DETECTING ADVERSARIAL PERTURBATIONS WITH SALIENCY

Anonymous authors
Paper under double-blind review

ABSTRACT

In this paper we propose novel method for detecting adversarial examples by training a binary classifier with both origin data and saliency data. In the case of image classification model, saliency simply explain how the model makes decisions by identifying significant pixels for prediction. Perturbing origin image is essentially perturbing saliency of right output w.r.t. origin image. Our approach shows good performance on detecting adversarial perturbations. We quantitatively evaluate generalization ability of the detector where detector trained with strong adversaries and its' saliency perform well on weak adversaries. In addition, we further discuss relationship between solving adversary problem and model interpretation, which helps us understand how convolutional neural networks making wrong decisions.

1 INTRODUCTION

Deep neural networks (DNNs) have made significant progress in classification problems \cite{Krizhevsky2012,Simonyan2014,Szegedy2015,He2016}, which have shown to generate good results when provided sufficient data. However, DNNs are found to be easily fooled by adversarial examples generated by adding small and visually imperceptible modifications on normal samples, leading to wrong classification. The existence of adversarial examples reminds us rethinking differences between human visual system and computer vision system based on DNNs.


Recently, improving DNNs’ robustness to adversarial examples has attracted the attention of many researchers. Several defense methods are proposed to classify adversarial examples correctly, while most of these methods are easily to be attacked as well.

Detection on adversarial examples is another defense task focusing on distinguishing between clean samples and adversarial samples \cite{Feinman2017,Bhagoji2017,Guo2017,Grosse2017,Metzen2017,Den2017,Li2016}. By assuming that adversarial dataset and origin dataset are intrinsically different, classifiers are trained to determine if a sample is clean or adversarial. However, these detection are can be easily destroyed by constructing a differentiable function that is minimized for fooling both classifier and detector with strong iterative attacks.

In this work, we adopt saliency, explaining how a classification DNN can be queried about the spatial support of a particular class in a given image, to tackle with detecting adversarial examples. To calculate saliency for an output w.r.t. input image, we use calculations with gradients to figure out importance of each individual pixels which is meant to reflect their influence on the final classification. Notice that a model learns wrong classification output always learns wrong features and wrong saliency as well. Using the DNN’s intrinsic quality that adversarial samples don’t completely match it’s saliency guides us training a binary classifier to know whether a given sample is real or fake.
2 BACKGROUND

In this section, we introduce notations that are used for analyzing adversarial detection problem, introduce 4 attack methods and 3 defense methods, and introduce image-specific class saliency.

2.1 Notation

Formally, given an image $x$ with ground truth $y = f_\theta(x)$, non-targeted adversarial example $x^*$ targeted adversarial example $x^*_t$ for target $t$ are suppose to satisfy the following constraints:

\begin{align*}
  f_\theta(x^*) &\neq y \\
  f_\theta(x^*_t) &= t \\
  d(x, x^*) &\leq B
\end{align*}

where function $d$ denote distance metric to quantify similarity and $B$ denote upper bound of allowed perturbation $\epsilon$ to origin image.

In the case of DNNs, the classification model $f_\theta$ is a highly non-linear function. To seek out which pixels leading to wrong classification when given adversarial sample, $f_\theta$ is usually approximated as a linear function:

\[
  f_\theta(x) = \theta_w x + \theta_b
\]

The image-specific class saliency can be calculated as the derivative of $f_\theta$ w.r.t. input at the image $x$.

\[
  \theta_w = \frac{\partial f_\theta(x)}{\partial x}
\]

The computation of the image-specific saliency map for a single class is extremely quick, since it only requires a single back-propagation pass.

2.2 Crafting Adversarial Examples

Fast Gradient Sign Methods (FGSM) and Iterated Fast Gradient Sign Methods. \cite{Goodfellow2014} proposed a simple gradient based algorithm to generate adversarial examples. With a hyper-parameter step-width $\epsilon$, adversarial example can be generated by performing one step in the direction of the gradients sign:

\[
  x^* = x + \epsilon \cdot \text{sign}(\frac{\partial f_\theta(x)}{\partial x})
\]

FGSM is a weak attack which is not designed for generating the minimal adversarial perturbations. \cite{Kurakin2016} introduced an iterative version of the fast gradient sign methods, where replace step-width $\epsilon$ with multiple smaller steps $\alpha$ and setting clip value $\epsilon$ for accumulated perturbations in all iterations. Iterated FGSM start by setting $x^*_0 = x$, and for each iteration $i$ computing $x^*_i$ with:

\[
  x^*_i = \text{clip}_\epsilon(x^*_{i-1} + \alpha \cdot \text{sign}(\frac{\partial f_\theta(x)}{\partial x}))
\]

Jacobian-based Saliency Map Approach (JSMA). \cite{Papernot2015} proposed a greedy algorithm using the Jacobian to determine choosing which pixel to be perturbed.

\[
  s_t = \frac{\partial t}{\partial x_t}; s_o = \sum_{j \neq t} \frac{\partial j}{\partial x_t}; s(x_t) = s_t | s_o | \cdot (s_t < 0) \cdot (s_o > 0)
\]

In Equation 8, $s_t$ represents the Jacobian of target class $t$ w.r.t. input image and $s_o$ represents sum of Jacobian values of all non-target class. Changing the selected pixel will significantly increase the likelihood of the model labeling the image as the target class. Clearly, JSMA attack works towards optimizing the $L_0$ distance metric.
**C&W’s Attack.** [Carlini & Wagner (2017b)] proposed an attack by approximating the solution to the following optimization problem:

\[
\arg\min_d (s, x + \delta) + c \cdot l(x + \delta) \tag{9}
\]

where \(l\) is objective function for solving \(f(x + \delta) = t\). In this work, we choose \(l(x^* = \max(\max_i(Z(x^*)), t) - Z(x^*_t, -\kappa)\), where \(\kappa\) is the hyper-parameter controlling the confidence of misclassification.

### 2.3 Detecting Adversarial Examples

[Grosse et al. (2017)] train a “\(N + 1\)” classification model \(D\) to detect adversarial examples with the method of adding these samples to the training set, assigning a new \(N + 1\)st label for them. However, experiments in [Carlini & Wagner (2017a)] shows that this detection failed distinguishing adversarial examples at nearly 0\% accuracy under a second round attack. Experiment in in [Carlini & Wagner (2017a)] also shows that this detection methods cannot resist black-box attack where attackers have no access to \(D\). By splitting training set in half for individually training two models, \(D\) and imitated \(D\), C&W’s Attack succeed 98\% when fooling \(D\) using parameters for attacking imitated \(D\).

[Gong et al. (2017)] construct a “\(1 + 1\)” classification model by means of regarding real data and fake data as two completely different datasets despite being visually similar. Because of the intrinsic similarity between “\(N + 1\)” detection model and “\(1 + 1\)” detection model, this method also failed at second round attack in nearly 0\% accuracy for detecting adversarial examples. Black-box attack on “\(1 + 1\)” doesn’t show significant difference with “\(N + 1\)”.

[Metzen et al. (2017b)] augment the base network by adding subnetworks as branches at some layers and produce an output \(p_{adv} \in [0, 1]\) representing the probability of the input being adversarial. By training the subnetworks with a balanced binary classification dataset consist of clean data and fake data generated by attacking freezed base network, the subnetwork can detect adversarial examples at the inner convolutional layers of the network. Similar to above two second round attacking methods, [Metzen et al. (2017b)] propose an iterative calculating methods:

\[
x_{adv}^{n+1} = \text{clip}_x \left\{ x_{adv}^n + \alpha (1 - \sigma) \cdot \text{sgn}(\nabla_x J_f(x_{adv}^n, y_{true}(x))) + \sigma \cdot \text{sgn}(\nabla_x J_d(x_{adv}^n, 1)) \right\}
\]

Parameter \(\sigma\) is used for trading off objective for base classifier \(f\) and objective for detection classifier.

### 2.4 Gradients as Saliency

A common approach to understanding the decisions of image classification systems is to find regions of an image that were particularly influential to the final classification [Baehrens et al. (2010); Zeiler & Fergus (2014); Springenberg et al. (2014); Zhou et al. (2016); Selvaraju et al. (2016); Zintgraf et al. (2016)]. At visual level, saliency represents discriminative pixels for model making decisions and Karen Simonyan Simonyan et al. (2013) launch weakly supervised object segmentation experiment only rely on saliency map. Saliency of wrong decision caused by fake sample always visually different from Saliency derived from right sample.

### 3 Methodology

As is shown in [4] when an image is perturbed by attacking method, saliency of classification output w.r.t. adjusted image is perturbed as well. Accordingly, We follow the steps below building our detection system.

**Step1.** Train a classifier \(f\) with origin training dataset \(X_{train}\), then craft adversarial dataset \(X_{adv}^{train}\) and \(X_{adv}^{test}\) by attacking \(f\) using FGSM/Iterated FGSM/JSMA.

**Step2.** By calculating saliency for each image in \(X_{train}, X_{test}, X_{adv}^{train}\) and \(X_{adv}^{test}\) based on the attacked classifier \(f\), we create saliency dataset \(S_{train}, S_{test}, S_{adv}^{train}\) and \(S_{adv}^{test}\).

**Step3.** We apply both raw data and saliency data as input for training binary classifier \(D\). Raw data and saliency data are concatenated on channel axis in our experiment.
Figure 1: Origin image from MNIST, CIFAR10, IMAGENET dataset and their corresponding saliency. For each four-grids sample, left parts display clean data and right parts display fake data attacked by FGSM. Lower half in four-grids sample represent corresponding saliency for upper half images.
Table 1: Accuracy on adversarial samples generated with FGSM, Iterative $l_1$ and Iterative $l_\infty$

<table>
<thead>
<tr>
<th>Dataset</th>
<th>FGSM Iterative $l_\infty$/ Iterative $l_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$f(x_{test})$</td>
</tr>
<tr>
<td>MNIST</td>
<td>0.99/ 0.99/ 0.99</td>
</tr>
<tr>
<td>CIFAR10</td>
<td>0.81/ 0.81/ 0.81</td>
</tr>
<tr>
<td>10-IMAGENET</td>
<td>0.90/ 0.90/ 0.90</td>
</tr>
</tbody>
</table>

We evaluate false positive and true positive rates of detector. Furthermore, we evaluate two kinds of generalizability of $D$: 1) Attack by the same adversary with different $\epsilon$ and 2) Attack by one adversary when tested on data from other adversaries when fixing $\epsilon$.

4 EXPERIMENT

In this section, we present results of accuracy on detecting adversarial samples generated with FGSM, Iterative $l_1$ and Iterative $l_\infty$ with 3 datasets: MNIST, CIFAR10, IMAGENET subset Russakovsky et al. (2015). We evaluate generalizability of $D$ for the same attack on $f$ with different choices of $\epsilon$. We also evaluate generalizability of $D$ for the same perturbation extent $\epsilon$ with different attacking methods on $f$.

4.1 IMPLEMENTATION DETAILS

Our experiment is implemented in Keras 2.0 and tensorflow 1.0 (Abadi et al., 2016). Deep neural networks we adopt for Classifier $f$ network and Detector $D$ network are shown in Figure 2. For MNIST/CIFAR10 dataset, Detector(D) network is smaller than Classifier network since intuitively adversarial binary classification task extract less features. Besides, all DNNs for MNIST/CIFAR10 datasets are trained from scratch. We follow Metzen et al. (2017) dataset collecting method, randomly selecting 10 classes from Imagenet training set and validation set. The random selected classes are: mongoose; plant, flora, plant life; Yawl; timber wolf, grey wolf, gray wolf; Canis lupus; dugong, Dugong dugon; hammer; sunglasses, dark glasses, shades; typewriter keyboard; triumphal arch; mushroom. Therefore, We have 10000 images in train set, 3035 images in validation set and 500 images (from ImageNets validation data) in test set. The motivation of using subset instead of full-dataset is of two-fold: 1) to reduce computation cost of crafting adversarial dataset, 2) to avoid adversarial conversion between similar classes, eg. perturbing image recognized as sea snake to image recognized as water snake is not constructive. We employ VGG16 and its parameters from Caffe model zoo on initializing $f$ and $D$ for 10-CLASSES IMAGENET.

We employ 4 typical attacking algorithms in this paper: FGSM, Iterative FGSM with $l_2$ distance, Iterative FGSM with $l_\infty$ distance and JSMA. We revise origin FGSM to avoid label leaking problem (Kurakin et al., 2016). JSMA is not applied to Imagenet subset for its’ low efficiency on pixel searching when attacking images of size 224*224*3.

4.2 MNIST/CIFAR10

We train MNIST-NET-f shown in Figure 2 for 10 epochs with Adam optimizer (Kingma & Ba, 2014) and learning rate was set to 0.001. MNIST-NET-f run up to 99.73% and 99.32% accuracy on training data and test data respectively. Afterwards, adversarial dataset was generated with 4 attacks. With clean data and adversarial data, we calculate saliency maps for all images. MNIST-NET-D are trained for 10 epochs with Adam optimizer where learning rate was set to 0.0001. CIFAR10-NET-f are trained for 100 epochs with Adam optimizer where learning rate was set to 0.0001, CIFAR10-NET-f run up to 83.89% and 81.32% accuracy on training data and test data respectively. CIFAR10-NET-D are trained for 5 epochs with Adam optimizer where setting learning rate as 0.0001. False positive and True positive rates of MNIST-NET-D and CIFAR10-NET-D are shown in Table 1.

Results in Figure 3 show similar performance of generalizability where a $D$ trained with large $\epsilon$ cannot reach a good effect on adversarial samples generated with small $\epsilon$. Meanwhile, $D$ trained with adversarial samples crafted with small $\epsilon$ generalized acceptably well to all adversarial samples.
Figure 2: Deep neural network used in our implementation for different datasets, called MNIST-NET-f, MNIST-NET-D, CIFAR10-NET-f, CIFAR10-NET-D, VGG16-f and VGG16-D in following passage. MNIST-NET-f, MNIST-NET-D, CIFAR10-NET-f, CIFAR10-NET-D, VGG16-f and VGG16-D are trained from scratch, and left two are finetuned with VGG parameters from Caffe Model Zoo. All pooling operations and activations are set to maxpooling and relu respectively, which are not shown in this figure.

Figure 3: Accuracy metric on MNIST/CIFAR10 of detector trained for adversary with maximal distortion $\epsilon_{fit}$ when tested on the same adversary with distortion $\epsilon_{test}$. Following [Metzen et al. (2017b)], we set $\epsilon$ as minimal under the constraint that the classification accuracy is below 30%. Result in Figure 4 shows that FGSM and JSMA generalized not good enough with detector trained with iterative($l_2$) and detector trained iterative($l_\infty$), but iterative($l_2$) based detector and iterative($l_\infty$) based detector perform well to FGSM-based adversaries and JSMA-based adversaries. CIFAR10 dataset show similar character with MNIST experiment except that JSMA and FGSM cannot generalized well to each other. Therefore, we draw the conclusion for our detec-
Results in Figure 4 shows that detector trained with stronger adversaries generalize well to detector trained with smaller perturbation upper-bound generally perform well on higher ones but not vice versa. Results in Figure 5 shows similar direction with MNIST/CIFAR10 experiment: detectors trained positive rates of VGG16-D are shown in Table 1.

For 1000 epochs with Adam optimizer Kingma & Ba (2014). Initial learning rate was set to 0.001 after 1000 epochs, and further reduced to 0.00001 after 1000 epochs. VGG16-D are trained epochs with Adam optimizer

In this experiment, we use only 3 attacking methods: FGSM, Iterative(\(l_1\)) and Iterative(\(l_{\infty}\)) for their suitable demand for computation recourses. We fine-tuning VGG16-f shown in Figure 2 for 100000 epochs with Adam optimizer [Kingma & Ba 2014]. Initial learning rate was set to 0.001, reduced to 0.0001 after 100 epochs, and further reduced to 0.00001 after 1000 epochs. VGG16-f run up to 91.82% and 89.83% accuracy on training data and test data respectively. VGG16-D are trained for 100 epochs with Adam optimizer where learning rate was set to 0.0001. False positive and True positive rates of VGG16-D are shown in Table 1.

Results in Figure 5 shows similar direction with MNIST/CIFAR10 experiment: detectors trained with smaller perturbation upper-bound generally perform well on higher ones but not vice versa. Results in Figure 6 shows that detector trained with stronger adversaries generalize well to detector trained with weaker adversaries, which is identical to MNIST/CIFAR10 evaluations.

Figure 4: Accuracy metric on MNIST/CIFAR10 of detector trained for one adversary when tested on other adversaries. The maximal distortion of the adversary (when applicable) has been chosen minimally such that the predictive accuracy of the classifier is below 30%. Numbers correspond to

Figure 5: Accuracy metric on IMAGENET subset of detector trained for adversary with maximal distortion when tested on the same adversary with distortion test. Evaluation method is the same as MNIST/CIFAR10 evaluation settings.

In this section, we concentrated on studying one question: if our detection approach could perform well on eye-level images. Empirically, adversarial examples on MNIST/CIFAR10 usually show visually distinguishable perturbation even texture and structure of origin image are changed. Therefore, many researches on defending MNIST/CIFAR10-level adversary helps little to find out the extrinsic difference between human visual system and deep neural networks. Take MNIST adversary for example, saliency of wrong output w.r.t. adversarial example seems visually approximate to its’ perturbation. However, in Imagenet-level images, these unreasonable properties found on MNIST/CIFAR10-level no longer appear.

4.3 IMAGENET SUBSET

In this section, we concentrated on studying one question: if our detection approach could perform well on eye-level images. Empirically, adversarial examples on MNIST/CIFAR10 usually show visually distinguishable perturbation even texture and structure of origin image are changed. Therefore, many researches on defending MNIST/CIFAR10-level adversary helps little to find out the extrinsic difference between human visual system and deep neural networks. Take MNIST adversary for example, saliency of wrong output w.r.t. adversarial example seems visually approximate to its’ perturbation. However, in Imagenet-level images, these unreasonable properties found on MNIST/CIFAR10-level no longer appear.

In this experiment, we use only 3 attacking methods: FGSM, Iterative(\(l_1\)) and Iterative(\(l_{\infty}\)) for their suitable demand for computation recourses. We fine-tuning VGG16-f shown in Figure 2 for 100000 epochs with Adam optimizer [Kingma & Ba 2014]. Initial learning rate was set to 0.001, reduced to 0.0001 after 100 epochs, and further reduced to 0.00001 after 1000 epochs. VGG16-f run up to 91.82% and 89.83% accuracy on training data and test data respectively. VGG16-D are trained for 100 epochs with Adam optimizer where learning rate was set to 0.0001. False positive and True positive rates of VGG16-D are shown in Table 1.

Results in Figure 5 shows similar direction with MNIST/CIFAR10 experiment: detectors trained with smaller perturbation upper-bound generally perform well on higher ones but not vice versa. Results in Figure 6 shows that detector trained with stronger adversaries generalize well to detector trained with weaker adversaries, which is identical to MNIST/CIFAR10 evaluations.
Figure 6: Accuracy metric on IMAGENET subset of detector trained for one adversary when tested on other adversaries. Evaluation method is the same as MNIST/CIFAR10 evaluation settings.

Figure 7: Weighted summation of feature maps in 'relu2-1', 'relu4-1' and 'pooling5' in VGG16 model. These three Weighted summation of feature maps roughly represent feature extracted by shallow layers, middle layers and deep layers. Visualization shows that shallow layers are robust enough to adversarial examples while middle layers start to extract wrong features, leading to deep layers' failure.

5 DISCUSSION

When we dive into the feature extraction procedure of deep convolutional neural networks, saliency seems to be semantic enough but not express enough features. We dissect deep convolutional network with a revised version of Grad-CAM to find out how adversarial examples contributing to wrong output classification.

We generalize interpretability of saliency maps to each layers by computing gradient of output w.r.t. feature maps in certain layer as feature map weights $\alpha$. Intuitively, $\alpha$ represents influence of a feature map for the final decision. Weighted summation feature maps in Figure 7 are generated referring to Selvaraju et al. (2016). Weighted summation feature maps in 'relu2-1', 'relu4-1' and 'pooling5' in VGG16 model. These three Weighted feature maps roughly represent feature extracted by shallow layers, middle layers and deep layers. Visualization shows that shallow layers are robust enough to adversarial examples while middle layers start to extract wrong features, leading to deep layers' failure.
6 CONCLUSION

We have proposed a approach for detecting adversarial examples by training a binary classifier by taking saliency perturbation information into consideration. Our approach shows 100% accuracy on detecting adversarial perturbations on MNIST dataset and show above 90% accuracy on CIFAR10, IMAGENET subset under FGSM/Iterative($l_2$)/Iterative($l_{\infty}$)/JSMA attack. By quantitatively evaluate generalization ability of the detector, we conclude that our detector trained with strong adversaries performs well on weak adversaries, proving its' generalizability and transferability. Afterwards, we further discuss relationship between solving adversary problem and model interpretation, claiming that shallow layers are robust to adversarial attack and middle layers start calculating wrong decisions.

REFERENCES


Xin Li and Fuxin Li. Adversarial examples detection in deep networks with convolutional filter statistics. 2016.


