experiment on detection or correction rates of adversarial and fooling images.

To implement OSDN on MNIST, the mean activation vectors of the first fully-connected layer (ip1) is computed from the correctly classified training images of each of the 10 classes, and the cosine distances of all training set activation vectors to their corresponding MAV's is computed. Weibull probability distributions are estimated on these distances, which allows us to estimate the probability of exclusion for any input image. For a test image, the activation vector is computed, compared to all MAV's, and probably of it being unknown is computed via the Weibull-model for each class. These exclusion probabilities are then merged with the original decision of the network via the openmax algorithm. The result is 11 probabilities, one for each class plus one for the input being unknown. Note that these results rely on some parameters of the OSDN, such as the *tailsize* and α, which need to be selected appropriately. For more details, please refer to [1].

Tab. 3 presents results using the OSDN to classify MNIST examples, with columns for closed set accuracy after openmax processing, estimated probably of the input to be

Data	**Accuracy**	**Unknown**	**Same Target**	**Other Label**
Test Set	98.07 %	0.63 %	—	1.30 %
FGS Adversarial	91.11 %	1.82 %	5.76 %	1.31 %
FGV Adversarial	99.29 %	0.30 %	0.00 %	0.40 %
HC-1 Adversarial	98.90 %	0.60 %	0.10 %	0.40 %
Fooling 50 %	—	69.06 %	18.89 %	12.06 %
Fooling 90 %	—	69.28 %	26.96 %	3.75 %
Fooling 99 %	—	15.46 %	83.12 %	1.42 %

Table 3: Open Set Network. *Images are rejected to be unknown by the open set deep network. The first row shows classification accuracy and rejection rate on the original test data. Below, rejection rates for adversarial images of different types are presented, including the number of adversarial images that are still classified as the targeted class. Note that adversarial images are created only for those test images that were correctly classified by the original LeNet. Finally, rejection rates and targeted class rates for fooling images of different strength are provided.*

unknown, frequency of the image staying in its target class, and the frequency in which it ended up in another incorrect class. The first row of the table shows that the classification accuracy on the original images dropped moderately, i.e., from 99.09 % to 98.07 %, but not dramatically. The next three rows present the results on the different types of adversarial images, which were created from the images that were classified correctly by the original network. For FGV and HC-1 adversarial images, almost all of these images are *corrected*, i.e., they are now classified as the original class. Below 1 % of the adversarial examples are rejected to be unknown, and almost no image was classified as the same (wrong) class as it was before – and that it was aimed at in case of HC-1. FGS adversarial images seem to be a little bit harder to detect, but still 91 % of them are classified as the originating class by the OSDN, 2 % are classified to be unknown and 6 % survived at the same class. The remaining images changed the class label to another class, which is neither the adversarial nor the originating class.

To test the ability of the OSDN to reject fooling images, we created such images using an adapted version of the source code provided by [9], which is provided in our source

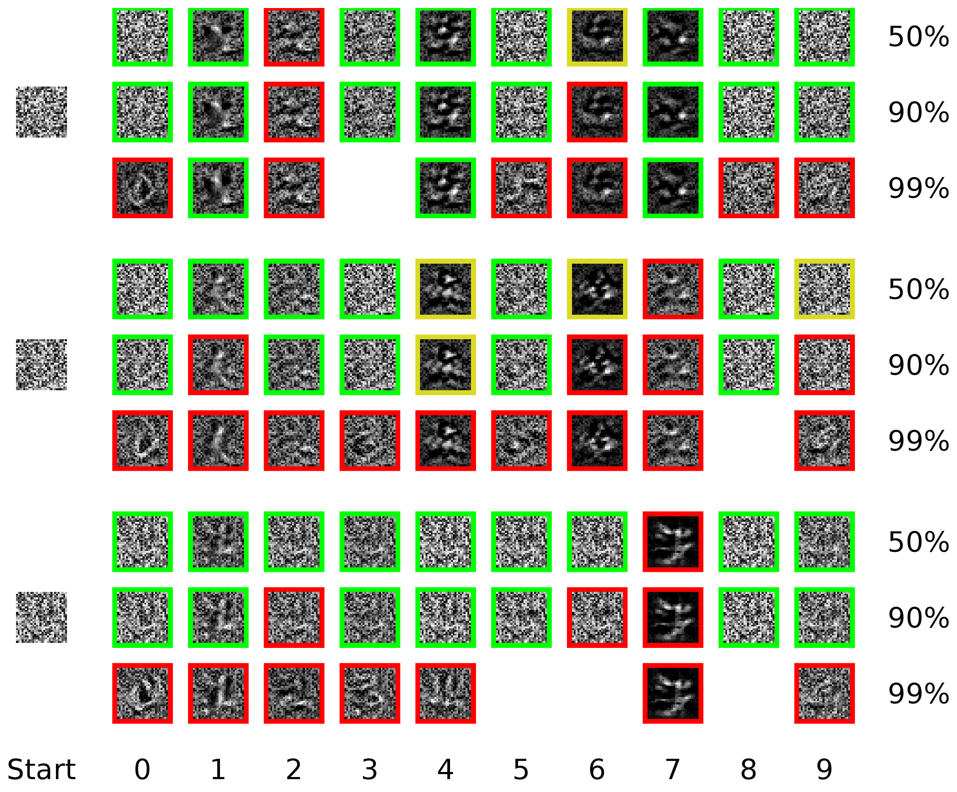


Figure 4: Fooling Images from Random Initialization. *Fooling images are created for each of the 10 digits for three values of classification probabilities: 50 %, 90 % and 99 %. For each of three random starting images, all fooling images are presented. Images with green border are classified unknown, red border images are accepted as the target class, and yellow border images are accepted for another class. For missing images, the generation of the fooling image did not succeed.*

code. Starting from random images, we created 100 fooling images for each digit, in several classification probabilities, i.e., F50, F90, and F99. Examples are shown in Fig. 4. For F50 and F90, most of the images do not represent the targeted class very well, while for F99 fooling images, several of the images seem to represent instances of the targeted classes (with a little bit of imagination).

The response of the OSDN on fooling images is in the last three rows of Tab. 3. For the F50 and F90 fooling images, OSDN was able to reject around 70 % of the images as unknown. For F50, 12 % of the images were assigned a different label, and only 19 % still obtained the targeted label. Looking at the examples in Fig. 4, we can see that most of the images that have contents that could be interpreted as the according digit are classified by the network to contain that digit, indicated by a red border. Few of the images, those with yellow border, are accepted by the network, but with a different label as the one used to produce the fooling image. Images with a green border are detected by the OSDN to be unknown, which is what we want to happen.

Here, we report results for a tailsize of 200 and *alpha* of 2, which provides a reasonable trade-off between accuracy on the raw images and correction or rejection rate for adversarial and fooling images, but we note that a wide range of performance can be achieved depending on the desired accuracy or robustness. While running the OSDN experiments, we found that the tailsize influences the classification accuracy in the clean test set and the classification accuracy (i.e., the correction) of adversarial images. Larger tailsizes will results in better accuracy on the clean test set, and smaller tailsizes increase the adversarial image correction. When using a tailsize of 1000, for example, the classification accuracy on the test images drops to 93 %, but it removes all FGS, FGV and HC-1 adversarials from reaching the target, and rejects 41 % of the F99 fooling images as unknown, and 88 % of the F50 and F90 fooling images. A tailsize of 20, on the other hand, keeps the classification accuracy of clean samples as high as 99 %, but leaves 51 % of the FGS and 11 % of the FGV and HC-1 adversarial images to be classified as the targeted class, as well as 83 % of the F90 fooling images and 97 % of the F99 fooling images. A more detailed analysis of the parameters is left as exercise for the readers, e.g., using the provided code.

One difference between our experiments and the work of Bendale and Boult [1] is, which layer of the network was used. While we used a deeper layer (ip1) in the network, Bendale and Boult performed experiments with the last network layer (FC8 on AlexNet). However, in unreported results, other layers (FC7 of AlexNet) performed similarly. We also tested OSDN on LeNet by computing the MAV from layer ip2. While it did very well on adversarial images, it was not as accurate on fooling images. The much lower dimensionality of ip2 is likely why using that layer provides much weaker performance for MNIST fooling images. Bendale and Boult [1] worked with features of 1000 classes; it is interesting to see that the

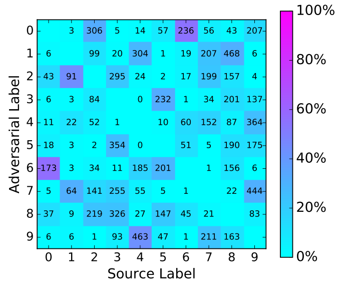
OSDN model transitioned quite well on MNIST when using ip1.

3 Can Adversarial Images be explained by Aliasing?

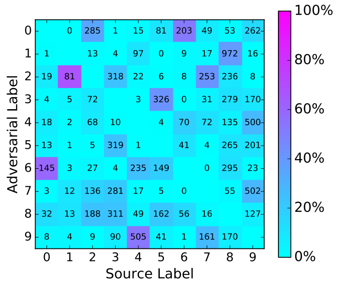
So far, we showed techniques to help manage adversarial images. However, we still do not know, why adversarial images exist. Adversarial perturbations tend to add high frequency information to the image, leading to a misclassification. Furthermore, DNNs perform some kind of spatial resolution reduction. Looking at this from a signal processing point of view, it seems obvious that aliasing could impact networks and that aliased information might lead to misclassification. As the Niquist-Shannon sampling theorem [10, 15] shows, instead of being ignored, high-frequency information above the sampling rate folds over into low-frequency information. While not presented here, examining the FFT of MNIST network data at various layers shows significant high-frequency information within each layer of the network. Hence, the (imperceptible) high-frequency information that is introduced by adversarial images could naturally be turned into low-frequency information, i.e., it would be aliased. Could this be the root cause of adversarial images?

To investigate if the aliasing effect causes adversarial images to be so successful, we ran experiments using LeNet. In LeNet, the two convolutional layers (conv1 and conv2) are each followed by a max-pooling layer, which performs sampling by taking taking the maximum of a 2×2 pixel region. To make it impossible for the network to be affected by aliasing, before each of the pooling layers we added another convolutional layer with fixed (non-learnable) weights. These convolution layers perform a Gaussian blurring of the output of the previous (original) convolution layer, with a Gaussian standard deviation σ set to 1.25, so that basically no high-frequency information survived.

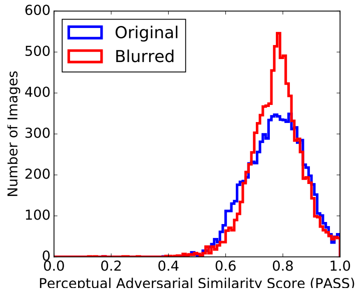
We trained both the LeNet with these additional blurring layers for anti-aliasing, and the original LeNet on the MNIST training set for 50k iterations using the same training strategy. Interestingly, classification accuracies between the original network (99.17 %) and the blurred network (98.84 %) differ only slightly. Then, we tried to generate FGV adver-



(a) Original LeNet



(b) Anti-Aliased LeNet



(c) PASS distribution

Figure 5: ADVERSARIAL LABELS OF LENET WITH AND WITHOUT ALIASING. *FGV adversarial image labels were generated with (a) the original LeNet and (b) the LeNet trained with blurring to reduce aliasing. Colors in the confusion matrices indicate the relative number of labels, while exact numbers are given in the cells. (c) contains a histogram of PASS values for both networks.*

sarials of the 10000 MNIST test images with both networks. If aliasing causes adversarial images, we would expect to see dramatic reduction in the number of adversarial images that we could generate, or in the PASS quality of them. Surprisingly, the number of adversarial images that we could generate using the anti-aliased network (9483) is even higher than using the original network (8811). Also, as shown in Fig. 5, the distribution of PASS values is very similar, and even the distribution of adversarial labels, i.e., the labels that the adversarial images obtained, is very comparable.

Note that, in unpublished work, we have found similar results on more difficult datasets with deeper network structures. This conclusively shows that aliasing is *not* the cause of adversarial images, and puzzling anti-aliasing may actually increase susceptibility to adversarial generation. While our experiments relatively convincingly show that aliasing is not the cause of adversarial images, since networks repeatedly downsample without filtering, the experiments raise their own interesting questions about the robustness of deep networks – "why does aliasing not seem to impact deep network performance?"

4 What the Results Tell us

As we have shown in our experiments, adversarial and fooling images do not present such a big threat as people might think. Even simple image preprocessing techniques such as blurring, shifting or adding noise removes a good portion of the adversarial images. As Graese *et al.* [4] showed, these small perturbations will automatically occur in a normal image acquisition process, i.e., when an adversary uses a printed adversarial image to attack. Furthermore, a technique that is applied to most of the classification networks to improve classification accuracy, namely extracting different crops of the image, was able to eradicate 90 % of the adversarial images. In our example, we used only five different crops, which were only one pixel apart from each other. Other network architectures use many more, e.g., [17, 18, 16] use 144 different crops of the image. Extrapolating from our experiments, it is very unlikely that any adversarial image that was generated using only the center crop would survive averaging 144 crops. However, in the future people might be able to generate adversarial images that are based on several crops of the image, so a simple cropping might not help against those images.

Another technique that we have used is the Open Set Deep Network (OSDN) [1], which inherently is designed to reject images that are of classes that the network did not train on. Although Bendale and Boult [1] claimed that the OSDN could easily detect adversarial and fooling images, they did not perform a statistical analysis. Here, we showed that the OSDN corrects most of the adversarial images, i.e., they are classified as the original class, and are, hence, not adversarial anymore. Also, many of the fooling images – at least the ones which would be difficult to classify by humans – are detected as unknown. We have to note that we only used difficult adversarial images, with a minimal perturbation for which the adversarial class is predicted with only a slightly higher probability than the original class. Hence, the OSDN was able to correct these weights easily. When creating adversarial images with larger perturbations it would increase the probability of the adversarial class, e.g., a minimum probability of 50 %, many of these images might not be detected or corrected by the OSDN. However, the quality of the adversarial images will also get worse, e.g., in terms of PASS values, and it will be more visible that these images are modified. Its is also likely that detectors can be trained for such adversarial images [7] and it would be easy to extend the OSDN to include such a detector. Also, we have performed experiments only for three different types of adversarial image generation techniques; other techniques might generate much better adversarial images that are harder to detect or correct by the OSDN.

In this tutorial, we have run experiments only on a small-scale dataset and a small-scale network. The main purpose of this tutorial is to provide an easy entrance into A) generating adversarial and fooling images, B) implementing easy image processing algorithms to fight adversarial images, and C) using the open set deep network addition to normal classification networks to correct adversarial and detect fooling images. For this purpose, we provide the source code to rerun our experiments, so that researchers have a starting point to develop similar ideas or test them on more difficult databases. With this, we hope to foster more research in the direction of open set classification and improving the robustness of deep networks.

References

[1] A. Bendale and T. E. Boult. Toward open set deep networks. In *The IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1563–1572, 2016. 5, 12, 15, 18

[2] N. Carlini and D. Wagner. Defensive distillation is not robust to adversarial examples. *arXiv preprint arXiv:1607.04311*, 2016. 6

[3] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *Int. Conf. on Learning Representation (ICLR)*, 2015. 1, 2, 4, 5, 6

[4] A. Graese, A. Rozsa, and T. E. Boult. Assessing threat of adversarial examples on deep neural networks. In *IEEE Int. Conf. on Mach. Learn. and App. (ICMLA)*, 2016. 5, 8, 9, 10, 11, 18

[5] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Int. Conf. on Multimedia*. ACM, 2014. 6

[6] A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial examples in the physical world. In *Workshop at Int. Conf. Learning Representatin*, 2017. 11

[7] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff. On Detecting Adversarial Perturbations. In *ICLR 2017*, 2017. 19

[8] 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. 2

[9] A. Nguyen, J. Yosinski, and J. Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2015. http://www.evolvingai.org/fooling. 1, 2, 3, 13

[10] H. Nyquist. Certain topics in telegraph transmission theory. In *American Institute of Electrical Engineers Transactions*, volume 47, pages 617–644, April 1928. 16

[11] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *IEEE Sym. Security and Privacy*, 2016. 6

[12] A. Rozsa, M. Günther, E. M. Rudd, and T. E. Boult. Are facial attributes adversarially robust? In *Int. Conf. on Pattern Recognition (ICPR)*, 2016. 5, 10

[13] A. Rozsa, E. M. Rudd, and T. E. Boult. Adversarial diversity and hard positive generation. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2016. 4, 5, 7

[14] S. Sabour, Y. Cao, F. Faghri, and D. J. Fleet. Adversarial manipulation of deep representations. *Int. Conf. Learning Representations (ICLR) 2016*, 2016. 5

[15] C. Shannon. Communication in the presence of noise. *Proc. of the IRE*, 37(1):10–21, 1949. 16

[16] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Int. Conf. on Learning Representation (ICLR) Workshop*, 2016. 18

[17] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015. 18

[18] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, 2016. 18

[19] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *Int. Conf. on Learning Representations*. Computational and Biological Learning Society, 2014. 1, 2, 3, 6