VE488 Project Report: Generating Adversarial Sample with G-ATN

Liu Hongyi Zhang Bohao Guo Lingyun Fang Yan

August 7, 2018

Abstract

In this project, we present an innovative method to generate adversarial samples for CNN image classifier. The idea is to train a model which takes the original image and a target label as inputs and output a new image with targeted misclassification label as well as a minimized difference on original features. With the information of gradients on original classifier as extra input, we effectively trained a neural network image generator. We tested the feasibility of our method on the mnist dataset with two different versions of the model on two different classifiers. The result verified that our method is effective. Our model can efficiently reduce the performance of both classifiers while preserving a high consistency with the original images. On the other hand, our model can mislead the classifier to generate a specific-targeted wrong prediction with high accuracy with a small executing time cost.

1 Introduction

Deep convolutional neural networks have made remarkable progress in the field of image recognition. Despite their fair performance on training set, they are vulnerable to adversarial attack and might fail to classify on images with small perturbation or modification (Akhtar and Mian 2018) [1]. In other words, if specific adjustments (which are trivial to human eyes) are made on images, the model may fail to provide a correct answer. Therefore, it is significant to generate training samples either to attack the model or to reinforce its learning ability. Training adversarial examples is an efficient way to achieve both goals.

Multiple approaches have been proposed on how to generate adversarial examples effectively. Usually, these methods focus on the direct manipulation of image pixels: (1)optimize a loss function on local generalization such as L-BFGS (Szegedy et al. 2014) [2], (2)alter a few pixels such as JSMA (Papernot et al. 2015) [3] and one pixel attack (Su et al. 2017) [4], (3) utilize gradient information of CNN to cheat the machine in oriented direction such as FGSM (Goodfellow et al. 2015) [5] and Houdini(Cisse et al. 2017) [6]. In this project, we propose a rather different approach to generate adversarial samples, namely, the Gradient Adversarial Transformation Network (G-ATN) model.

G-ATN is an advanced version established on the basis perturbation adversarial transformation network(ATN) (Baluja et al. 2017) [7]. Perturbation ATN is a targeted, white box adversarial attack algorithm which maps an original input $\mathbf{x}$ to a new adversarial example $g(\mathbf{x}) = \mathbf{x'}$ by introducing a perturbation on the residual block: $g(\mathbf{x}) = \tanh(\mathbf{x} + G(\mathbf{x}))$. Weighted by some small valued vectors, effective perturbations could be added to cause misclassification by machine while preserving the original feature of the image.

In our work, we wish to enhance the ability of ATN to generate adversarial samples by adding the gradient values of the network of the classifier with respect to original images as an input to the original ATN. With this additional gradient information (which has been applied in other adversarial attacks such as FGSM (Goodfellow et al. 2015) [5]), the attack of ATN could be more targeted and consequently, more effective. Combine the attack algorithm based on gradient and ATN together, we wish to improve the attack efficiency or optimization speed of the adversarial sample generator.

2 Gradient Adversarial Transformation Networks

2.1 Objective

The objective of a G-ATN is learning to transform a sample to an adversarial sample against a classifier taking both the original image and the gradient of the classifier with respect to the image as inputs. In our work we construct it as an targeted G-ATN since an untargeted G-ATN is just a special case of a targeted G-ATN.

Denote the image space as $\mathcal{X}$, the set of possible class as $\mathcal{K}$, the target class as t and the classifier we are attacking as $f : x \in \mathcal{X} \to y$ where y is a probability distribution on $\mathcal{K}$, a G-ATN is defined as following:

$$g_{f,\theta,t}(x, \nabla_x f_t(x)) : x \in \mathcal{X} \to x', \tag{1}$$

where $\nabla_t f(x)$ is the gradient of probability of target class with respect to the original image [7].

Denote $x' = g_{f,\theta,t}(x, \nabla_x f_t(x))$, similar to ATN, the training of G-ATN is to solve the optimization problem of:

$$\underset{\theta}{\operatorname{argmin}} \sum_{x \in \mathcal{X}} \{\beta L_{\mathcal{X}}(x', x) + L_y(f(x'), t)\}, \tag{2}$$

where the image reconstruction loss function $L_{\mathcal{X}}$ and adversarial attack loss function L_y are similarity measuring function on the image space and the output space. The target t is interpreted as an output distribution targeted to class t, where we use one-hot encoding in our work. Equation 2 serves as a loss function consists of two part. output space loss function L_y controls how well G-ATN can fool the classifier and the image space loss function L_X regularize the distortion on an image such that it's still trivial for human to recognize the original class of the image. Coefficient β controls the regularization intensity.

2.2 Network Architecture

A G-ATN consists of two parts: a gradient residual block that learns the perturbation and an image norm activation function that turns the image and the perturbation into a valid output in image space. Altogether, G-ATN output an adversarial sample directly against the classifier.

Gradient Residual Block To prevent the problem of gradient explosion/vanishing and optimize the performance, we use residual blocks in our network architecture [8], as shown in fig 1. An explicit residual block is used to generate perturbation based on gradient and the whole G-ATN is actually an implicit residual block used on original image for generating adversarial samples. Previous work explores various methods using gradient information for the adversarial attack, including targeted fast gradient sign method(T-FGSM), gradient iteration attack [9] and boosted gradient attack using momentum [10]. As discussed in the FGSM, simply learning sign function is powerful to generate adversarial examples, thus learning a residual mapping would possibly be even powerful during an adversarial attack. The residual block in the architecture shown in figure 1 could be cascaded for arbitrary times to construct a deeper network as we wish.

Image Norm Activation Function The object of this function is to make the output image a valid sample in image space while preserving the output as close as the output of the penult layer. In our

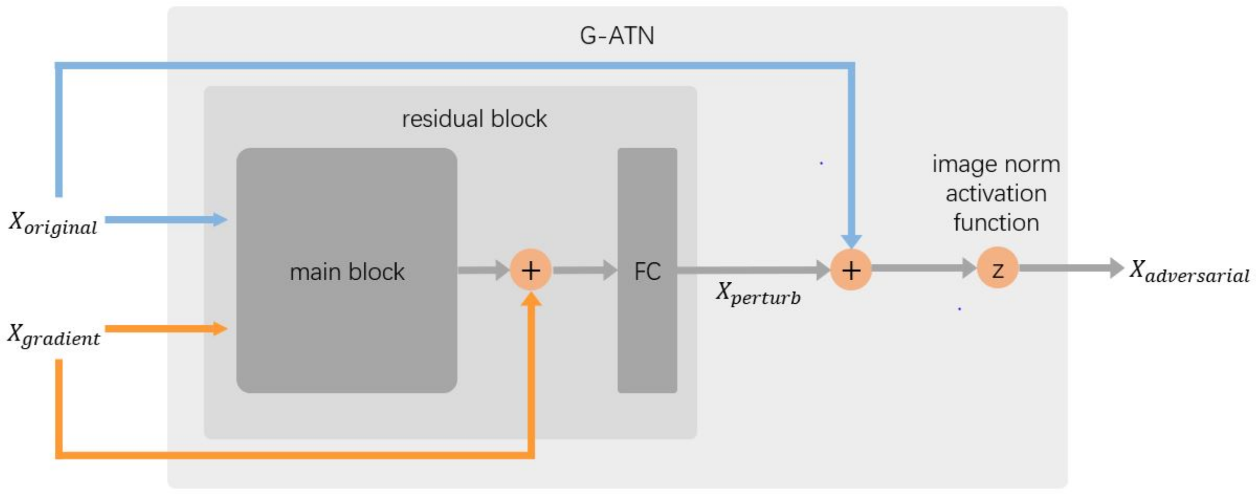


Figure 1: A general structure of G-ATN. Input an original image x and its gradient into G-ATN. After the main block, $x_{gradiant}$ is added so that the main block could learn a residual mapping. The residual block outputs the perturbation, then the perturbation is added to the original image and goes through the image norm activation function to output the adversarial sample.

representation, the image space consists of pixels on interval [0,1]. To make the adversarial sample a valid image in image space, the previously generated perturbation is added to the original image and go through a specially designed function $z(x) = \text{sigmoid}(5(x - 0.5))$. This special function automatically norms all numbers into the interval $[0, 1]$ and it preserves the major perturbation since $z(x)$ is very similar to the identity function on the interval of $[0, 1]$.

Structure Figure 1 shows the overview structure of a typical G-ATN. The gradient is added directly onto the output of the main block and goes into a final fully connected layer. This layer is used to prevent learning the identity function on gradient and to provide more diversity on generated adversarial samples. The output perturbation is then added with the original image and plugged into the image norm activation function. The major work of learning the G-ATN function $g_{f,\theta,t}(x, \nabla_x f_t(x)) = z(x + x_{perturb}) = z(x + p_{f,\theta,t}(x, \nabla_x f_t(x)))$ is implicitly done by learning the perturbation generating function $p_{f,\theta,t}$.

3 MNIST Experiment

3.1 Training Classifier and G-ATN

CNN Classifier We trained two different G-ATN against two different CNN on MNIST dataset, result in forty different targeted attacks considering that MNIST has 10 classes [11] [12]. The two classification models have a combination of fully connected linear layers (FC) and convolutional layers. In the whole training procedure, they differ only in the structure and hyper-parameters. This helps us to determine the generalization ability of G-ATN over different target classifiers. The training of classification model follows a traditional manner of softmax regression. The accuracy of classification model is shown in table 1.

G-ATN Structure The G-ATN also uses two different structure. A G-ATN$_{\text{FC}}$ uses only FC in the main block and a G-ATN$_{\text{conv}}$ uses convolutional layers to extract feature in both the original image and

Name	Architecture	Accuracy
CNN1	3×3 Conv→3×3 Conv→3×3 Conv→FC→FC	99.0%
CNN2	3×3 Conv→FC→FC→ FC	97.6%

Table 1: Architect and accuracy of classifier

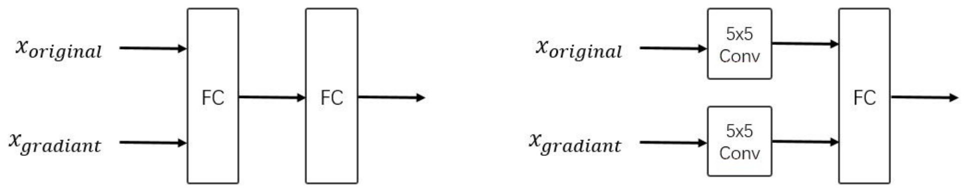


Figure 2: The main blocks of two different G-ATNs used to generate adversarial samples in MNIST experiment. The output of the main block flows then into the architecture shown in figure 1 to form the whole network. Left: G-ATN$_{\mathrm{FC}}$ that uses only FC layers. Right: G-ATNconv that use both a FC layer and a convolution layer.

the gradient. The structure of the main block of these two different G-ATN is shown in figure 2. The output of the main block then flows through the architecture shown in figure 1.

G-ATN Training Before training a G-ATN all classifiers are trained and fixed. The gradients with respect to target class are pre-calculated over the whole dataset. Weight updates are done by computing and minimizing the loss function described in equation 2. In this experiment we use an L2-norm for both the image space and output space, however more sophisticated loss function such as perceptual loss function on image space [13] or reranking loss function on output space are also possible [7]. To accelerate the convergence we use an Adam optimizer for parameter updating. To balance the regularization coefficient β in the loss function, we apply a self-adjust manner to β with predetermined thresholds. In other words, we increase β when the image distortion is too big and decrease β when the target rate is too high. The thresholds of these values are empirically determined. The training results hence become less sensitive to the initial value of β and the training becomes much easier.

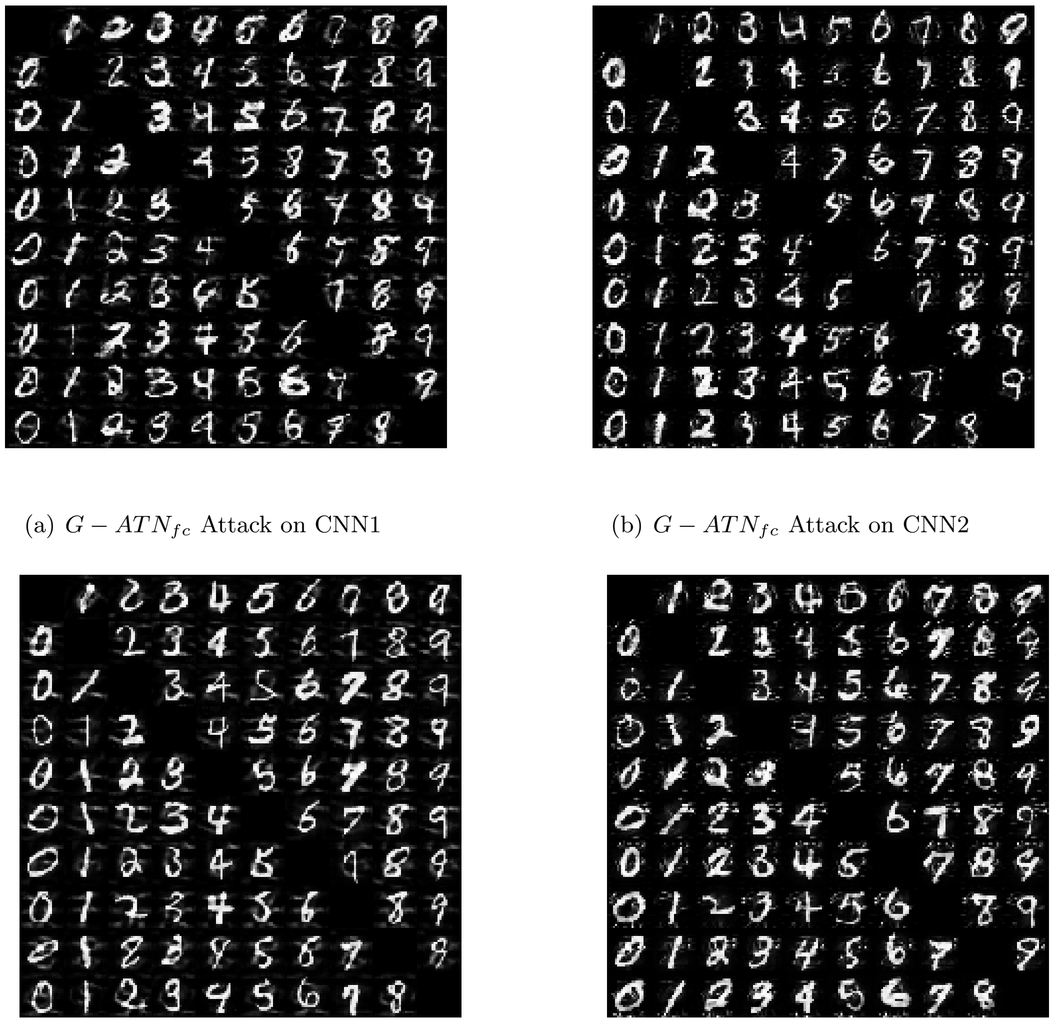
3.2 Results

ATN	Classifier	Average Accuracy	Average Target Rate	Average MSE (10^{-2})
G-ATN$_{fc}$	CNN1	26.02%	79.46%	1.2897
G-ATN$_{fc}$	CNN2	19.06%	87.56%	1.3078
G-ATN$_{conv}$	CNN1	16.76%	90.52%	1.6318
G-ATN$_{conv}$	CNN2	13.70%	94.63%	1.5979

Table 2: Evaluation of Attack Results

We first view the general result of G-ATN attack as shown in Table 2. The third column shows the average classification accuracy among the G-ATN with the 10 different target. The fourth column shows the average target rate among the different targets. And the last column shows the average MSE between the adversarial pictures and the original pictures among all the adversarial samples we generate.

From the results, we can see that the convolutional G-ATN is more powerful than the fully connected G-ATN. It decreases the average accuracy to a lower level and has a target rate that is larger than G-ATN$_{\mathrm{fc}}$. However, it also makes more changes to the original picture as shown by the average MSE. But these changes are still reasonable such that the adversarial samples can be classified correctly by human eyes as shown in figure 3. Moreover, the classifier CNN1 which has more complex architecture and more convolutional layers is slightly less affected by the G-ATN attack we generate.



(a) $G - ATN_{fc}$ Attack on CNN1

(b) $G - ATN_{fc}$ Attack on CNN2

(c) $G - ATN_{conv}$ Attack on CNN1

(d) $G - ATN_{conv}$ Attack on CNN2

Figure 3: Adversarial examples generated by the two kinds of G-ATN to attack two CNNs with different target labels. Images in the 0-9th column have true label 0-9 correspondingly and are classified correctly by the classifier before adding perturbation. Images in 0-9th row are classified falsely to the target class 0-9 correspondingly. For example, the image in the 0th row and the 5th column (start counting from 0th) shows an adversarial sample of digit 5 that is classified as 5 initially but classified as 0 after the attack.

In figure 3, each row represents the adversarial attack with a specific target on the test set of MNIST that were originally classified correctly by two CNN classifiers with different labels. And each column represents a successful attack with different target on images with the same label. For example, in the third row of the figure, all the images, which have already been classified correctly into their actual labels, are now classified into '2' after the attack in our G-ATN. For most of the images, our G-ATN is able to find a corresponding relatively reasonable adversarial image.

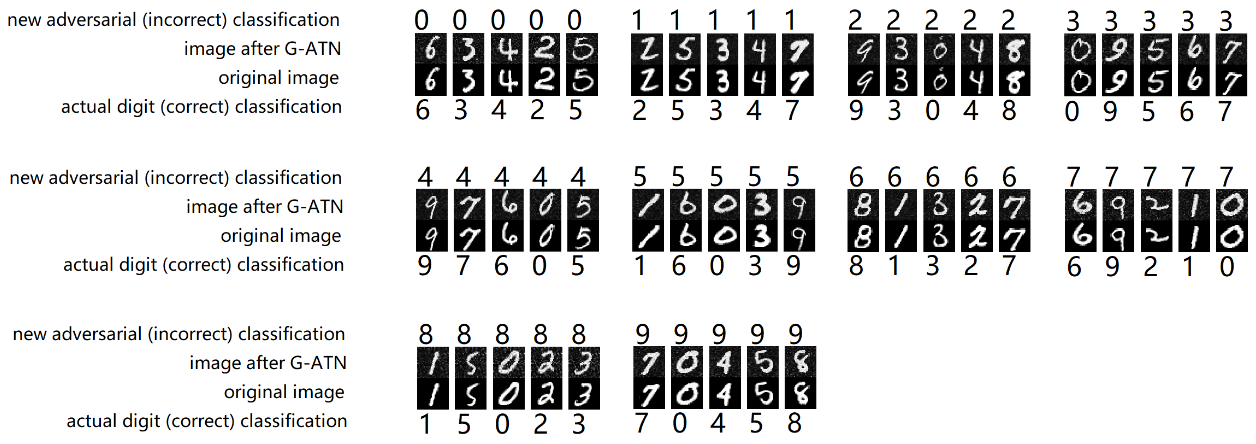


Figure 4: Examples of how pictures are changed when the convolutional G-ATN attacks CNN_{FC}. The first line represents the targets the G-ATN makes the classifier predicts. The second and third lines are the adversarial samples and the original pictures, respectively. And the last line is the true label, which the classifier correctly classified originally.

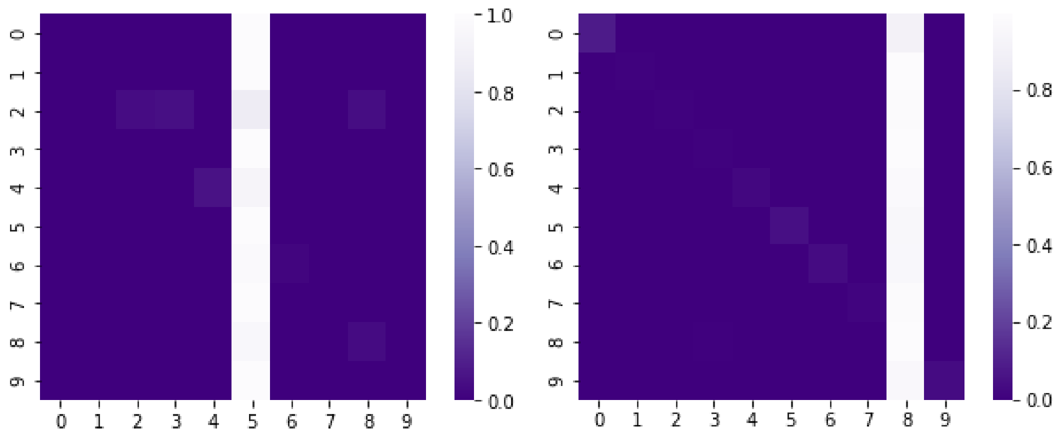


Figure 5: Sample demonstration of confusion matrix for target 5(left) and 8(eight) attacked by CNN_{conv} on CNN_2. The y-axis represents the original label of images and x-axis represents prediction made by classifier. As we can observe from the graph, for each column where the x-axis represents the target, the squares are significantly lighter than other squares of the matrices, which indicates the high probability of images having been successfully mispredicted as the target. This verifies the efficiency of G-ATN.

According to the crosswise comparison, the more sophisticated the ATN is and the simpler the CNN is, the average adversarial success rate and target rate will be higher. That indicates we can estimate and control the strength to attack by controlling the structure of ATN. To some extent, the deeper and wider the ATN is, the better it will perform on attacking.

In figure 4, a closer look on the input and output of ATN is provided. All of these images are selected from the test set of MNIST. A few spotlights should be noted.

- For a constrained input label and target label, We can clearly see that after training, the noise converges to a nearly constant value, only varying in one or two pixels for different images.
- By observation, we can clearly see that the network is trying to attack by erasing features that the target images don't have and add specific features that original images don't have. That indicates the network is able to capture different features in different digits and try to apply them during attacking.

In figure 5, two of the confusion matrices for each target are selected to present the overall performance of G-ATN. Each square represents a value in the confusion matrix. While the y-axis implies the original label of images, the x-axis represents prediction made by classifiers. As indicated by the color bar, the lighter the color of a square, the higher the percentage is for images (of one row) being classified to this label. From figure 5, we can clearly distinguish the target from the white column in the matrix. Some other squares whose color are slightly lighter than the color level at 0.0 indicates failed cases of the attack. The confusion matrices confirm the overall performance of G-ATN.

3.3 Comparison with other algorithms

3.3.1 Comparison with gradient based algorithms

To evaluate our algorithm, we evaluate the attack efficiency with other state-or-art gradient-based adversarial attack algorithms. We first introduce these algorithms below.

Fast Gradient Sign Method A simple but useful way of generating adversarial samples original proposed by Goodfellow in 2015 [14] called the fast gradient sign method (FGSM) is described by:

$$x_{adv} = x + \varepsilon \text{sign} \nabla_x J(x, y_{true}) \tag{3}$$

where function $J(x, y_{true})$ is the classification loss function and ε here controls the image reconstruction loss. This is a very fast method for calculation while the attack success rate is generally not so high.

One Step Target Class Method The one step target class method (OSTSM) is a targeted version of FGSM, where the gradient is calculated with respect to the target label instead of true label [9]. Using the same notation above, the algorithm is described as:

$$x_{adv} = x - \varepsilon \text{sign} \nabla_x J(x, y_{target}) \tag{4}$$

Sometimes we use the least likely class(y_{LL}) as a target class where $y_{LL} = \arg\min_y p(y|x)$.

Basic Iterative Method Basic iterative method(BIM) performs FGSM in an iterative way with an additional clipping procedure to control the image reconstruction loss:

$$x^0 = x, \quad x_{adv}^{t+1} = \text{Clip}_{X,\varepsilon}\{x_{adv}^t + \alpha \text{sign} \nabla_x J(x, y_{true})\} \tag{5}$$

Model	Overall accuracy	Non-target accuracy	Target rate	$L_{\mathcal{X}}(\times 10^{-2})$	Time cost
G-ATN$_{FC}$	35.3%	27.9%	69.9%	**1.07**	**167ms**
G-ATN$_{Conv}$	15.6%	**4.3%**	**92.4%**	1.26	**186ms**
FGSM	44.1%	43.8%	/	2.04	**164ms**
OSTCM	58.8%	54.1%	20.0%	2.04	**160ms**
BIM	**4.7%**	4.9%	/	1.31	14.1s
TBIM	27.2%	18.8%	77.0%	1.33	14.3s

Table 3: Comparison of gradient based algorithms with G-ATN. The overall accuracy is evaluated over whole testing set while non-target accuracy is evaluated excluding all target labeled samples. The image reconstruction loss $L_{\mathcal{X}}$ here represents the pixelwise mean square error. The evaluation is done over classifier CNN1 with target 7. Bold shows the best model(s) in each column index. The target rates of non-target methods are filled with a slash

where the clipping procedure follows element-wise clipping, mapping $\text{Clip}_{X,\varepsilon}(A)$ into the interval $[X - \varepsilon, X + \varepsilon]$. With a small step controlled by α in each iteration and the final reconstruction loss controlled by ε, BIM provides a more powerful attack compared with FSGM. In our experiment, the parameters are set according to the following discussion done by Kurakin [15] [9].

Target Basic Iteration Method Target basic iteration method(TBIM) is a straightforward extension over BIM:

$$x^0 = x, \quad x_{adv}^{t+1} = \text{Clip}_{X,\varepsilon}\{x_{adv}^t - \alpha \text{sign} \nabla_x J(x, y_{target})\} \tag{6}$$

Same as BIM, TBIM provides more powerful attack than OSTCM. In our experiment, the parameters of TBIM are set the same as in BIM.

We tested these algorithm together with our G-ATN over the accuracy, the target rate, the image reconstruction loss and the time cost. The result are shown in table 3. We notice that G-ATN performs effectively as expected.

- Among all target method, $G - ATN_{conv}$ produces the highest target rate and the lowest non-target accuracy while making a second lowest image reconstruction loss. Performance of G-ATN exceeds far beyond two fast methods and also exceeds the two iterative methods considering it is a target method. This confirms that G-ATN learns a nontrivial transformation based on the gradient.
- While implementing the attack, most of the computation resources are spent on gradient calculation. G-ATN and fast method cost nearly the same time since they only require the gradient to be calculated only once while iterative methods are considerably slower. During practical application, those require a faster method to implement. G-ATN serves as an alternative way for many other gradient-based algorithms.
- Considering the distortion on the image, we may notice that controlling the image reconstruction loss of a G-ATN is slightly harder than simple gradient-based methods. We may need to carefully adjust the weighting parameter β to balance the performance between image reconstruction loss and the attack loss. In some cases, the optimum interval of β may be very narrow and the model may be hard to train. This is a disadvantage of G-ATN compared with simple gradient-based algorithms.

3.3.2 Comparison with other network based adversarial attack

In this part, we compared our method with the original ATN [7] and another network that produces adversarial samples proposed by Xiao et al. in 2018 called advGAN [16]. The last method uses a GAN to generate adversarial sample both in black-box and white-box attack in a manner similar to our training. Here we only compare the results in the white-box attack on MNIST. We must notice that the classifier

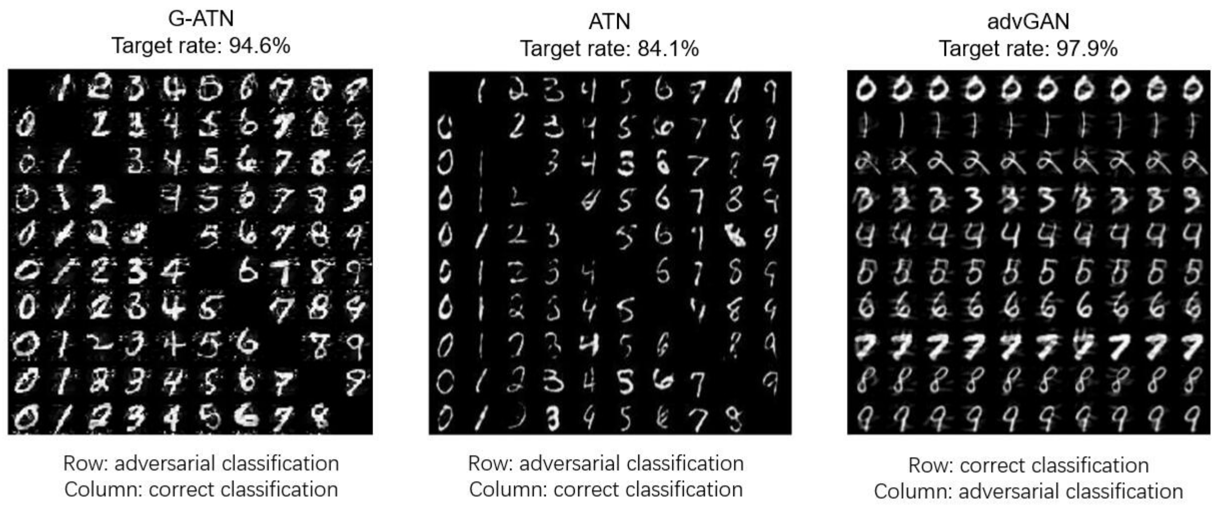

G-ATN
Target rate: 94.6%
Row: adversarial classification
Column: correct classification
ATN
Target rate: 84.1%
Row: adversarial classification
Column: correct classification
advGAN
Target rate: 97.9%
Row: correct classification
Column: adversarial classification

Figure 6: Comparison of adversarial samples generated by G-ATN, ATN, and advGAN [7] [16]. Note this is only a qualitative comparison since the classifierS differ in each case.

0Swe attack are different and the image reconstruction loss of these two work are also not found as well, hence the comparison lies only in a qualitative way. The result is shown in figure 6.

In all three cases the network shows its ability to attack the classifier. However, we could see an interesting connection between the architectural features of adversarial samples. Our G-ATN learns to add noises in a pixel-wise manner, which is similar to pixel-based methods such as one-pixel attack [4]. This means G-ATN successful learn how to use and modify the gradient information to fool the classifier. In original ATN, the network tends to blur the original image. This is probably caused by the training objects that ATN learns to hide and modify the important features of each digit. The adversarial samples generated by advGAN consists of a lot of "stroke-like" lines that seem to recreate the targeted digit, which is reasonable since the generator in a GAN wishes to create images that are similar to original samples in the dataset. This comparison shows we may be able to control the features generated by the network by designing specific architecture.

4 Related Discussion

The architectural and training procedure of G-ATN is associated with many other related works. In this section, we dive deeper to explore the connection of G-ATN with artistic transformation and GAN with further insight.

Image Transformation Network in Artistic Transfer The object of G-ATN could be classified as a more general task of feed-forward image transformation and this task introduces a lot of similarities on network architecture. A previously proposed image artistic transform method [13] trained an image transformation network based on a pre-trained classifier by minimizing the total loss consists of the image reconstruction loss and the style reconstruction loss. Both the objective of adversarial and artistic transfer concerns realizing a specific task while preserving the information of the original image to some extent, where the loss function is evaluated with the help of a pre-trained network and the error is propagated back to the transformation network. Our work could be pushed even further by looking at these related networks.

Generative Adversarial Network In a typical Generative Adversarial Network [17], a generator is used to fool a discriminator and the training is done by optimizing a minimum maximum problem. In our work, we implement the adversarial part using a transformer instead of a generator to attack a pre-trained classifier. A transformer is much easier to train and more targeted. However, a generator could provide more diversity on generated samples.

5 Conclusion

In our work, we successful implemented G-ATN and enhanced the adversarial attack based on ATN. The result shows that G-ATN learns a nontrivial function that generates powerful adversarial samples using the information of the gradient. This provides a powerful adversarial attack algorithm based on existing methods. Comparing with ATN or other gradient-based attack algorithms, our method needs more information and computational resource on training, however producing a more powerful attack with a very low computation cost during execution. Our G-ATN is able to serve as a useful method for evaluating the robustness of networks.

In future, we may explore the possibility mainly in three directions: G-ATN facilitates adversarial training, black-box attack, and improvement in the architecture of G-ATN.

- Previous work had shown that training with specific adversarial examples could enhance the robustness over particular attack algorithm [14]. Training with network generated adversarial samples had already been done by Lee et al. in 2015 [18] using a generative adversarial network. We expect that G-ATN-facilitated training will enhance the robustness of the network, especially against a gradient-based attack.
- Implementing G-ATN during a black-box attack is also possible. Work done by Papernot et al. shows that using a substitution network we could approximate the target network well. Training of the G-ATN could be built based on such substitution network.
- The improvement in architecture lies mainly in the loss function of a G-ATN. In our work, we use the most straightforward pixel-wise mean square error as an image reconstruction loss. Perceptual loss [13] or other loss function based on higher features may introduce more diversity into adversarial samples.

References

[1] N. Akhtar and A. S. Mian, “Threat of adversarial attacks on deep learning in computer vision: A survey,” *IEEE Access*, vol. 6, pp. 14 410–14 430, 2018.

[2] Szegedy, Zaremba, Sutskever, Bruna, Erhan, Goodfellow, and Fergus, “Intriguing properties of neural networks.” *arXiv: Neural and Evolutionary Computing*, 2014.

[3] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, “The limitations of deep learning in adversarial settings.” *In Proceedings of IEEE European Symposium on Security and Privacy*, 2015.

[4] J. Su, D. V. Vargas, and S. Kouichi, “One pixel attack for fooling deep neural networks.” *arXiv preprint*, 2017.

[5] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” *arXiv preprint*, 2015.

[6] M. Cisse, Y. Adi, N. Neverova, and J. Keshet, “Houdini: Fooling deep structured prediction models,” *arXiv preprint*, 2017.

[7] S. Baluja and I. Fischer, “Adversarial transformation networks: Learning to generate adversarial examples.” *arXiv: Neural and Evolutionary Computing*, 2017.

[8] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *computer vision and pattern recognition*, pp. 770–778, 2016.

[9] A. Kurakin, I. J. Goodfellow, and S. Bengio, “Adversarial machine learning at scale,” *international conference on learning representations*, 2017.

[10] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li, “Boosting adversarial attacks with momentum,” *computer vision and pattern recognition*, 2018.

[11] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, “Automatic differentiation in pytorch,” 2017.

[12] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[13] J. Johnson, A. Alahi, and L. Feifei, “Perceptual losses for real-time style transfer and super-resolution,” *european conference on computer vision*, pp. 694–711, 2016.

[14] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Experiencing and harnessing adversarial examples,” *International Conference on Learning Representations*, 2015.

[15] A. Kurakin, I. J. Goodfellow, and S. Bengio, “Adversarial examples in the physical world,” *arXiv: Computer Vision and Pattern Recognition*, 2017.

[16] X. C. Chaowei, B. Li, J.-Y. Zhu, W. He, M. Li, and D. Song, “Generating adversarial examples with adversatial networks,” *arXiv: Cryptography and Security*, 2018.

[17] I. J. Goodfellow, J. Pougetabadie, M. Mirza, B. Xu, D. Wardefarley, S. Ozair, A. C. Courville, and Y. Bengio, “Generative adversarial networks.” *arXiv: Machine Learning*, 2014.

[18] H. Lee, S. Han, and J. Lee, “Generative adversarial trainer: Defense to adversarial perturbations with gan.” *arXiv: Learning*, 2017.