Unrestricted Adversarial Defending Deep Neural Network

Qiwen Wang and Xinshuo Zhang

I. Introduction

Although known to be robust to random perturbations of the input, deep neural networks can be quite vulnerable to adversarial perturbations, which are designed specially to fool the model into making blatant errors[1]. One even more annoying fact is that DNN even report high confidence on such errors. Unbelievable as it may seem, a single specific rotation on a vulture picture can fool the neural network into reporting a cat picture instead. However, it also inspires us to reflect on such loopholes and try to train a more holistic DNN to avoid such errors, which leads to our projectunrestricted adversarial defending deep neural network.

Given the image classification realm, usually the goal of the adversarial attacks is to add a tiny perturbation to the image, which lead to misjudgment of a particular model yet keep the picture classifiable to human eyes[2]. In our study, we combine several images processing methods and the state-of-art adversarial defending methods. We test these methods on a binary classification task to classify handwritten digits "6" and "7". The final decision is based on the majority vote of all of these methods.

Github link: https://github.com/qwang70/cleverhans.

II. Dataset

To train and test our model, we use the 'MNIST' handwritten digit dataset. Because our task is to only classify digits "6" and "7", we take the subset of the dataset, and only train and test on digits "6" and "7". We collect 12,000 handwritten images for training, developing, and testing. Each image is in black and white, with size 28×28. As for testing, we collect 2000 images and attack these images with different attacking algorithm such as FGSM and JSMA, etc. To make sure our training and testing is unbiased, the data for our training, developing, and testing set have the same distribution, which means the numbers of digit "6"s and "7"s in each set are roughly equal.

III. Method and Result

A. Related work on Attacks

In this section, we introduce the attacking methods we use to test the model.

1) Spatial Grid Attack: The spatial grid attack[7] is based on the observation that simple image rotation and transformation can leads to image classification. The attack is composed of three steps: a) iteratively take steps in the direction of the loss functions gradient to locally maximizes the loss of the classifier, b) use grid search to explore possible parametrization of the attack, and find the parameter that makes the classifier to give a wrong prediction, and c) randomly sample k attack parameters and choose the parameter that the model performs the worst.

The parameters involved in the spatial grid attack are rotation degree θ and pixel transformation $(\delta u, \delta v)$. Let $T(x; \delta u, \delta v, \theta)$ be the transformation of x and y be the correct label of x. Then the problem can be formalized as

$$\max_{\delta u, \delta v, \theta} L(T(x; \delta u, \delta v, \theta), y).$$

2) JSMA: Basically, what JSMA does can be summarized into two steps. Step one, we find input and output pairs where output includes every class other than the labeled one and we compute the Jacobian component corresponding to each pair. The Jacobian matrix indicates changing which pixel can have the most significant influence on the output by the model. Step two, generate the adversarial input based on the Jacobian matrix. Here, since our input images are "6" and "7", JSMA takes input "6" and output class "7" as a pair and compute Jacobian matrix for such pair. Based on the matrix, JSMA changes a few pixels, namely the several white dots in Figure 1, which

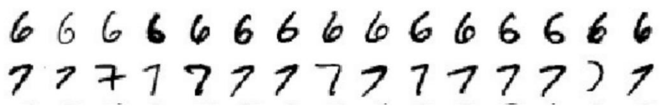
Fig. 1. MNIST handwritten digit dataset of “6”s and “7”s

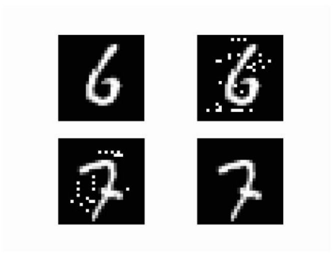
Fig. 2. The raw images and the images after JSMA attack. The images on the left are classified as “6”, and images on the right are classified as “7”.

successfully fool the network to classify the “6” into “7” yet it remains “6” to human eyes. Figure 2 is an example of the result of JSMA attack.

3) Fast gradient sign method: Goodfellow et al.[8] proposed the fast gradient sign method (FGSM) to generate adversarial examples.

Fast Gradient Step Method (FGSM) adds some weak noise on each step, trying to maximize the error and drifting the images classification towards the other class, while keeping the image look similar to human. In each step, FGSM performs

$$x + \epsilon \cdot sgn(\nabla x L(\theta, x, y)).$$

We use the cross entropy

$$-(y(p) + (1-y)(1-p))$$

as our loss function, where y is a binary indicator of the class label, and p is the predicted probability.

B. Baseline Models

We train and test our baseline model by 1) not applying any image processing method, 2) only pre-processing images but not using any adversarial dense method during the training, and 3) training a model using adversarial training. The baseline models are evaluated by looking at accuracy, recall, F1 score, and the confusion matrix.

C. CNN model without pre-processing

We first train a two-layer convolutional model with max-pooling for 10,000 iterations. We then test our model on a clean dev set, and dev set that is attacked by three individual methods, spatial grid attack, JSMA and FGSM. We achieve a near 100% accuracy on the clean dev set, but we have 0% accuracy for the three attacks. The result is reasonable, because these attack algorithms are designed specifically to let the image be misclassified.

TABLE I
BASELINE RESULT FOR USING CNN WITH ATTACKED IMAGES

	Precision	Recall	F1
Clean	1	1	1
Spatial Grid	0	0	0
JSMA	0	0	0
FGSM	0.722	0.746	0.733

D. CNN model with pre-processing

1) Gaussian Blurring: We apply Gaussian blurring on the training image to train the model. Usually, a modern image uses 8 bits per pixel, which means all information below 1/255 of the dynamic range is discarded, i.e. the precision of the features is limited. Such feature decides that any perturbation smaller than the precision should not be detected. However, as the dimension of the weight vector grows, even though the perturbation remains unchanged, the changes in activation caused by the perturbation grows. This is a simple explanation why adversarial model works and by applying Gaussian blurring, hopefully we can average out such changes in a image. The model is trained using the same CNN design as in the first baseline. When testing the model, after applying the attacks to the testing image, we also apply the same Gaussian blurring with the same parameter . A Gaussian blur is the result of blurring an image with a Gaussian function and is typically used for reducing image noise and reducing detail.

By applying Gaussian blur, that is

$$G(x, y) = \exp(-\frac{x^2 + y^2}{2\sigma^2}),$$

the attack on the image can be dispersed. Then the same evaluation metrics are used to test the result. We first train the CNN model after applying Gaussian blurring, than test on attacked images that are applied Gaussian blurring. The result in Table II is still not promising on spatial grid and FGSM attack, but is better than the first baseline.

TABLE II

EVALUATION RESULT FOR THE CNN MODEL WITH IMAGE PRE-PROCESSED BY GAUSSIAN BLURRING WITH $\sigma = 2$.

	Precision	Recall	F1
Clean	0.999	1	0.999
Spatial Grid	0.535	0.627	0.577
FGSM	0.761	0.549	0.638

2) Feature Squeezing: Proposed by Xu et al.[9], this is another spatial smoothing method that reduces the color bit depth of each pixel. Since the input spaces of the images are often unnecessarily large, and provide the adversarial attacking methods opportunity to construct adversarial example, this approach squeezes the features and provides less degree of freedom to be attacked. It runs a sliding window on each pixel to substitute color with median value in the sliding window. This median selection can effectively remove sparsely-occurring black and white pixels, whilst preserve edges of objects. We first train the CNN model after applying Feature Squeezing, than test on attacked images that are applied Feature Squeezing. The result in Table III performs better than the baseline without image pre-processing, and performs better on FGSM compared to using Gaussian blurring.

TABLE III

EVALUATION RESULT FOR THE CNN MODEL WITH IMAGE PRE-PROCESSED BY FEATURE SQUEEZING.

	Precision	Recall	F1
Clean	0.999	1	0.999
Spatial Grid	0.483	0.633	0.548
FGSM	0.924	0.931	0.928

E. CNN Model Trained with state-of-art Defending Algorithms

1) FGSM Defending: One of the attacks we're aiming to defend the category of targeted attacks that maximize the probability of targeted adversarial class. According to Szegedy et al.[5], training on a mixture of clean images and adversarial attacked images can generalize neural network to a certain point. Therefore, here we train the model while applying FGSM attack on training set, and test the model with other attacks.

The result for detecting the FGSM attack is promising, because the model is trained based on it, but we make the assumption that it can also detect other attacks in the same category.

We trained the model by generating FGSM example and defending. Then we tested the model on images with attacks without pre-process the images. The result is shown in Table IV. Since the model is trained based on FGSM attack, the testing result for the FGSM attack outperforms the previous image processing approach

TABLE IV

EVALUATION RESULT USING FGSM DEFENDING ALGORITHM.

	Precision	Recall	F1
Clean	0.999	1	0.999
Spatial Grid	0.496	0.672	0.57
FGSM	0.99	0.981	0.986

2) Black-box: Unlike other methods mentioned above, which apply gradient method through direct access to real network gradient factors, here black-box construct an implicit approximation to the network gradient, namely the so-called substitute model, through local search technique[6]. We try to minimize the probability that the input image is classified as its labeled class by computing an implicit approximation gradient of the current image. In each iteration of the local search, we select a set of pixels based on the pixels perturbed in the previous iteration, using the pixel selection formula:

$$(Px, Py) = \bigcup_{\{(a,b)\in(P_X^*, P_Y^*)_{i-1}\}} \bigcup_{\{x\in[a-d,a+d], y\in[b-d,b+d]\}} (x, y).$$

After selecting, we apply perturbation function on pixels selected. See perturbation function below:

$$I_p^{(x,y)}(b,u,v) = \begin{cases} (I(b,u,v)), & \text{if } x \neq u \text{ or } y \neq v \\ p \times sign(I(b,u,v)), & \text{otherwise} \end{cases}.$$

After several rounds, we get a set of images in which different pixels are perturbed and we sort these images in descending order of how much decrease it leads to the original classification probability, which indicates how robust an adversarial image the corresponding pixels contributes to.

Table V shows the result running test example on black-box algorithm trained model. The algorithm performs slightly worse on clean image, and performs very poor on FGSM attack.

TABLE V
EVALUATION RESULT USING BLACK-BOX DEFENDING ALGORITHM.

	Precision	Recall	F1
Clean	0.995	0.985	0.990
Spatial Grid	0.526	0.502	0.514
FGSM	0.015	0.014	0.015

F. Integrated Adversarial Defending Model

As stated before, most attacks are crafted specially to cause deep learning algorithms to misclassify. Reflecting upon this feature, a conclusion drew directly from the majority vote can be misleading. Since a particular attack is very likely to be only detected by the network which is trained on the dataset attacked by the very same attacking algorithm and thus fool all the other networks to vote for it instead of against it, the majority vote may just output the contrary result. Under this circumstance, training another neural network using the output of all the neural network we trained for adversarial attack defending as input seems a good way to go.

1) Train models with image pre-processing and defending algorithms: From Section III-D and III-E, we found that by pre-processing images or by using the state-of-art defending algorithms, we are able to increase the classification accuracy for some of the attacks. It is nature to that combining these two approaches can lead to a better result. In the training phase, we first apply image processing to the input images, then use FGSM or black-box defending algorithms to train models based on the processed image. Since we experimented multiple image processing approaches and defending methods, with different combinations, we ended up with 6 models in this step. The 6 trained models are described in Table VI.

2) Integrate trained models into one adversarial defending model: Since these models can differentiate attacks in different extent, for some unknown attacks, it is likely that some models can classify the attack better than others. We trained an integrated model by using the logits of each of the 6 models as features, training a multi-layer perceptron (MLP), and using the original labels of the training images as the labels of the integrated neural network. The intuition behind the network is to find the most similar logit features of the trained image to the logit features of the attack image.

Since the problem is a binary classification task, every model has two logits before outputting the classification label. Each input data for training the MLP model has $2 \times 6 = 12$ features. Thus the input feature is of size $\mathbb{R}^{N \times 12}$ where $N = 12000$ is the number of training data. The MLP model has 2 hidden layer with 256 neurons, and 1 output layer that maps the data into 2 classes representing "6" and "7". Figure 3 and 4 show the structure of the Integrated Adversarial Defending Model for training and evaluating.

TABLE VI
MODELS TRAINED WITH IMAGE PROCESSING AND DEFENDING ALGORITHM COMBINATION

	Model 1	**Model 2**	**Model 3**	**Model 4**	**Model 5**	**Model 6**
Pre-process	Clean Image	Gaussian Blurring	Feature Squeezing	Clean Image	Gaussian Blurring	Feature Squeezing
Defend Method	FGSM	FGSM	FGSM	Black-box	Black-box	Black-box

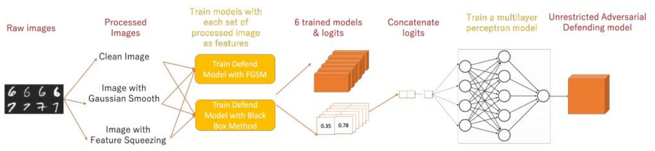
Raw images
Processed Images
Train models with each set of processed image as features
6 trained models & logits
Concatenate logits
Train a multilayer perceptron model
Unrestricted Adversarial Defending model
Clean Image
Image with Gaussian Smooth
Image with Feature Squeezing
Train Defend Model with FGSM
Train Defend Model with Black Box Method
0.35
0.78

Fig. 3. Integrated Adversarial Defending Model Architecture for Training

G. Model Result

We applied image pre-processing methods and Adversarial defending algorithms on 12,000 input images of handwritten digits "6" and "7". After we trained 6 individual models with a combination of

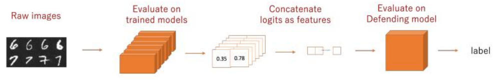
Raw images
Evaluate on trained models
Concatenate logits as features
Evaluate on Defending model
0.35
0.78
label

Fig. 4. Integrated Adversarial Defending Model Architecture for Evaluating

the above approaches, we concatenate the logits of each model as features and trained the final integrated model as shown in Figure 3. Then we evaluate the 2,000 images on the model by first applying attacks, then applying image processing. We passed the image to 6 models, to get the features for the integrated model. Then the final label is decided by the integrated model (Table VII).

TABLE VII
EVALUATION RESULT USING TESTING DATA ON INTEGRATE MODEL.

preprocess	attack	acc	precision	recall	f1
clean	none	0.998	0.996	1	0.998
	spatial_grid	0.51	0.522	0.722	0.606
	fgsm	0.019	0.027	0.026	0.027
gaussian	none	0.999	0.999	1	0.999
	spatial_grid	0.519	0.529	0.734	0.615
	fgsm	0.472	0.315	0.017	0.031
squeeze	none	0.998	0.998	0.999	0.998
	spatial_grid	0.5125	0.525	0.715	0.605
	fgsm	0.232	0.056	0.03	0.039

Compared to the baseline and individual models we generated, notice that classifying FGSM attack only performs better than using the black-box model. From Table V, the black-box method performs badly on FGSM attack. It is likely that MLP puts considerate weight on black-box models, but since the FGSM performs so badly on black-box model, and we don't train image based on the attacked image, the final classification for FGSM attack is bad.

Despite of the fact, the precision, recall f1 of classifying FGSM attacked images that is clean, Gaussian blurred, and feature squeezed is better than only using the black-box model. Also performing Gaussian blurring and feature squeezing gives better result than without perform image preprocessing. It verifies that image pre-processing is useful for the adversarial defend.

The accuracy, precision, recall and f1 score for classifying spatial grid attack with different image pre-processing are roughly the same. Applying Gaussian blurring and squeezing on the testing images gives slightly better result than using the clean image. Also the result from the integrated model is much better than purely using FGSM defending and is slightly better than purely using the black-box defending algorithm.

IV. CONCLUSION

In conclusion, we can't prove that our new model doesn't significantly perform better than any other individual model. We have shown that the image pre-processing and using the state-of-art defending models can improve the classification performance, but we can only conclude that our model performs better than the individual model with the worst performance.

In the future work, instead of using logits of size 2 as feature, we can explore using more logits even for the binary classification. In our work, the MLP only uses two basic fully connected layers. We can improve the model by substituting the final MLP model with a neural network with more complicated structure. Mostly importantly, we want to try more combination of image processing and defending algorithm, and select some of them that perform well for the integrated model. One of the problem we have is to integrate all the models we have. Some of them might only add little useful information to the overall model, and increase the dependency between features.

V. CONTRIBUTION

Qiwen Wang and Xinshuo Zhang contribute to the project equally.

REFERENCES

[1] Papernot, Nicolas, et al. "cleverhans v2. 0.0: an adversarial machine learning library." arXiv preprint arXiv:1610.00768(2016).
[2] Samangouei, Pouya, Maya Kabkab, and Rama Chellappa. "Defense-GAN: Protecting classifiers against adversarial attacks using generative models." arXiv preprint arXiv:1805.06605 (2018).
[3] Google. Unrestricted Adversarial Example. https://github.com/google/unrestricted-adversarial-examples
[4] Tensorflow. Cleverhans. https://github.com/tensorflow/cleverhans
[5] Szegedy, Christian, Zaremba, Wojciech, Sutskever, Ilya, Bruna, Joan, Erhan, Dumitru, Goodfellow, Ian J., and Fergus, Rob. Intriguing properties of neural networks. ICLR, abs/1312.6199, 2014b. URL http: //arxiv.org/abs/1312.6199.
[6] Narodytska, Nina, and Shiva Prasad Kasiviswanathan. "Simple Black-Box Adversarial Attacks on Deep Neural Networks." CVPR Workshops. Vol. 2. 2017.
[7] Engstrom, Logan, et al. "A rotation and a translation suffice: Fooling cnns with simple transformations." arXiv preprint arXiv:1712.02779 (2017).
[8] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. CoRR, abs/1412.6572, 2014. URL http://arxiv.org/abs/1412.6572.
[9] Xu, Weilin, David Evans, and Yanjun Qi. "Feature squeezing: Detecting adversarial examples in deep neural networks." arXiv preprint arXiv:1704.01155 (2017).