CAAD 2018 ATTACK AND DEFENCE

Xiaoyi Dong
USTC
dlight@mail.ustc.edu.cn

1. Attack Method

In this paper, we propose an improvement of Adversarial Transformation Networks(ATN) to generate adversarial examples, which can fool white-box models and black-box models with a state of the art performance. In this section, we first introduce the whole architecture about our method, then we present our improvement on loss functions to generate adversarial examples satisfying the L_∞ norm restriction in the non-targeted attack problem. Then we illustrate how to use a robust-enhance module to make our adversarial examples more robust and have better transfer-ability. At last we will show our method on how to attack an ensemble of models.

1.1. Model Architecture

Our work is based on ATN and propose a new training framework and two powerful loss functions for improving the transfer-ablity and training speed. Figer 1 shows our framework

Our framework is composed by a Generate module and a Robust-enhance module. In the Generate module, there are the Encoder and Decoder just like the ATN. But before feeding the clean image and adversarial example into the pre-trained model, we add a Robust-enhance module to imitate the image pre-process used in some defence methods.

The generating part could be defined as a neural network:

$$g_{k,\theta}(x) : x \in \chi \to x' \tag{1}$$

where θ is the parameter of $g(x)$, $k(x)$ is the target model which outputs a probability distribution across class labels and $k_f(x)$ is the feature map before the target model's last average pooling layer. x is the input image and χ is the distribution domain of the input image, x' is the adversarial example generate by the $g(x)$ and

$$argmax\ k(x) \neq argmax\ k(x') \tag{2}$$

1.2. Loss Function

With the L_∞ norm restriction, we abandon the space-domain loss and use the following loss functions to improve performance.

Feature Based Loss function: inspired by the SRGAN and Guided denoise, we use the target model's feature map before the last average pooling layer as the input images' feature and try to maximum the L_1 distance between the feature of real image and the adversarial example.

$$l_F(x, x') = 1 - L_1(k_f(x), k_f(x')) \tag{3}$$

We use this feature level loss function in the DLight team's attack model and got the third prize.

Prediction Based Loss function: we also concerned about the loss based on the target model's output and found a simple but powerful loss function.

$$l_P(x, x') = \begin{cases} P'_{fir} - P'_{sec}, & \text{if } argmax\ f(x) = fir \\ P'_{sec} - P'_{fir}, & \text{if } argmax\ f(x) \neq fir \end{cases} \tag{4}$$

where for target model f with N class, it's output prediction $f(x') = [P'_1, P'_2, \dots P'_N]$, we define fir and sec as labels of the top two probabilities in $f(x')$ and P_{fir} and P_{sec} are their corresponding probabilities. In the following experiments we will show that adversarial examples generated by the model trained with this simple loss have strong transfer-ability when comparing with other methods.

We use this loss function in the Hooin Zira's submission and won the second place in the Non-target attack competition.

1.3. Robust-enhance Module

As mentioned in 1.3, there are two main methods for defending adversarial examples: adversarial training and image pre-process. In order to attack the second defence methods, we insert a Robust-enhance module in the training process, after we get the output adversarial examples with clip, we add this Robust-enhance module on the adversarial examples and feed the processed images to the following target model. We find that with this Robust-enhance module, our adversarial examples are more robust to the second sort defence methods and get more powerful transfer-ability when attacking models trained with the adversarial examples(first sort defence methods).

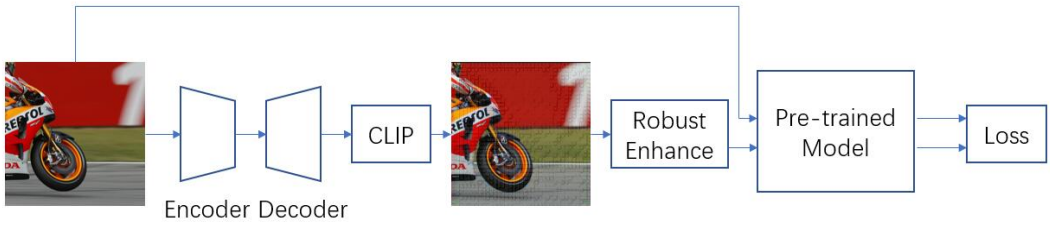

Figure 1. Architecture of our method

Considering about the implement of back propagation, we use three basic image process in the Robust-enhance module.

Random Noise: Add random noise.

Pre-trained Filter: We randomly use the Random Noise or a small pre-trained filter network to process the adversarial examples.

Training Filter:We randomly use the Random Noise or a small pre-trained filter network to process the adversarial examples, during training the Generate module, we also train the filter.

1.4. Attacking ensemble of models

In this section, we show how to attack an ensemble of models. Previous researches and competitions show that ensemble methods is a efficient method for enhancing performance and improve robustness in many area including adversarial examples. Adversarial examples generated by ensemble training are able to fool many white-box models at the same time and have better transfer-ability to attack black-box models.

We propose to attack multiple models by adding the loss together and we call this $ensemble\ in\ loss$. As our loss functions are defined in both feature level and prediction level, this ensemble in loss method could easily used with any kind of loss functions. When it comes to the back propagation step, every loss will calculate gradients individually and sum at each parameter. This will guide the model to learn how to be more aggressive to all the target models. Specifically, to attack an ensemble of N models, we fuse the losses as

$$l(x) = \sum_{n=1}^{N} w_n l_n(x) \tag{5}$$

where $l_n(x)$ are the loss function of the n-th model, the loss function here could be the feature based loss function or the prediction based loss function w_n is the ensemble weight with $w_n > 0$, we use it to keep balance of the gradients' magnitude from different models.

Meanwhile, as show in Fig.1, we find when our adversarial examples are adversarial to a model, it will be more adversarial in a few iterations. In order to forbid our generate model tend to one of the target models, we add a threshold γ for our prediction based loss l_P and get new loss $l_{ensP}(x, x')$:

$$l_{ensP}(x, x') = max(\gamma, l_P(x, x')) \tag{6}$$

2. Experiments

In this section, in order to validate the effectiveness of the proposed methods, we generate adversarial examples for fooling classifiers pre-trained on the ImageNet dataset, which consist of 1.2 million natural images collected from Internet and categorized into 1000 classes. We first specify the experimental settings in Sec.4.1. Then we show the results for attacking a single model in Sec.4.2 and an ensemble of model in Sec.4.3.

2.1. Experiment setting

We use eleven models, nine of which are normally trained models–Inception V3(Inc-v3), Inception V4(Inc-v4), Inception Resnet V2(IncRes V2), Resnet V2-101(Res-101), PolyNet, SENet154(SENet),PNASNet 5-Large(PNASNet), NASNet-A-Large(NASNet), DenseNet 121(Den-121). In order to avoid the evaluate result influenced by resize operation and easy for the ensemble models training, we fine-tune all the models with a 299×299 input by modify the pooling size of last average pooling layer. the other twp models are trained by ensemble adversarial training—Inc-v3$_{ens3}$,Inc-v3$_{ens4}$, as we don't have enough time and resources for prepare these models, we used the models shared by dongyp and transfer them from tensorflow model to pytorch model.

As we focus on fooling the target white or black box models, we use the fooling rate instead of the attack success rate. We define the fooling rate as how many adversarial examples' prediction labels are different from the origin images' prediction label. Because the classifier is not correct for all the input, in most cases, the attack success rate is higher than the fooling rate. We use the DEV imageset released by CAAD to test all of our methods.

In our experiments, we compare our methods to FGSM (one-step gradient-based) methods, MI-FGSM (iterative

Attack	Inc-v3	Inc-v4	IncRes-v2	PolyNet	NasNet	Res-101	Inc-v3$_{ens3}$	Inc-v3$_{ens4}$	Mean
FGSM	0.71	0.24	0.23	0.37	0.18	0.34	0.13	0.11	0.22
PGD	**0.99**	0.18	0.12	0.22	0.09	0.18	0.11	0.07	0.14
MI-FGSM	**0.99**	0.42	0.38	0.46	0.23	0.44	0.13	0.11	0.31
F-ATN(No Robust)	0.91	0.63	0.59	0.69	0.47	0.65	0.28	0.27	0.51
F-ATN	0.91	0.69	0.6	0.74	0.51	0.76	0.30	0.29	0.55
P-ATN(No Robust)	0.98	0.97	0.92	0.96	0.91	0.93	0.58	0.47	0.82
P-ATN	0.97	**0.97**	**0.93**	**0.96**	**0.89**	**0.94**	**0.83**	**0.80**	**0.90**

Table 1. Results for single model attack. Inc-v3 is the target white-box model and others are black-box models. We compare our methods P-ATN, F-ATN with FGSM, PGD and MI-FGSM, we find our method have a similar performance comparing with start of the art method MI-FGSM on white-box attack and have a much higher fooling rate when attacking black-box models. We also test the performance of Robust-enhance module, P-ATN(No Robust) and F-ATN(No Robust) show the fooling rate of model without Robust-enhance module during training, results shows that with Robust-enhance module, P-ATN's performance have a great progress when attacking black-box models, but F-ATN only have a small improvement. The last row shows the average black-box fooling rate, P-ATN have the best performance

methods) and PGD(iterative methods). Meanwhile we also compare with the method we followed — ATN(based on Autoencoder), as ATN's loss function is designed for targeted attack, we modify the loss for non-targeted by minimize the true label's prediction. Since optimization-based methods cannot explicitly control the distance between the adversarial examples and the corresponding real images, we don't compare with these methods.

2.2. Attacking a single model

We show the fooling rates of attacks against the models we consider in Sec.4.1 in Table 1. The adversarial examples are generated for Inc-v3 useing FGSM, MI-FGSM and PGD and four of our methods:feature loss based ATN(F-ATN), prediction loss based ATN(P-ATN) and both of the models without the Robust-enhance module. Inc-v4, IncRes-v2, PolyNet, NasNet, Res-101, Inc-v3$_{ens3}$, Inc-v3$_{ens4}$ are black-box models for evaluate transfer-ability of all the methods.The maximum perturbation ϵ is set to 16 among all experiments, with pixel value in [0,255]. The number of iterations is 10 for MIM-FGSM, and the decay factor μ is 1.0 as used in MIM. The noise mean factor β is 6 in both P-ATN and F-ATN

From the table we can observe that our two models could attack the white-box model with a near 100% fooling rate like MI-FGSM and better than FGSM and PGD. But when it comes to the black-box attack, it can be seen that the performance of FGSM, MI-FGSM and PGD are decrease largely, especially when attacking the adversarial trained models Inc-v3$_{ens3}$ and Inc-v3$_{ens4}$, all of the three attack is powerless. But with Robust-enhance module, both of our F-ATN and P-ATN still keep a high fooling rate. The last row of Table 1 shows the average black-box fooling rate, P-ATN have the best performance.

Meanwhile, We also test the performance of Robust-enhance module, P-ATN(No Robust) and F-ATN(No Robust) show the fooling rate of model without Robust-enhance module during training, results show that with Robust-enhance module, P-ATN's performance have a great progress when attacking black-box models, but F-ATN only have a small improvement.

Roubst method	Inc-v3	Inc-v3$_{ens3}$	Inc-v3$_{ens4}$
None	0.98	0.58	0.47
Random Noise	0.97	**0.83**	**0.80**
Pre-trained Filter	0.97	0.43	0.56
Training Filter	0.97	0.59	0.43

Table 2. Results.

Although our method improve the success rates greatly for black-box attack even the target is adversarial trained, the performance is not good enough(less than 90%), we will show that with muti-model ensemble training, our methods will get a better result.

2.2.1 Performance related with Robust-enhance module

The Robust-enhance module is the most important part for improving our model's transfer-ability and robustness. Therefore, we study the difference between the Random Noise method, Pretrain Filter method and Maxmin Filter method.

We attack Inc-V3 model by P-ATN with three only random noise method, pretrained filer combine with noise and Maxmin method. For the noise method, the noise max mean factor β is 6. We show the fooling rate of the generated adversarial examples against Inc-v3, Inc-v3$_{ens3}$, Inc-v3$_{ens4}$ in Table 2. Table 2 shows the result of different methods and random noise is the best one.

Attack	Resize	Inc-v3	IncRes-v2	Res-101
FGSM	N	0.71	0.28	0.34
FGSM	Y	0.46	0.21	0.30
PGD	N	0.99	0.12	0.18
PGD	Y	0.36	0.09	0.13
MI-FGSM	N	0.99	0.38	0.45
MI-FGSM	Y	**0.63**	0.25	0.34
P-ATN	N	0.97	0.93	0.95
P-ATN	Y	0.54	**0.34**	**0.72**

Table 3. Results. Ours is better.

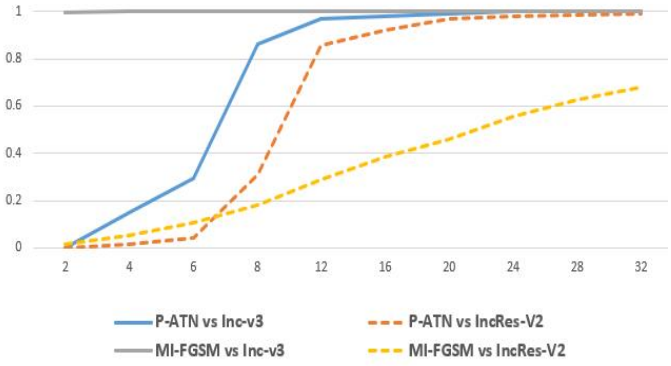

Figure 2.

2.2.2 Performance on attacking model with image resize

We then study the adversarial examples' robustness when the black-box model use some image preprocess methods to defence attack. We use the same hyper-parameters for all the attack methods as Sec.4.2, and before we feed the adversarial examples to the target model, we resize it from 299 to 399, then from 399 to 199, finally we resize it back to 299. And the attack performance shows in Table 3.

The result shows that after the resize option, all the attack performance for white-box attack(Inc-v3) decrease, especially our methods. As for black-box attack(IncRes-v2 and Res-101) results, the gradients based methods' performance just have a slightly decrease and our method's fooling rate are influenced seriously, but still better than other methods.

2.2.3 Performance related with size of perturbation

We finally study the influence of the size of adversarial perturbation on the fooling rates. We attack the Inc-v3 model by P-ATN and MI-FGSM with ϵ from 1 to 32 with a granularity 4 and the pixels range is [0,255]. We evaluate the attack performance on white-box model Inc-v3, a black-box model IncRes V2. For P-ATN, we use Random Noise module with $\beta = 6$ and step size α for PGD and MI-FGSM is 10. As it cost many time for training P-ATN with different epsilon, we just train the model with epsilon 4,8,16,32 and clip to generate different perturbation.

Attack	IncRes-v2	Inc-v3$_{ens3}$
MI-FGSM	0.955	0.949
F-ATN	0.971	0.963
P-ATN	**0.998**	**0.997**

Table 4. Results.

Figer. 2 show the result. We find that when attacking white-box model Inc-v3, MI-FGSM keeps a high fooling rate for all the epsilon. When the epsilon is small(2,4,6), P-ATN have a poor performance, with the epsilon grow, the fooling rate reach 100%. When attacking the black-box model, fooling rate of MI-FGSM grow linearly with the size of perturbation, and P-ATN's fooling rate grow exponentially, when the epsilon is large than 8, P-ATN got a better performance and as last reach 99%.

2.3. Competition by ensemble of models

There are three sub-competitions in Competition on Adversarial Attacks and Defenses 2018 organized by GeekPwn, which are the Non-targeted Adversarial Attack, Targeted Adversarial Attack and Defense Against Adversarial Attack. The organizers provide 1000 ImageNet-compatible images for evaluating the attack and defense submissions. in the non-targeted attack, we won the second place by P-ATN and third place by F-ATN.

For both of the network, we used ten pretrained models mentioned in Sec4.1 except Inc-v3. and Robust-enhance module. The noise mean factor β is 6 in both P-ATN and F-ATN and we set the ensemble threshold factor γ as -0.9. For PolyNet and Inc-v3$_{ens3}$, we set the weight as 0.5 and the rest Inc-v3$_{ens4}$, Inc-v4, IncRes V2, Res-101, SENet, PNASNet, NASNet and Den-121, we set the weight as 1.0. Table 4 shows the performance comparing with last year's winner's submission .

3. Defence Method

In this paper, we propose an combination of Neural Network based denoise filter and adversarial training to defence adversarial examples. In this section, we first show the architecture of our method, then we present the detail of the denosie filter, at last we will show our method on adversarial training.

3.1. Method Architecture

In recent research, there are about two main sort of methods for defending adversarial examples. First sort of defence is image preprocess, this sort of methods try to interfere the adversarial noise's effect or remove the adversarial noise from the adversarial examples. Reszie, compress

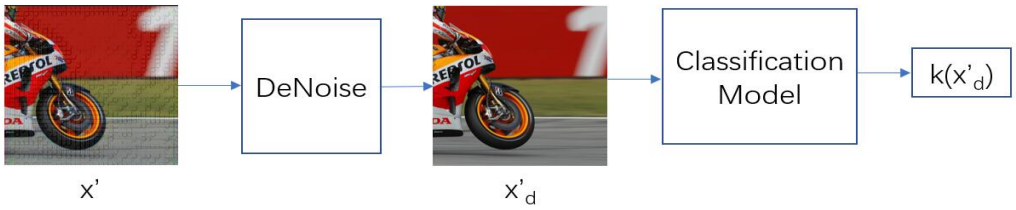

Figure 3. Architecture of our method

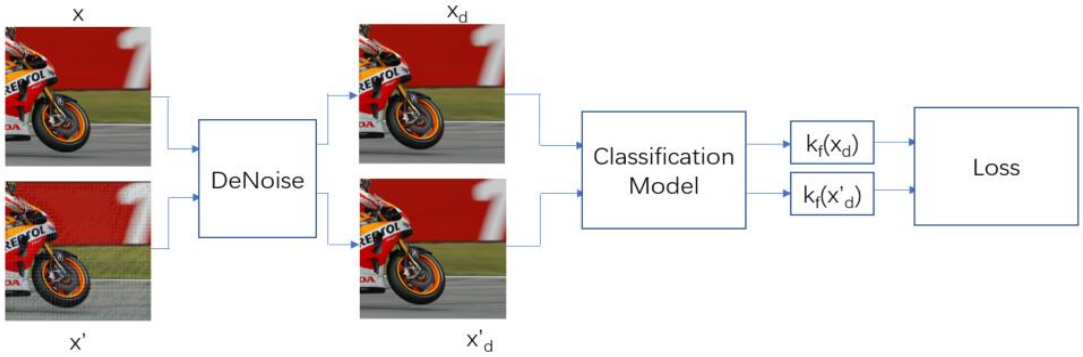

Figure 4. Process of train Denoise model

layer name	output size	
Down Sample	144 × 144	
conv1	144 × 144	3×3,64
ResidualBlock1	144 × 144	3×3,64
ResidualBlock2	144 × 144	3×3,64
ResidualBlock3	144 × 144	3×3,64
ResidualBlock4	144 × 144	3×3,64
ResidualBlock5	144 × 144	3×3,64
ResidualBlock6	144 × 144	3×3,64
conv3 +maxpooling	13 × 13	3×3,192*2,stride 1
conv4	13 × 13	3×3,192*2,stride 1
conv5	13 × 13	3×3,128*2,stride 1
maxpooling		
fc1	1 × 1	4096
fc2	1 × 1	4096
fc3	1 × 1	1000

Table 5. Architecture of Donoise model

or padding are used in the interfere idea. For the denoise idea, researchers use many kind of filters to denoise the image, we follow this idea and use Neural Network based denoise filter like High-Level Representation Guided Denoiser(HGD). Second sort of defence is adversarial training, it means finetune model with adversarial examples to make the model more robust to the adversarial examples, we follow this idea and use the training method like Adversarial Logit Pairing(ALP).

3.2. Denoise

Inspired by HGD and Super Resolution GAN (SRGAN), we use a autoencoder as denoise model D to denoise the adversarial examples. The Denoise model could be defined as a neural network:

$$D_\theta(x) : x \in \chi \rightarrow x_d \tag{7}$$

where θ is the parameter of $g(x)$, x is the input image and x_d is the denoised image.

As shown in Figure 4, during training, we feed the origin image x and it's adversarial examples x' into the denoise model and get the output, instead of using the L_1 or L_2 distance between the denoised origin image x_d and adversarial examples x'_d, we use a extra classify model k to calculate the similarity of the images, we define $k_f(x)$ as the feature map before the classify model's last average pooling layer and use the feature map's L2 distance to calculate the loss function.

$$l_F(x, x') = L_1(k_f(x), k_f(x')) \tag{8}$$

Our model architecture is like the SRGAN, and show in Table 5.

We use two method to get better performance of our model. First, we train our model D with two kind of ad-

versarial examples, one is the gradient based adversarial examples, we choose the MI-FGSM and generate adversarial examples with attack eight models. Another is generate base adversarial examples, we use our non-attack model to generate adversarial examples. Second, we use ensemble of classify models to get the mean loss of $L_F x, x'$, this lead the denoise model to generate denoise images could be right classified by many different models, and this means the denoised images x_d and x'_d are more similar.

3.3. ALP

In the adversarial training part, we use the denoised origin images x_d and adversarial examples x'_d as the input pair to train our classification model T and we define $T(x)$ as the logist output of the model. Like ALP, we define two loss function.

$$L_{CE}(x) = cross\_entropy(x, label) \tag{9}$$

$$L_{ALP}(x, x') = L_1(T(x), T(x')) \tag{10}$$

and the final loss function is

$$L(x, x') = L_{CE}(x) + L_{CE}(x') + L_{ALP}(x, x') * \beta \tag{11}$$

where β is the weight factor of the ALP loss.

References

References