Adversarial Attack using Neural Image Modification

This paper was downloaded from TechRxiv (<https://www.techrxiv.org>).

LICENSE

CC BY 4.0

SUBMISSION DATE / POSTED DATE

07-01-2022 / 25-01-2022

CITATION

Lana, Jandrik (2022): Adversarial Attack using Neural Image Modification. TechRxiv. Preprint.
<https://doi.org/10.36227/techrxiv.18012122.v3>

DOI

[10.36227/techrxiv.18012122.v3](https://doi.org/10.36227/techrxiv.18012122.v3)

Adversarial Attack using Neural Image Modification

Jandrik Lana

January 17, 2022

Abstract

The increasing capacity and capability of neural networks over the years have made dependence on neural networks to increase. Along with this came methods to cause neural networks to perform ineffectively and produce mistakes, called adversarial attacks. These adversarial attacks can negatively affect neural network based security systems, bypass networks that are used to constrict access, and similar scenarios where neural networks are applied. In detail, these attacks can cause a target model to perform with low accuracy, or can cause the target model to output a certain output, called a targeted attack. In order to help development into analyzing the characteristics of adversarial sample generation, this work proposes an adversarial attack that utilizes neural image modification to generate an adversarial sample. This method proves to be effective in reducing a target network’s accuracy in both untargeted and targeted attacks with good success rates. This method also shows some effectiveness against defensive distillation, but not transferrable between multiple models.

1 Introduction

Over the years, neural networks have been increasing in function and capabilities. These include breakthroughs in image classification [1], object detection [2], image generation [3], natural language processing [4], security [5], and many more. However, these neural networks are susceptible to adversarial samples, which are clean samples modified to decrease the performance of a neural network. These almost imperceptible modifications are commonly called perturbations, whose applications can be used to maliciously compromise neural networks involved in security purposes. Many defenses have been proposed to develop more robust and secure neural networks, such as defensive distillation [6], determining IPT (image processing technique) sequences [7], training models to be efficient with corrupted inputs [8]. Even so, different types and approaches to generating adversarial samples which evade these defenses have evolved in recent times. Adversarial attacks can be categorized into “white box attacks” and “black box attacks”. White box attacks use knowledge of the model (i.e., the network architecture, model’s gradients, etc.) to generate adversarial samples. This involves access to the whole model. These attacks are predominantly demonstrated in image classification models [9]. Black box attacks on the other hand generate adversarial samples without any or limited knowledge of a specific model. This may be done through thorough analysis of the inputs and outputs of the model. Consequently, this can be used to fool various models with different architectures, termed the transferability of an adversarial attack [10]. This study proposes a novel framework that utilizes neural image editing and convolutional neural networks for generating adversarial samples. Unlike most frameworks that utilize generative adversarial networks (GANs) [11] using three components (a discriminator, generator, and a target network), the proposed framework uses only two components: a convolutional neural network (CNN) and a target network. Along with the capability of similar frameworks to generate adversarial samples even after access to the model is removed, the proposed framework can be modified to either decrease the performance of the target network or cause the target network to output a chosen class (targeted attack).

2 Related Work

In this section, threat models of various adversarial attacks will be outlined in comparison to the proposed attack.

2.1 Fast Gradient Sign Method (FGSM)

FGSM, developed by Goodfellow, et. al. (2014), uses the sign of the gradients to produce an adversarial sample. The gradient of the loss with respect to the input image is used. The update rule is as follows:

$$x^{adv} = x + \varepsilon (\nabla_x J(x, y_{true})) \quad (1)$$

where x is the input image, x^{adv} is the adversarial sample, J is the loss function, ∇_x represents the gradient with respect to x , and ε is a constant that confines the perturbations to a small amount. In the case of FGSM, instead of iterating, it performs the attack in a single step [12].

2.2 Carlini Wagner Attack

The Carlini Wagner Attack (CW) [13] attempts to minimize the accuracy of the model with the constraint of minimizing the Euclidean distance (L_2 norm) between the generated adversarial sample and the clean sample for each class. The adversarial sample having the lowest L_2 norm is used. Below is the process of the attack, with w determining the strength of perturbations and a loss function f [12]:

$$\text{minimize } \|\frac{1}{2}(\tanh(w) + 1) - x\|_2^2 + c \bullet f(\frac{1}{2}(\tanh(w) + 1)) \quad (2)$$

Similar to this study's proposed architecture, neural networks also have been used recently to generate adversarial samples, several of which are outlined below:

2.3 AdvGAN

AdvGAN is a generative adversarial network consisting of a generator, a discriminator, and a target neural network developed by [14]. This generates adversarial samples starting with passing the original image into the generator. The generator output is added to the original image, which is then inputted into the discriminator. The discriminator is tasked with classifying between adversarial samples and clean samples. They described the loss of the adversarial network targeting a class t is as follows using the original loss ℓ_f used to train the target neural network:

$$\mathcal{L}_{adv}^f = E_x \ell_f(x + G(x), t) \quad (3)$$

2.4 GAN Adversarial Sample Generation targeting speech

D. Wang et al. [15] procured a GAN capable of generating adversarial speech samples for a speech recognition model. These adversarial samples are crafted to induce the model to output a particular class, whilst preserving quality. Their architecture consists of a generator, discriminator, and a target model, similar to the previous architecture above. The generator produces a perturbation to be added to the original input. This will be the adversarial sample to be inputted into the discriminator and the target model. The following loss function is used in their paper:

$$L_G = L_{adv}^f + \alpha L_{fool} + \beta L_{hinge} + \gamma L_2 \quad (4)$$

L_{adv}^f measures the loss of the generator in attacking the target model, and L_{fool} measures the loss of the generator in fooling the discriminator. To regularize the model, the hinge loss L_{hinge} and L_2 norm loss are added.

2.5 Adversarial Sample generation using Convolutional Neural Network

Qiu, et. al. [16] developed a convolutional neural network that takes a random noise as input and extracts features from it to form perturbations, which are added (multiplied by a constant η to scale the perturbation) to a clean image to fool a classifier. The generator loss is composed of two separate losses, one which minimizes the perturbation, and another which is the mean of the difference between the classifier prediction and a target prediction. This is shown below:

$$l_{total} = \text{Maximum}(\alpha l_{diss}, \beta l_{pred}) \quad (5)$$

2.6 In comparison to the proposed framework

Unlike FGSM and the CW that attempts to solve optimization problems, the proposed framework uses a neural network to generate adversarial samples, identical to the last two methods above. The proposed framework has an objective analogous to the loss function of the CW. Instead of having a generator, discriminator, and a target network, only a generator (CNN) and a target network will be used, therefore only needing to train one network. Regarding the work of Qiu, et. al., one main difference between this proposed method and theirs is the generator architecture, which is composed purely of convolutional layers. Another difference is the generation of adversarial samples, wherein their generator takes in a noise input, while this proposed method takes in the input image itself. In addition, unlike the work of Qiu, et. al., the degree of perturbation in this work is controlled through penalization of a term in the formulated loss function.

3 Attack framework and Neural Image Modification Network

3.1 Defining the problem

The target of the generator $G(x)$ with an input image x is to lower performance of the target network $F(x)$. To achieve this, the generator's loss function must be defined in order for backpropagation to optimize the generator. It is important that the output of the loss function does not deviate from the original input image, yet at the same time causes the target network to predict wrong outputs. Two terms can be defined: one which maximizes the target networks loss, and another which minimizes the differences between the input image and the adversarial image.

3.2 Defining the loss function of the generator

To satisfy the requirements outlined in section 3.1, the loss function of the generator is defined as follows:

$$L_G = \frac{\gamma}{L_T} + L_2(\hat{Y}_{Gi} - x) \quad (6)$$

where $\hat{Y}_{Gi}$ is the output adversarial images of the generator equal to $G(x)$, x being the input images, L_G is the generator's loss function, L_2 being the L_2 norm, and L_T being the categorical cross entropy loss defined as:

$$CCE = -\frac{1}{m} \sum_{i=1}^m \sum_{j=1}^n (Y_{ij} \log(\hat{Y}_{ij})) \quad (7)$$

with m being the amount of training samples, n is the number of classes, Y_{ij} is the label of the input image of class j , sample i , and $\hat{Y}_{ij}$ is the prediction of the target network given input image of class j , sample i , equal to $F(x)$. A hyperparameter γ in Equation 6 penalizes the first term resulting in modulating the strength of the perturbations.

Minimizing this loss function L_G results in the target network having decreased accuracy. An indicator of good performance can be a low accuracy obtained by the target network on the generated adversarial samples, and a success rate defined as the ratio of incorrect predictions to the total number of input samples.

For a targeted attack, the loss function above can be changed, replacing the term $\frac{\gamma}{L_T}$ with the categorical cross entropy loss CCE , except that the labels are the same across all training samples. The labels will be replaced with the target class Y_t . The loss function is now:

$$L_G = \gamma(CCE(\mathbf{Y}_t, \hat{\mathbf{Y}}_{ij})) + L_2(\hat{Y}_{Gi} - x) \quad (8)$$

Minimizing this forces the target network to output a chosen class. The gradients of the generator can be then represented simply as

$$\frac{\partial L_G}{\partial \theta_{gen}} = \frac{\partial L_G}{\partial L_T} \frac{\partial L_T}{\partial \theta_{targ}} \frac{\partial \theta_{targ}}{\partial \theta_{gen}} \quad (9)$$

with θ_{targ} and θ_{gen} being the parameters of the target network and generator respectively.

Figure 1: Proposed framework for generating adversarial samples

3.3 Attack Framework

The framework requires two neural networks: the generator, and the target network. The target network is a pre-trained classifier whose parameters will not be update during generator training. Input images will go through the generator and its output will be inputted into the target network. Given the target network’s output and the generator’s output, the loss function of the generator is computed, then its gradients, then backpropagated. Figure 1 shows the diagram for the framework.

3.4 Experimental Setup and Network Architectures

Two target networks, one trained on the Fashion-MNIST dataset [17] and another on the CIFAR-10 dataset [18] will be used to evaluate the performance of the generator on targeted and non-targeted attacks.

Each generator has four convolutional layers (see Figure 2), including image padding so that each output preserves the original dimensions. Before adversarial training, each generator will be pretrained to copy its input on their respective datasets. This is done in order to speed up training time. Afterwards, the two target networks will be trained, using a softmax layer for the output layer.

Training lasts for 30 epochs using Adam (Adaptive Moment Estimation) and SGD (stochastic gradient descent) with a momentum of 0.9 for backpropagation. Each has a learning rate of 0.001. The performance of the target networks in terms of categorical accuracy on the generator’s adversarial images and the success rates of the generator will be recorded. To observe the effect of various values of γ , γ values ranging from 0.001 to 0.1 will be tested in untargeted attacks. Good indicators of performance of the generator can be a low target network accuracy (near 0%) and a high success rate (near 100%). For the targeted attacks, the mean probabilities of each output from the softmax layer will be analyzed, in order to observe how well the generator causes the target network to increase its confidence in outputting a target class. If the mean probability output corresponding to the target class is high (near 100%), this implies a good performance of the generator, in addition to the two aforementioned indicators.

To evaluate the performance of the attack under various conditions, untargeted and targeted attacks will be tested against a model with a different architecture, and a model trained using defensive distillation. Defensive distillation is an adversarial defense mechanism that uses soft labels instead of hard labels, which are acquired from training another model, letting the model learn from information relative between classes [6].

4 Experimental Results

It is observed in both targeted and untargeted attacks that higher values of γ decrease the target network’s accuracy, however this also causes the perturbations to be more noticeable. Below shows the performance of the generators on the Fashion-MNIST dataset and the CIFAR-10 dataset on various values of γ . As a basis, the target networks achieve an accuracy of 90.23% on the fashion-MNIST dataset, and 78.74% on the CIFAR-10 dataset.

Figure 2: Generator Architecture for Fashion-MNIST (left) and CIFAR-10 (right)

Table 1: Attack summary (fashion-MNIST; untargeted; Adam optimizer)

λ	Acc. on adversarial samples	Generator Success Rate
0.001	77.85%	22.15%
0.01	67.25%	32.75%
0.02	53.32%	46.68%
0.03	51.17%	48.83%
0.04	41.37%	58.63%
0.05	40.85%	59.15%
0.06	40.02%	59.98%
0.07	19.50%	81.50%
0.08	18.50%	82.50%
0.09	19.65%	81.35%
0.1	18.01%	82.99%

Figure 3: Target network accuracy vs. γ (untargeted attack with fashion-MNIST dataset; Adam optimizer)

4.1 Untargeted Attacks

4.1.1 Untargeted attack using the fashion-MNIST

Table 1 and Figure 5 indicate that the network effectively decreases the target network’s accuracy to a minimum of 18.01% and a max success rate of 82.99% with unnoticeable perturbations.

4.1.2 Untargeted attack using CIFAR-10

The adversarial images produced by the generator on the CIFAR-10 dataset contain an issue. At $\lambda \leq 0.04$, the generator proceeds to blur the image (Figure 8). In Table 2, the generator decreases the target network’s accuracy to a minimum of 25.80% and a max success rate of 74.20% disregarding blurred images, with highly visible perturbations.

Figure 4: Generator success rate vs. γ (untargeted attack with fashion-MNIST dataset; Adam optimizer)

Figure 5: Adversarial Images for each gamma (fashion-MNIST; untargeted; Adam optimizer)

Table 2: Attack summary (CIFAR-10; untargeted; SGD optimizer)

γ	Acc. on Adversarial samples	Generator Success Rate
0.001	53.80%	47.20%
0.01	34.47%	65.53%
0.02	29.52%	70.48%
0.03	27.05%	72.95%
0.04	25.80%	74.20%
0.05	19.60%	80.40%
0.06	18.30%	81.70%
0.07	17.75%	82.25%
0.08	17.10%	82.90%
0.09	16.43%	83.56%
0.1	16.15%	83.85%

Figure 6: Target network accuracy vs. gamma (untargeted attack with CIFAR-10 dataset (SGD optimizer))

Figure 7: Generator success rate vs. γ (untargeted attack with CIFAR-10 dataset; SGD optimizer)

4.2 Targeted Attacks

For targeted attacks, each class in the fashion-MNIST and CIFAR-10 datasets were targeted. Adam was used as the optimizer and $\gamma = 0.01$ for the attacks.

4.2.1 Targeted attack using fashion-MNIST

Plotting the mean output corresponding to the target class of the softmax layer in Figure 10, the generator is shown to perform as expected, ranging from 0.77 to 0.94, maximum success rate of 87.48% (Table 3). Standard deviations range from 0.147 to 0.290.

4.2.2 Targeted Attack using CIFAR-10

In this attack, the generator achieved a max success rate of 89.71%, and a range of 0.95 to 0.99 mean output corresponding to the target classes (Standard deviations range from 0.0016 to 0.0580; Table 4).

Table 3: Attack summary (fashion-MNIST; targeted)

Class	Accuracy on Adversarial Samples	Generator Success Rate	Mean Softmax output of class
Class 0	12.52%	87.48%	0.94
Class 1	16.39%	83.61%	0.86
Class 2	15.74%	84.26%	0.86
Class 3	16.93%	83.07%	0.82
Class 4	16.04%	83.96%	0.89
Class 5	19.77%	80.23%	0.8
Class 6	12.59%	87.41%	0.93
Class 7	18.82%	81.18%	0.77
Class 8	13.40%	86.60%	0.94
Class 9	14.45%	85.55%	0.9

Figure 8: Adversarial Images for each gamma (CIFAR-10; untargeted; SGD optimizer)

Figure 9: Success Rates for each class (targeted attack with fashion-MNIST)

Figure 10: Mean softmax output corresponding to each target class (targeted attack with fashion-MNIST dataset)

Figure 11: Adversarial Images targeting each class in fashion-MNIST)

Table 4: Attack summary (CIFAR-10; targeted)

Class	Accuracy on Adversarial Samples	Generator Success Rate	Mean Softmax output of class
Class 0	10.64%	89.36%	0.97
Class 1	10.81%	89.19%	0.96
Class 2	10.60%	89.40%	0.98
Class 3	11.42%	88.58%	0.96
Class 4	10.51%	89.49%	0.98
Class 5	10.29%	89.71%	0.99
Class 6	10.80%	89.20%	0.97
Class 7	10.83%	89.17%	0.98
Class 8	11.15%	88.85%	0.95
Class 9	10.66%	89.34%	0.98

Figure 12: Success Rates for each class (targeted attack with CIFAR-10)

Figure 13: Mean softmax output corresponding to each target class (targeted attack with CIFAR-10)

Figure 14: Adversarial Images targeting each class in CIFAR-10)

Table 5: Comparison of success rates between different target networks

	Original Target Network	Diff. Architecture	Distilled Model
Success Rate (Untargeted)	89.15%	32.38%	84.48%
Success Rate (Targeted)	88.25%	31.23%	71.55%

4.3 Testing the Attack on various conditions

To evaluate the effectiveness of the attack under different conditions, particularly on a model with different architecture and a distilled model ($\alpha = 0.1$, with a temperature of 10) with the same architecture, both on the CIFAR-10 dataset. A generator was trained on the same target network above to test the untargeted attack, with $\gamma = 0.09$. Another generator was trained for the targeted attack with $\gamma = 0.02$, targeting the class “truck”.

The low success rates attained by the generator on the model with a different architecture shows that the generator’s attack is not transferrable between models, hence not applicable for black box attacks. Compared to the success rates of both untargeted and targeted attacks, the distilled model shows a 5.5% and 18.9% decrease in success rate, respectively.

4.4 Experiment Summary

The generators were shown to be effective in fooling the target networks. Maximum success rates of 83% and 74% (84% if including blurred instances, which may be attributed to the model reaching a saddle point or local optima on the loss function) were achieved for the fashion-MNIST and CIFAR-10 datasets, respectively. For the targeted attacks, they demonstrated an average of 84% and 89% success rates on the fashion-MNIST and CIFAR-10 datasets respectively.

The attack can be considered as a white-box attack, since an attack on a target network with a different architecture is shown to be of little effect. However, on a defensively distilled model with the same architec-

Figure 15: Targeted Attack on various conditions: Mean of each softmax output for each class compared with the original target network, one with a different architecture, and a distilled model.

ture, the attack is shown to have a 6% and 19% decrease in success rate for untargeted and targeted attacks, respectively.

5 Conclusion

This study proposes a framework for generating adversarial samples, which attempts to utilize a convolutional neural network as a method of neural image modification to modify an image into an adversarial sample. It does not use any dimensionality reduction, in order to retain the fine features of the input image. While this method of attack successfully fools a target network, it is not transferrable between models, and is robust to some degree against defensive distillation. This attack can therefore be classified as a white-box attack. Once trained, however, the generator can be used to generate adversarial samples independent of information on the target network. Regarding recommendations for future research, tests could be made on different convolutional generator architectures. It is also suggested that the generator be trained along with a ensemble of target networks, in order to develop transferability. In addition, the generator can be made to only modify a small specific part of an image (which can be specified or learned by the model) in order to minimize perturbations.

References

- [1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.
- [2] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.
- [3] Xiaolong Wang and Abhinav Gupta. Generative image modeling using style and structure adversarial networks. In *European conference on computer vision*, pages 318–335. Springer, 2016.
- [4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.
- [5] Min-Joo Kang and Je-Won Kang. Intrusion detection system using deep neural network for in-vehicle network security. *PloS one*, 11(6):e0155781, 2016.
- [6] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE symposium on security and privacy (SP)*, pages 582–597. IEEE, 2016.
- [7] Kishor Datta Gupta, Dipankar Dasgupta, and Zahid Akhtar. Determining sequence of image processing technique (ipt) to detect adversarial attacks. *SN Computer Science*, 2(5):1–20, 2021.

- [8] Ofer Dekel, Ohad Shamir, and Lin Xiao. Learning to classify with missing and corrupted features. *Machine learning*, 81(2):149–178, 2010.
- [9] Hongying Liu, Zhenyu Zhou, Fanhua Shang, Xiaoyu Qi, Yuanyuan Liu, and Licheng Jiao. Boosting gradient for white-box adversarial attacks. *arXiv preprint arXiv:2010.10712*, 2020.
- [10] Siddhant Bhambri, Sumanyu Muku, Avinash Tulasi, and Arun Balaji Buduru. A survey of black-box adversarial attacks on computer vision models. *arXiv preprint arXiv:1912.01667*, 2019.
- [11] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples, 2015.
- [12] Austin Short, Trevor La Pay, and Apurva Gandhi. Defending against adversarial examples. Technical report, Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), 2019.
- [13] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57, 2017.
- [14] Chaowei Xiao, Bo Li, Jun-Yan Zhu, Warren He, Mingyan Liu, and Dawn Song. Generating adversarial examples with adversarial networks. *arXiv preprint arXiv:1801.02610*, 2018.
- [15] Donghua Wang, Li Dong, Rangding Wang, Diqun Yan, and Jie Wang. Targeted speech adversarial example generation with generative adversarial network. *IEEE Access*, 8:124503–124513, 2020.
- [16] Zhongxi Qiu, Xiaofeng He, Lingna Chen, Hualing Liu, and LianPeng Zuo. Generating adversarial samples with convolutional neural network. In *Proceedings of the 2019 the International Conference on Pattern Recognition and Artificial Intelligence*, pages 41–45, 2019.
- [17] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.
- [18] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.