Adversarial Examples Detection in Deep Networks with Convolutional Filter Statistics

Xin Li, Fuxin Li
School of Electrical Engineering and Computer Science
Oregon State University
urumican@gmail.com, lif@eecs.oregonstate.edu

Abstract

Deep learning has greatly improved visual recognition in recent years. However, recent research has shown that there exist many adversarial examples that can negatively impact the performance of such an architecture. This paper focuses on detecting those adversarial examples by analyzing whether they come from the same distribution as the normal examples. Instead of directly training a deep neural network to detect adversarials, a much simpler approach was proposed based on statistics on outputs from convolutional layers. A cascade classifier was designed to efficiently detect adversarials. Furthermore, trained from one particular adversarial generating mechanism, the resulting classifier can successfully detect adversarials from a completely different mechanism as well. The resulting classifier is non-subdifferentiable, hence creates a difficulty for adversaries to attack by using the gradient of the classifier. After detecting adversarial examples, we show that many of them can be recovered by simply performing a small average filter on the image. Those findings should lead to more insights about the classification mechanisms in deep convolutional neural networks.

1. Introduction

Recent advances in deep learning have greatly improved the capability to recognize visual objects [13, 26, 7]. State-of-the-art neural networks perform better than human on difficult, large-scale image classification tasks. However, an interesting discovery has been that those networks, albeit resistant to overfitting, would have completely failed if some of the pixels in the image were perturbed via an adversarial optimization algorithm [28, 4] . An image indistinguishable from the original for a human observer could lead to significantly different results from a deep network(Fig. 1).

Those adversarial examples are dangerous if a deep network is utilized in any crucial real application, be it autonomous driving, robotics, or any automatic identification (face, iris, speech, etc.). If the result of the network can be hacked at the will of a hacker, wrong authentications and other devastating effects would be unavoidable. Therefore, there are ample reasons to believe that it is important to identify whether an example comes from a normal or an adversarial distribution. A reliable procedure can prevent robots from behaving in undesirable manners because of the false perceptions it made about the environment.

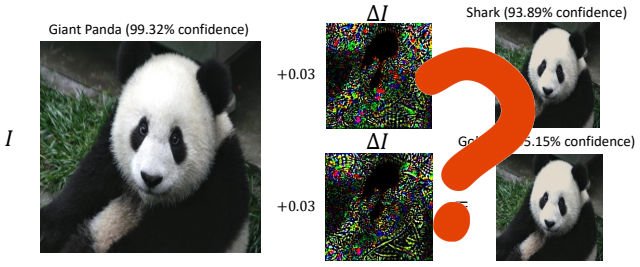


Figure 1. An optimization algorithm finds adversarial examples which, with almost negligible perturbations to human eyes, completely distort the prediction result of a deep neural network [28]. Such algorithms have been found to be universal to different deep networks. This paper studies their properties and seeks a defense.

The understanding of whether an example belongs to the training distribution has deep roots in statistical machine learning. The *i.i.d.* assumption was commonly used in learning theory, so that the testing examples were assumed to be drawn independently from the same distribution of the training examples. This is because machine learning is only good at performing *interpolation*, where some training examples surround a testing example. *Extrapolation* is known to be difficult, since it is extremely difficult to estimate data labels or statistics if the data is extremely different from any known or learned observations. Many current approaches deal with adversarial examples by adding them back to the training set and re-train. However in their experiments, new adversarials can almost always be found from the re-trained classifier. This is because that the space of extrapolation is significantly larger than the area a machine learning algo-

rithm can interpolate, and the ways to find vulnerabilities of a deep learning system are almost endless.

A more conservative approach is to refrain from making a prediction if the system does not feel comfortable about it. Such an approach seeks to build a wall to fence all testing examples in the extrapolation area out of the predictor, and only predict in the small interpolation area. Work such as [16] provides basic theoretical frameworks of classification with an abstain option.

Although these concepts are well-known, the difficulties lie in the high-dimensional spaces that are routinely used in machine learning and especially deep learning. Is it even possible to define interpolation vs. extrapolation in a $4,000$-dimensional or $40,000$-dimensional space? It looks like almost everything is extrapolation since the data is inherently sparse in such a high-dimensional space [9, 6], a phenomenon well-known as the curse of dimensionality. The enforcement of the *i.i.d.* assumption seems impossible in such a high-dimensional space, because the inverse problem of estimating the joint distribution requires an exponential number of examples to be solved efficiently. Some recent work on generative adversarial networks proposes using a deep network to train this discriminative classifier [3, 22], where a generative approach is required to generate those samples, but it is largely confined to unsupervised settings and may not be applicable for every domain convolutional networks (CNNs) have been applied to.

In this work we propose a discriminative approach to identify adversarial examples, which trains on simple features and can approach good accuracy with limited training examples. The main difference between our approach and previous outlier detection/adversarial detection algorithms (e.g. [2]) is that their approaches usually treat deep learning as a black box and only works at the final output layer, while we believe that the learned filters in the intermediate layers efficiently reduce the dimensionality and are useful for detecting adversarial examples. We make a number of empirical visualizations that show how the adversarial examples change the prediction of a deep network. From those intuitions, we extract simple statistics from convolutional filter outputs of various layers in the CNN. A cascade classifier is proposed that utilizes features from many layers to discriminate between normal and adversarial examples.

Experiments show that our features from convolutional filter output statistics can separate between normal and adversarial examples very well. Trained with one particular adversarial generation method, it is robust enough to generalize to adversarials produced from another generation approach [20] without any special adaptation or additional training. Those confidence estimates may improve the safety of applying these deep networks, and hopefully provide insights for further research on self-aware learning. As a simple extension, the results from visualizations of the features prompted us to perform an average filter on corrupted images, and found out that many correct predictions can be recovered from this simple filtering.

2. Deep Convolutional Neural Networks

A deep convolutional neural network consists of many convolutional layers which are connected to spatially/temporally adjacent nodes in the next layer:

$$\mathbf{Z}_{m+1} = [T(\mathbf{W}_1 * \mathbf{Z}_m), T(\mathbf{W}_2 * \mathbf{Z}_m), \ldots, T(\mathbf{W}_k * \mathbf{Z}_m)] \tag{1}$$

where $\mathbf{Z}_m$ is the input features at layer m, $\mathbf{W}_1, \ldots \mathbf{W}_K$ are filters that could be much smaller than the size of $\mathbf{Z}_m$ (e.g. $3 \times 3, 5 \times 5, 7 \times 7$), $*$ is the convolution operator, and T is a nonlinear transformation function such as the rectified linear unit (ReLU) $T(x) = \max(0, x)$. Other commonly used layers in a CNN include max-pooling layers, or other normalization layers [13] such as batch normalization layers [10]. Most deep networks adopt similar principles while adding more structural complexity in the system such as more layers and smaller filters in each layer [26], multi-layered network within each layer [27], residual network [7], etc. A convolutional neural network makes sense in structured data because it naturally exploits the locality structure in data. In an image, pixels that are located close to each other are naturally more correlated than pixels that are far away [17]. The same holds for temporal data (video, speech) where objects (frames, utterances) that are temporally close can be assumed to be more correlated.

3. Understanding the Trained Deep Classifier Under Adversarial Optimization

3.1. Adversarial Optimization

The famous result that deep networks can be broken easily [28] is an important motivation of this work. The idea is to start from an existing example (image) and optimize to obtain an example that will be classified to another category while being close to the original example. Namely, the following optimization problem is solved:

$$\begin{aligned} \min_{r} \quad & c\|r\|_1 + L(f_\theta(\mathbf{x}_0 + r, y)) \\ s.t. \quad & \mathbf{x}_0 + r \in [0, 1]^d \end{aligned} \tag{2}$$

where $\mathbf{x}_0$ is a known example and y is an arbitrary category label, d is the input dimensionality. c is a parameter that can be tuned for trading off between proximity to the original example x_0 and the classification loss on the other category y. It has been shown, to the astound of many, that one can choose an r with very small norm while completely change the output of the algorithm (e.g. Fig. 1), this can even be done universally for almost all networks, datasets and categories [28, 4]. Besides, adversarials trained

from one network may even fool a related one trained from the same dataset [18]. This has led many people to question whether deep networks are really learning the "proper" rules for classifying those images.

3.2. Adversarial Behavior

In order to gain a deeper understanding of the behavior of a deep network and illustrate the difference between adversarial and normal example distributions, we utilize spectral analysis. As a starting point, we perform principal component analysis (PCA) [11] at the 14-th layer of a VGG network trained on the ImageNet dataset (the first fully-connected layer). The rationale behind using PCA is that each deep learning layer is a nonlinear activation function on a linear transformation, hence a large part of the learning process lies within the linear transformation, for which PCA is a standard tool to analyze.

A linear PCA is performed on the entire collection of $50,000$ images from the ImageNet validation set, as well as $4,000$ adversarials collected using the approach in (2), starting from random images in the collection. The result shows very interesting findings (Fig. 2) and sheds more light on the internal mechanics of those adversarial examples. In Fig. 2(a), we show the PCA projection onto the first two eigenvectors. This cannot separate normal and adversarial examples, as one could possibly imagine. The adversarial examples seem to exactly belong to the same distribution as normal ones. However, it does seem that the adversarial examples reside mostly in the center while the normal examples occupy a bigger chunk of space.

Interestingly, as we move to the tail of the PCA projection space, the picture starts to change significantly. In Fig. 2(b), we can see that there are a significant amount of adversarial examples that has extremely large values w.r.t. to the normal examples in the tail of the distribution. We chose to print the projection on the $3,547$-th and $3,844$-th eigenvector, but similar distributions can be found all over the tail. As one can see, at such a far end on the tail, the projections of normal examples are very similar to random samples under a Gaussian distribution. An explanation for that could be that under these "uninformative" directions, most of the weighted features are nearly independent w.r.t. each other, hence the distribution of their sum is similar to Gaussian, according to the central limit theorem[1]. However, although normal examples behave similarly to a Gaussian, some adversarial examples are having projections with a deviation as large as 5 or 10 times the standard deviation, which are extremely unlikely to occur under a Gaussian distribution.

[1]Note this is without a ReLU transformation. ReLU would destroy the negative part of the data distribution so that it no longer looks like a Gaussian. However, some tail effects can be observed even in the distribution after ReLU.

Fig. 2(c) and Fig. 2(d) show that there are two distinct phenomena:

- The extremal values and standard deviations on the projections onto the first $500 - 700$ eigenvectors are decidedly lower in adversarial examples than in normal ones.
- The extremal values and standard deviations on the projections onto the last $1,000 - 1,500$ eigenvectors are decidedly higher in the adversarial examples than the normal ones.

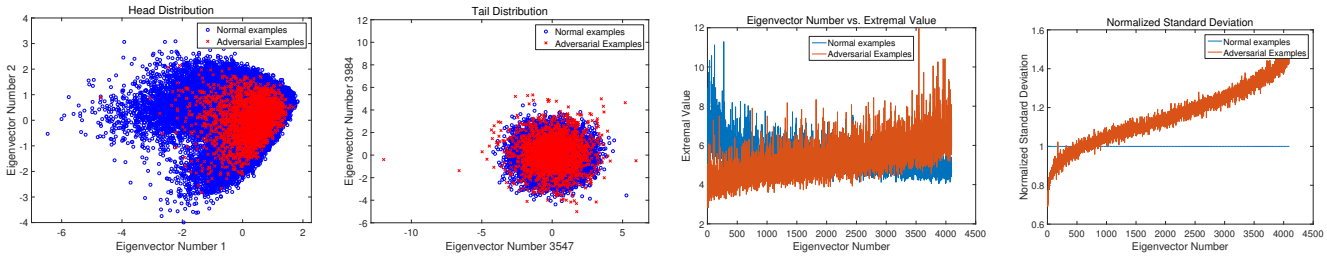
It is interesting to reflect about the causes and consequences of those properties. One deciding property is that there is a strong regularization effect in adversarial examples on almost all the informative directions. Hence, the predictions in adversarial examples are *lower* than those in normal examples, rather than the confidence values may have indicated (Fig. 1). In Fig. 3, we show the number of categories with a prediction higher than a threshold, before the final softmax transformation

$$p_i(x) = \frac{\exp(f_i(x))}{\sum_i \exp(f_i(x))} \tag{3}$$

that converts raw predictions $f_i(x)$ into probabilities. The result shows that normal examples have on average one category with a raw prediction value more than 20, however adversarial examples have only 0.01 category with raw predictions more than 20. The reason that those adversarial examples appear more *confident* after softmax is because that the predictions on all the other categories are regularized *even more*. Hence the normalization component of softmax has decided that the single prediction, although much less strong, should be assigned a probability of more than 90%. We note that this issue was also pointed out by [2] in a different manner and they proposed a solution in the OpenMax classifier, which we compare against in the experiments.

But besides that, it seems that such extremal and standard deviation statistics are evident features that could help discriminating normal and adversarial examples. Unfortunately, they only occur as a statistic from a large sample, as any single point in Fig. 2(a) looks similar to a single point in the normal distribution. We have tried to utilize the tail distributions (Fig. 2(b)) to create a classifier which easily achieved 99% accuracy separating adversarials from normals, however we subsequently found out that since the tail almost do not contribute to the classification, knowing this defense, the adversarial example can easily optimize to remove their footprints on the tail distributions.

This leads us to think about an approach that would turn a single image into a distribution, so that we can use statistics as detectors for adversarial examples. An image is a distribution of pixels. Especially, the output of each filter from each convolutional layer is an image which could be



(a) (b) (c) (d)

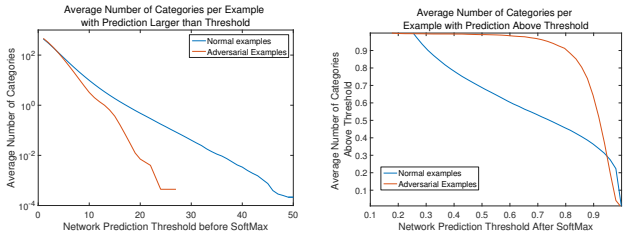
Figure 2. Blue indicates normal examples and red/orange indicate adversarial examples. Projections are normalized by dividing the standard deviation of all normal examples projected on the corresponding dimension. (a) The projection of the data at layer 14 onto the 2 most prominent directions; Adversarial example cannot be identified from normal ones. (b) Projection of the same data to the $3,547$-th and $3,844$-th PCA projections, some adversarial examples are having significantly higher deviation to the mean; (c) The absolute normalized extremal value in the projection to each eigenvector; (d) The average normalized standard deviation of normal and adversarial examples on each projection. Standard deviations of normal examples stand at 1 because of the normalization.

treated as a distribution where the samples are the pixels. Therefore, in the following section we aim to build a classifier based on collecting statistics from such distributions.

4. Identifying Adversarial Examples

4.1. Feature Collection

Suppose the output at a convolutional layer m is an $W \times H \times K$ tensor, where W and H represent the width and height of the image at that stage (smaller than original after max-pooling), and K represents the number of convolutional filters. Such a tensor can be considered as a K-channel image where each pixel has a K-dimensional feature. We consider the feature on every pixel to be a random vector drawn from the distribution D_m of convolutional pixel outputs, a K-dimensional distribution.



(a) (b)

Figure 3. Average number of categories per example with predictions higher than a threshold. (a) Before softmax; (b) After softmax. As one can see, in normal examples, there are on average about 1 category with a prediction score of more than 20 (before softmax), while with adversarial ones, only 1% examples have a category with a prediction score more than 20. However, since prediction values on all categories have dropped, after softmax adversarial examples obtain much higher likelihood on one category.

The list of statistics we collect is:

- Normalized PCA coefficients
- Minimal and Maximal values
- 25-th, 50-th and 75-th percentile values

on each of the K-dimensional features. Normalized PCA coefficients are collected via Algorithm 1. Extremal and percentile statistics are straightforward to understand.

The features we collect are non-subdifferentiable, hence essentially preventing adversaries to use gradient-based attacks to counter the classifier. Although we are interested in a generative adversarial network-type adversary which would learn to avoid our detector, such adversaries would have to resort to derivative-free optimization methods, which currently do not scale to the size of a realistic image. The best derivative-free approach we have tried scales up to several hundreds of variables. The genetic algorithm in [20] scales better, but as we will soon show, their low-level feature statistics are so different from natural images, making them very easy to be detected, even without training on any data from their adversarial generation algorithm.

Algorithm 1 PCA Statistics Extraction

1: **INPUT:** Image I, layer m.
2: For all normal images in a training set, compute their CNN filter output of layer m to form an example matrix $\mathbf{Z_m}$.
3: Compute the mean $\mathbf{e}$ and PCA projection matrix $\mathbf{W}$ of $\mathbf{Z_m}$.
4: Compute the standard deviation $\mathbf{s}$ on each dimension in the PCA projection $\mathbf{W}^\top(\mathbf{Z_m} - \mathbf{e1}^\top)$.
5: For each image I, project its CNN filter output of layer m $\mathbf{Z_{mI}}$ using PCA: $\mathbf{z_{mI}} = \mathbf{W}^\top(\mathbf{Z_{mI}} - \mathbf{e1}^\top)$, and normalize them by dividing the standard deviation s on each respective dimension.
6: Collect the statistic for each image as $\mathbf{x}_I = \frac{1}{n}\|\mathbf{z_{mI}}\|_1$, where L_1 norm is the vector L_1 norm. The resulting statistic is K-dimensional.

4.2. Classifier Cascade

[29] proposed a famous strategy for face detection by using a cascaded boosting classifier composed by a sequence

of base classifiers. A cascade classifier is ideal when it is easy to identify many of the examples from a category but some important cases can be difficult. In Fig. 4, SC_N represents the classifier at each stage. X is the input of the cascade classifier. The negatives in a cascade classifier from each stage will be outputted directly, while the positives will go to the next stage.

In our case, the normal category is much easier to detect than the adversarial category (see e.g. Fig. 6). In our initial experiments with VGGNet, we found that more than 80% of normal examples can be determined from the first convolutional layer with 100% precision. Therefore, we constructed a cascade classifier based on convolutional layers: the first stage works with features collected from the outputs of the first convolutional layer, the second with the second layer, etc. (Fig. 4). The base classifiers will not solely consider statistics from their own stage, instead, after one stage of training, the remaining positive examples will be concatenated to the corresponding features on the next stage.

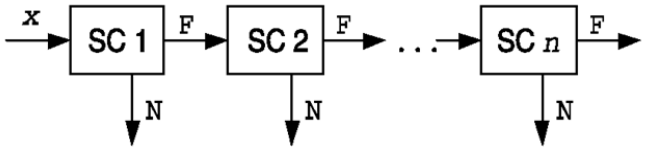


Figure 4. A cascade classifier is defined on each of the convolutional layers in a convolutional network (SC i represents the i-th convolutional layer)

The operations that are represented by Fig. 4 can also be summarized as Algorithm 2.

Algorithm 2 Training Process of a cascade of Classifier

1: $N_{pool} \leftarrow$ Normal example pool, $P_{train} \leftarrow$ Training set of N_p perturbed examples, $L \leftarrow$ Number Of convolutional layers
2: **while** current layer $\leq L$ **Or** $N_{pool} \neq \emptyset$ **do**
3: Draw N_p sized subset P_{normal} from N_{pool}
4: $T \leftarrow P_{normal} \cup P_{train}$
5: Train SVM on T
6: Predict SVM on N_{pool}, eliminate those predicted as normal above a threshold (described in text)
7: **end while**

The overall false positive rate of a K stage cascade classifier can be represented as: $F = \prod_{i=1}^{K} f_i$, where f_i is the false positive rate at each layer. And similarly the true positive rate can be represented in the same form: $T = \prod_{i=1}^{K} t_i$ where t_i is the true positive rate at each stage. In order to maximize recall, we maintain a high true positive rate and select a classification threshold which corresponds to a high true positive rate (97% in AlexNet and 98% in VGG).

5. Related Work

Szegedy et al. [28] proposes the adversarial optimization formulation in eq. (2). [4] proposes an explanation of the adversarial mechanism, and proposed a simpler adversarial optimization mechanism that only corrupts based on the signs of gradient of the network. The fact that such examples can be generated so easily with the gradient sign method shows that adversarial examples come from attacking the magnifying effect coming from the linearities in the network. [20] proposes another mechanism to generate adversarials using evolutionary optimization. The result of these do not resemble natural images but still can be classified by deep networks with high confidence(Fig. 5). [19] proposes another efficient approach. [23] proposes an approach to generate adversarials that match the convolutional filter outputs as well as perturbing the data. [25, 8] propose approaches to sample adversaries or minimax optimization for making learning more robust. While most of the work are done on standard benchmarks such as MNIST, CIFAR and ImageNet, [14] is an interesting work on projecting the adversaries in physical world.

Recently, there have been a lot of focus on training adversarial generation networks to create Generative Adversarial Networks (GANs) [3, 22, 32, 24]. These networks play a two-player game where a generator network aims to generate adversarials that will not be correctly classified by another discriminator network, and the goal is to generate images more and more similar to natural images. It has been shown that these networks generate images that resemble natural images. However, this generative approach is different from our goal, where we aim to create discriminative networks that discriminates from images that are already indistinguishable from natural images (e.g. Fig.1).

Mechanisms for detecting and countering adversarial examples have also been proposed [5, 21]. [18] proposes to use the foveation mechanism to alleviate adversarial examples when it is already known to be adversarial, but did not attempt to detect adversarials. The open-set deep networks proposed by [2] seek to alleviate concerns from a soft-max classification by creating an abstain option. The universum classifier [31] is similar but with more theoretical arguments.

Self-aware learning (classification with an abstain option) had been proposed in e.g. [12, 16]. It is relevant to robust learning (e.g. [15]), however robust learning usually seek to directly optimize the minimax loss under adversarial conditions, instead of outputting an abstain option. [1, 30] also focuses on classification with an abstain option.

6. Experiments

Our algorithm is tested on 2 approaches to generate adversaries. The main one is data generated using the L-BFGS

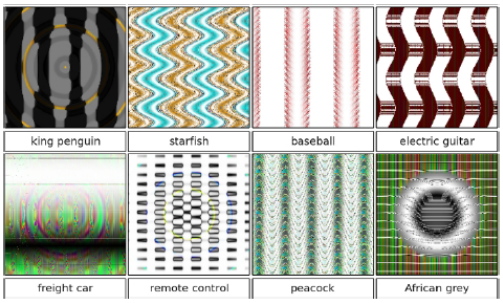


Figure 5. Evolved Adversarial Images Unrecognizable to Humans

algorithm by [28]. We generated $4,000$ adversarials from a random subset of the ILSVRC-2012 validation set (total of $50,000$ images). In order to test the out-of-sample generalization capability, we included another dataset, which includes $5,000$ EA-adversarial images generated using the algorithm in [20]. These datasets are tested with 2 different network structures, VGG-16 [26] and AlexNet [13]. All input images had been normalized and re-shaped into 224×224 for VGG-16 model and 227×227 for AlexNet. The MatConvNet toolbox is used for the deep networks. Features are extracted from the statistics of interest, mentioned in section 4.1, in the lower 9 convolutional layers from the VGG-16 network, and in all 5 convolutional layers from AlexNet.

All the classifiers were trained with a subset of $2,000$ L-BFGS adversarials and a randomly selected subset of $45,000$ normal images in the ILSVRC-2012 dataset. Testing was done on a random held-out dataset with the other $5,000$ normal images and $2,000$ L-BFGS adversarials for the L-BFGS experiments, and on a randomly selected $5,000$ normal images and $5,000$ EA-adversarials for the EA experiments. The training and testing sets do not overlap in any means: for all the testing LBFGS adversarials, the corresponding normal images were not seen in the training set. EA experiments are only conducted with AlexNet since we found out that EA-adversarials did not succeed fooling the VGG-16 network. We have also tried some other fast adversarial generation approaches such as [4], but they generally do not generate adversarials that fool classifiers as completely (with more than 90% confidence) or as universally (on 100% of the images) as the L-BFGS algorithm.

Additional experiment results using DeepFool and on ResNet can be found in the supplementary material.

6.1. Single-Layer Results

The first experiments we did were by extracting statistics on each single convolutional layer output, instead of using the cascade classifier. From Tables 1 and 2, one can see that the statistics from a single layer have some discriminative power to distinguish normal examples from adversarials, but are not extremely effective. However, EA-adversarials were much easier to distinguish, even though our classifier was trained only on L-BFGS adversarials instead of EA ones (Table 3). We only need the first three convolutional layers to reach an overall 97.34% classification accuracy. We believe the reason is that our features capture natural image statistics, and because EA-adversarials look so unnatural, their statistics are vastly different than natural images (see Sec. 6.3 for more discussions).

Table 1. Classification Result with AlexNet for Normal vs. LBFGS-adversarials

Network Layer	2nd	3rd	4th
Accuracy	57.5 ± 0.7	67.3 ± 0.7	70.9 ± 0.6
Network Layer	5th	6th	
Accuracy	74.9 ± 0.9	78.95 ± 0.6	

Table 2. Classification Result with VGG-16 for Normal vs. LBFGS-Adersarials

Network Layer	2nd	3rd	4th
Accuracy	72.1 ± 0.7	84.1 ± 0.7	80.3 ± 0.6
Network Layer	5th	6th	7th
Accuracy	81.4 ± 0.9	74.3 ± 0.6	73.9 ± 0.6
Network Layer	8th	9th	10th
Accuracy	74.2 ± 0.7	71.2 ± 0.7	74.3 ± 0.8

Table 3. Classification Result for Normal vs. EA-Adversarials

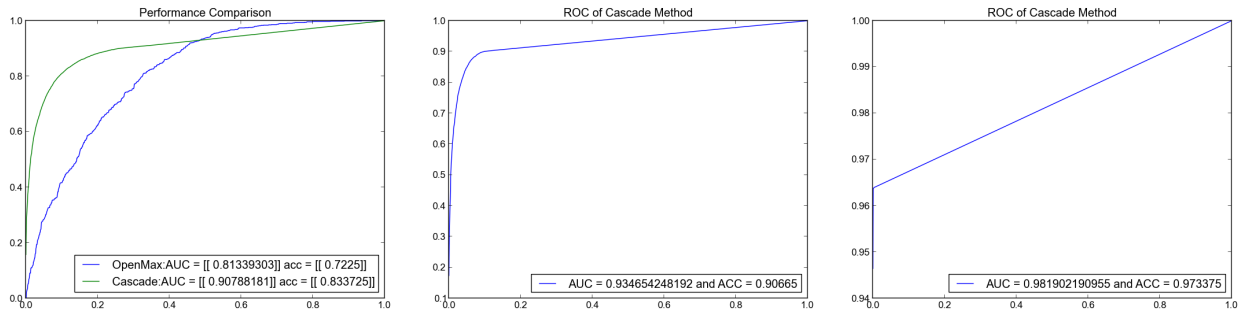
Layer	2nd	3rd	4th
Accuracy	93.45 ± 0.69	98.3 ± 0.73	97.9 ± 0.57

6.2. Experiment for LBFGS-Adversarials Detection

Next we test the cascade classifier on both AlexNet and VGG-16. The parameter C is set to 0.005. On AlexNet, the average accuracy of the cascade classifier reaches 83.4% over 20 random trials, and the AUC (area-under-curve) metric is 90.7%. We compared against the recently published OpenMax method [2]. To learn the Weibull distribution required for OpenMax, the EVT was applied on the same training set as the algorithm. Figure 6(a) shows the results, where we were able to outperform OpenMax by over 9% in area-under-curve (AUC) and 11% in terms of accuracy.

In VGG-16, the results were even better. The accuracy of the classifier was on average 90.665% over 20 random trials. Fig.6(b) shows the ROC curve. We believe the fact that VGG has a lot more layers than AlexNet helps setting more constraints on the layer statistics, and is subsequently helpful for detecting adversarial examples.

Finally, the cascade classifier was tested on EA-adversarials. We obtained more than 96% accuracy with 0 false positive rate, with a final accuracy of 97.3% and AUC of 98.2% (Fig.6(c)). In other words, our algorithm is rarely fooled by EA-adversarials, even without training on them.



(a) (b) (c)

Figure 6. (a) Comparison Between OpenMax detection Methods and Cascade Classifier: The blue curve represents the performace of OpenMax Method, and green curve represents the perfornace for Cascade Classifier.(b) Overall ROC Performance Curve of Cascade Classifier Trained on VGG-16 Network. (c) Overall ROC of data generated from EA-adversarials dataset on AlexNet.

6.3. Visualization of Statistics

Our experiment results show that EA-adversarials are easy to detect with our detector. To gain more insight into this result, we made a few comparisons between the statistics of interest extracted from normal images, LBFGS-adversarials and EA-adversarials.

We visualized the average of the statistics that are used for the detection task from the first layer of the AlexNet on all its dimensions. As can be seen in Fig.7(a), the difference on the PCA projection statistics on extracted from EA-adversarials and that of the normal images is very dramatic. Meanwhile, compared to the EA-adversarials, the statistics from LBFGS-adversarial have much less difference from the normal data and the difference does not change very much across different dimensions.

From Fig. 7(b), one can see that LBFGS-adversarials have smaller extremal values than normal images. This might imply that the LBFGS optimization worked to diminish strong signals from the original image by introducing small pixel perturbations, and that helped our classifiers separating them from normal images. From Fig. 7(c), we see the EA-adversarials evidently differ from normal images. Those results illustrate why EA-adversarials are easier to detect. We suspect it would be easy to reach 100% accuracy, had we actually trained on some EA-adversarials. The capability to generalize to EA-adversarials without training on them showed the general capability of our cascade classifiers to capture natural image statistics and distinguish natural images from unnatural ones.

7. Discussions

7.1. Self-Aware Learning with an Abstain Option

The framework of self-aware learning [16, 2, 31] considers the case where the learning algorithm has an abstain option of saying "I don't know", instead of always making an actual prediction. We define a framework that is slightly different than [16], avoiding the requirement in some frameworks of never making a mistake.

We assume that the training input is drawn i.i.d. from a distribution $P(\mathbf{x}, y)$, where $\mathbf{x}$ is the input and y is the output. Assume that the testing input is drawn from a mixture distribution between $P(\mathbf{x}, y)$ and $Q(\mathbf{x}, y)$:

$$P_m = \Omega P(\mathbf{x}, y) + (1 - \Omega) Q(\mathbf{x}, y) \tag{4}$$

, where $\Omega \in \{0, 1\}$ is an unknown mixture weight, and $Q(\mathbf{x}, y)$ is an adversarial distribution. Assume that we have a classifier that includes a function $f(\mathbf{x})$, and a boolean strategy a_i between `predict` and `abstain` that can be chosen for each individual $\mathbf{x}_i$. Assume that the expected error from our classifier on the adversarial distribution is e_q (which could be assumed, if no other prior is present, as the random guessing error of $\frac{C-1}{C}$ for a C-class classification problem). Further assume that abstaining always incur a fixed cost e_a. As long as $e_a < e_q$, abstaining would be better than predicting on the example drawn from the adversarial distribution, however, e_a should be set sufficiently large so that the classifier would still make predictions when confident, instead of abstaining everything.

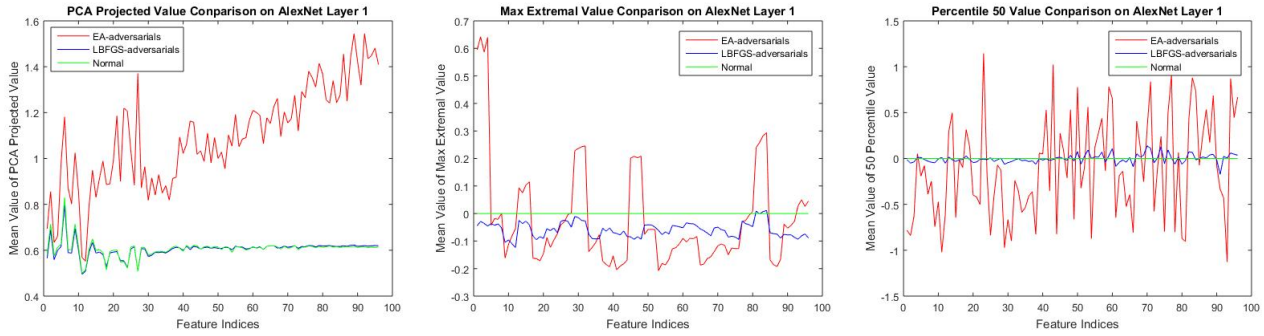
For each testing input, the testing of the self-aware classifier is then trying to optimize $\min_a E_{P_m} L_a(\mathbf{x}, y)$ where

$$L_a(\mathbf{x}_i, y_i) = \begin{cases} P(y_i \neq f(\mathbf{x}_i)), & \text{if } a_i = \texttt{predict}, \\ & (\mathbf{x}_i, y_i) \sim P(\mathbf{x}, y) \\ e_q & \text{if } a_i = \texttt{predict} \\ & , (\mathbf{x}_i, y_i) \sim Q(\mathbf{x}, y) \\ e_a & \text{if } a_i = \texttt{abstain} \end{cases} \tag{5}$$

hence the classifier needs to select between making a prediction using its function $f(\mathbf{x})$ and risk paying e_q versus abstaining. It is easy to derive the optimal strategy:

$$a_i = \texttt{predict}, \quad \text{if } P(\Omega = 1|\mathbf{x_i}) P(y_i \neq f(\mathbf{x_i})) + P(\Omega = 0|\mathbf{x_i}) e_q < e_a \tag{6}$$

$$a_i = \texttt{abstain}, \quad \text{otherwise} \tag{7}$$



(a) (b) (c)

Figure 7. (a) PCA Projection Comparison; (b) Maximum Feature Map Extremal Value Comparison; (c) Median Value Comparison

Our approach can be seen as estimating $P(\Omega = 1|x_i)$ in this framework. Experiments about the effect of such self-aware learning is shown in the supplementary material. We eagerly hope to apply it in realistic applications in future work.

7.2. Image Recovery

Insights from [4] indicate that the adversarial mechanism is very specifically attacking vulnerable gradients starting from the first convolutional layer. Insights from the previous experiments also suggest that LBFGS-adversarials work to diminish filter responses from the first convolutional layer. Therefore a natural idea would be to destroy the adversarial effects in the first convolutional layer to try to recover the original image. We tried a very simple approach: applying a small (e.g. 3×3) average filter on the adversarial image before using the CNN to classify it. The positive and negative adverse gradients will average out in this approach, and make the masked activations from the normal images more prominent. In Table 4 we illustrate such recovery results: after using a 3×3 average filter on identified adversarial examples, the classification accuracy improved from almost 0% to 73.0%, showcasing the effectiveness of this simple average filter.

Table 4. Recovery Results. Simply using a 3×3 average filter we can recover a large proportion of adversarial examples after detecting them using the algorithm described previously. More complex cancellation approaches such as foveation in [18] that utilizes cropping can achieve better results.

Approach	Top-5 Accuracy (Recovered Images)
Original Image (Non-corrupted)	86.5%
3×3 Average Filter	73.0%
5×5 Average Filter	68.0%
Foveation (Object Crop MP) [18]	82.6%

Those results show that we can both detect and recover from adversarial examples with high accuracy. But the main reason we performed this (overly simplistic) experiment is to show how simple it might be to cancel out some adversarial perturbations. Importantly, this result indicates that current deep convolutional networks are too locally focused: these are corruptions that can be cancelled out by a simple 3×3 average filter, however they can adversely impact the entire result of the deep network. For human with a large receptive field, they will not even care about what happens within a 3×3 area. Therefore, we believe that future deep learning approaches should focus on enlarging the receptive field in order to reduce the chance of being fooled by adversarial examples. Another potential direction is to research classification approaches that do not require a softmax-type normalization, in order to avoid regularizing attacks such as the ones used in the adversarial optimization in (2).

8. Conclusion

This paper proposes an approach that detects adversarial examples using simple statistics on convolutional layer outputs. A cascade classifier was designed based on simple statistics on filter outputs from each layer. And it was capable of detecting more than 85% of the adversarial examples. Experiments showed that our cascade classifier significantly outperforms state-of-the-art on detecting adversarial examples. Experiment also showed transfer learning capabilities of our classifier, since the classifier we trained with L-BFGS adversarials are capable of detecting EA-adversarials as well. Insights drawn from these experiments lead us to perform simple 3×3 average filter to corrupted images, which successfully recovered most of them. In the future, we would like to explore GAN-type generative adversarial networks from the current results, with multiple rounds of adversarial detection and counter-detection.

Acknowledgements

This paper was supported by Future of Life grants 2015-143880 and 2016-158701.

References

[1] A. Balsubramani. Learning to abstain from binary prediction. *arXiv preprint arXiv:1602.08151*, 2016.

[2] A. Bendale and T. E. Boult. Towards open set deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[3] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

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

[5] S. Gu and L. Rigazio. Towards deep neural network architectures robust to adversarial examples. *arXiv preprint arXiv:1412.5068*, 2014.

[6] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer-Verlag, New York, 2001.

[7] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[8] R. Huang, B. Xu, D. Schuurmans, and C. Szepesvári. Learning with a strong adversary. In *International Conference on Learning Representations*, 2016.

[9] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613, 1998.

[10] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[11] I. Jolliffe. *Principle Component Analysis*. Springer-Verlag, 1986.

[12] R. Kleinberg, A. Niculescu-Mizil, and Y. Sharma. Regret bounds for sleeping experts and bandits. *Machine learning*, 80(2-3):245–272, 2010.

[13] 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.

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

[15] G. R. Lanckriet, L. E. Ghaoui, C. Bhattacharyya, and M. I. Jordan. A robust minimax approach to classification. *Journal of Machine Learning Research*, 3:555–582, 2003.

[16] L. Li, M. L. Littman, T. J. Walsh, and A. L. Strehl. Knows what it knows: a framework for self-aware learning. *Machine learning*, 82(3):399–443, 2011.

[17] X. Li, F. Li, X. Fern, and R. Raich. Filter shaping for convolutional networks. In *International Conference on Learning Representations*, 2017.

[18] Y. Luo, X. Boix, G. Roig, T. A. Poggio, and Q. Zhao. Foveation-based mechanisms alleviate adversarial examples. *arXiv preprint arXiv:1511.06292v3*, 2016.

[19] S. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. Deepfool: a simple and accurate method to fool deep neural networks. *CoRR*, abs/1511.04599, 2015.

[20] A. Nguyen, J. Yosinski, and J. Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[21] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami. Distillation as a defense to adversarial perturbations against deep neural networks. *arXiv preprint arXiv:1511.04508*, 2015.

[22] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[23] S. Sabour, Y. Cao, F. Faghri, and D. J. Fleet. Adversarial manipulation of deep representations. In *International Conference on Learning Representations*, 2016.

[24] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. *arXiv preprint arXiv:1606.03498*, 2016.

[25] U. Shaham, Y. Yamada, and S. Negahban. Understanding adversarial training: Increasing local stability of neural nets through robust optimization. *arXiv preprint arXiv:1511.05432*, 2015.

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

[27] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *arXiv:1409.4842*, 2014.

[28] 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.

[29] P. Viola and M. J. Jones. Robust real-time face detection. *International journal of computer vision*, 57(2):137–154, 2004.

[30] Y. Wiener and R. El-Yaniv. Agnostic selective classification. In *Advances in Neural Information Processing Systems*, pages 1665–1673, 2011.

[31] X. Zhang and Y. LeCun. Universum prescription: Regularization using unlabeled data. In *AAAI Conference on Artificial Intelligence*, 2017.

[32] J. Zhao, M. Mathieu, and Y. LeCun. Energy-based generative adversarial network. *arXiv preprint arXiv:1609.03126*, 2016.

Supplementary materials for ICCV submission #3060

1 Results on DeepFool

For this experiment we used 5000 adversarial images generated with the DeepFool algorithm. We collected a training set of images using the adversarials, with an equal number of real images drawn from the ILSVRC2012 validation set. For the Deep- Fool adversarials we used the implementation given in the Foolbox algorithm library. Given this data, we chose to use the ResNet 50 architecture as the backbone CNN for our experiments. Because we use ImageNet data, we preprocessed our training set accordingly: all images were reshaped to 224x224x3, the channels were modified to BGR ordering, and the channel-wise mean was subtracted from each sample. We then performed a forward pass as usual on all images, performing the classifier cascade at each layer. Given the output of each convolutional layer, we extracted features that would characterize images as being either from a real or adversarial distribution. For each output we extracted PCA coefficients, extremal values, and values within the 25th, 50th and 75th percentile to form a new training sample. We then used an SVM to learn the real statistics from the adversarials. The real examples were given a ground truth label of 1, and the adversarials were given a label of 0. We performed a parameter search over kernel type and C value. In all our experiments a C of 0.005 and a linear kernel performed the best. We then tested on 2000 real images and 2000 DeepFool adversarials. In our experiments with ResNet, we only performed the cascade for the first three layers before we found nearly all of the adversarial images.

The result can be seen in Fig. 1. The algorithm maintained more than 90% AUC, showing that DeepFool did not fundamentally change the type of adversarials.

2 Results on Self-Aware Learning

We implemented self aware learning using ResNet 50, the goal being to use the softmax probabilities to learn parameters that would cause the network to more intelligently classify inputs. The network should classify an image if it was sure the image was real, or abstain if either the network was not sufficiently confident, or if the image was adversarial, as described in Sec. 7.1. To test

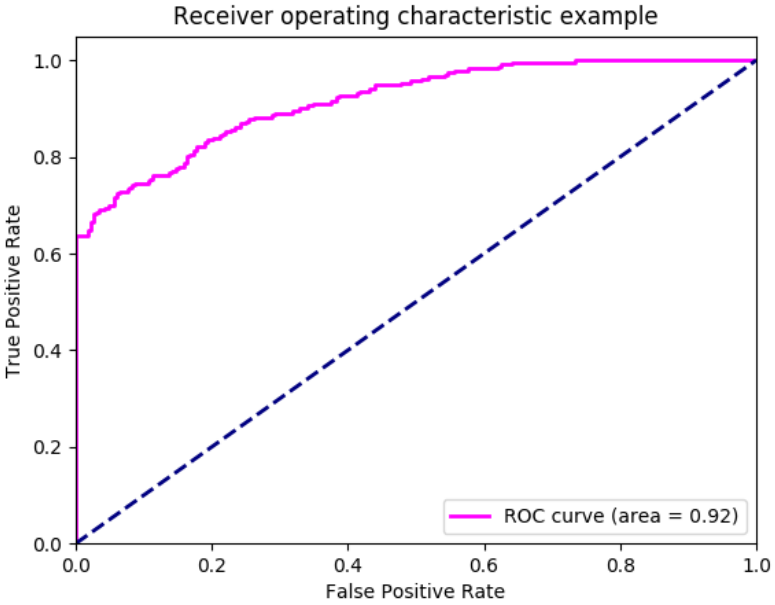


Figure 1: ROC Curve for Detecting DeepFool adversarials in ResNet-50, the algorithm maintained more than 90% in AUC

the presented algorithm, we use $2,000$ real images drawn from ILSVRC2012 validation set, and $2,000$ adversarial images from the testing set of the previous experiment, generated using the DeepFool algorithm. We tested the self aware learning algorithm with a high $e_q = 10$. This worked well enough that the network chose to abstain or classify, rather than incur a high penalty for guessing incorrectly. We observed that for each testing image, our estimation of the source distribution resulted in e_a between 2 and 8. We then varied e_a between these values to see if there was a threshold at which we could abstain from all adversarials, retaining predictions for only real examples. We were also interested in thresholds that maximized the true positive rate (prediction of real examples) while abstaining from as many adversarials as possible. We found the lower thresholds resulted in the abstaining from predicting on all adversarials, but it also abstained from many (but not all) real examples. Higher thresholds resulted in many more real predictions retained, but some also some adversarials made it through. High thresholds would finally result in the network not abstaining at all.

The results can be seen in Fig. 2. It can be seen that besides abstaining adversarial examples, the system also abstains from predicting on some normal examples that the classifier is not confident on. Hence, with a high abstain ratio the prediction accuracy on normal examples is also higher.

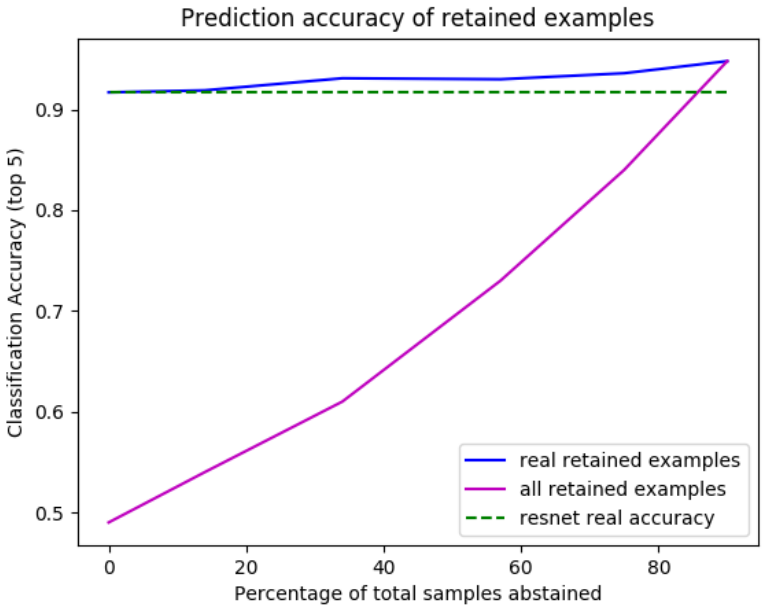


Figure 2: Self-aware learning results. In a mixture of half real and half adversarial examples, the classification accuracy of discarding nothing falls a little under 50%, with more examples abstained, the accuracy improves significantly. The accuracy of retained normal examples (blue curve) also improves when more examples are abstained, as the abstained examples also include normal examples that are not predicted confidently.

3 Images Classified Correctly and Incorrectly

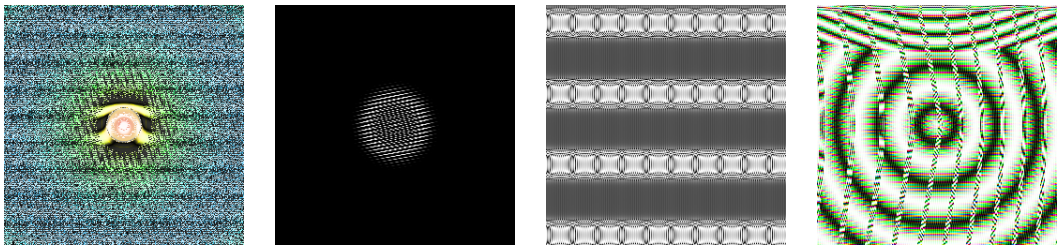
In this section we show some images classified correctly and incorrectly from the algorithm. Unfortunately we are not quite able to observe any particular visible trends, maybe due to the subtlety of adversarial images.

(a) (b) (c) (d)

Figure 3: Some of Misclassification on L-BFGS images by Our Classifier. (a) and (b) are from normal dataset. (c) and (d) are from LBFGS-Adversarial dataset, which is misclassified to category n02408429(water buffalo) and n01518878(ostrich, Struthio camelus).

(a) (b) (c) (d)

Figure 4: Some of Correctly Classified on L-BFGS images by Our Classifier. (a) and (b) are from normal dataset. (c) and (d) are from LBFGS-Adversarial dataset, which is misclassified to category n04209133(shower cap) and n02328150(Angora).



(a) (b) (c) (d)

Figure 5: Some of Misclassfied EA images by Our Classifier. From left to right, they are misclssified to category n03220513 (dome), n01749939 (green mamba), n04118776 (rule, ruler) and n03935335 piggy (bank, penny bank)

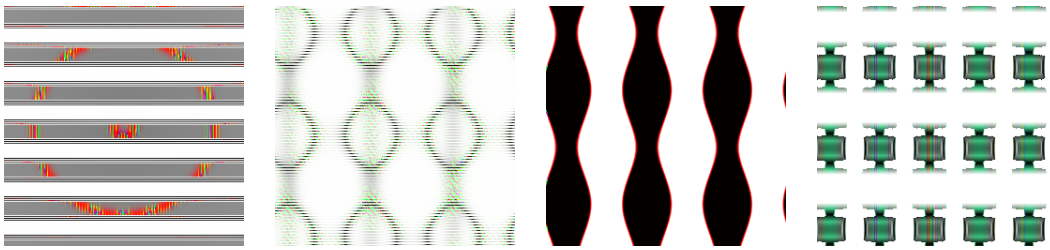


Figure 6: Some of Correctly Classified EA images by Our Classifier. From left to right they are misclassified to n06874185 (traffic light, traffic signal, stoplight), n03443371 (goblet), n04522168 (vase) and n03742115 (medicine chest, medicine cabinet)

(a) (b) (c) (d)

Figure 7: Images Misclassified by OpenSet Method but Correctly Classified by Our Classifier. (c) and (d) are from LBFGS-Adversarial dataset, which is misclassified to category n02133161(American black bear) and n02328150(Agona).