

Research on attack and defense algorithms for adversarial samples

1. Introduction

1.1. Research objectives

This project aims at the problem that neural networks are vulnerable to adversarial sample attacks and have security risks, and studies neural network adversarial defense techniques. Through adversarial training and provable robustness calculation techniques, it achieves the goal of increasing the cost of generating adversarial samples as much as possible and reducing the success rate of attacking neural networks with adversarial samples while maintaining the performance of neural networks to meet requirements, thereby promoting the application of neural networks in security fields.

1.2. Research content

This research focuses on neural network adversarial defense algorithm research. By deeply studying the effectiveness of adversarial training in improving neural network robustness, designing new types of adversarial training algorithms, and using provable robustness calculation methods to replace existing heuristic-based robustness calculation methods, it achieves robustness improvement for neural networks. The research content mainly includes the following aspects: The research content mainly includes the following aspects:

Neural network defense algorithm based on adversarial training

Existing adversarial training methods usually use various adversarial attack algorithms to generate adversarial samples and train neural networks with adversarial samples instead of real sample training. This method can achieve good robustness improvement effects in small data sets, but the defense effect obtained by this method has no defense against black box attacks. The network performance has a certain degree of decline in real samples, and it requires much more computing power than normal training a neural network. Therefore, special loss functions and new network structures need to be designed so that methods based on adversarial training can be extended to more extensive applications.

Defense algorithm based on denoising of adversary features

Existing neural network defense algorithms based on adversary training all need to adopt specific attack algorithms to attack neural networks, obtain adversary samples, re-enter adversary samples into the network and retrain them with correct labels. This method can usually achieve defense against trained attack methods, but its defense effect against newly emerging attack methods is not good. Therefore, we need to design a defense method that can improve the robustness of neural networks without relying on specific attack algorithms for adversary training. Research has found that activation caused by adversary perturbation is highly related to network parameters which usually manifests as sparse activation which differs greatly from real data activation Using traditional filtering denoising techniques can suppress this sparse activation thereby achieving improved robustness for neural networks.

1.3. Key issues to be solved

How to design a training process that reduces computational power requirements for calculating adversary samples How to design a network structure so that defensive effects can be generalized

in large-scale networks and complex data. How feature denoising algorithms are designed so that feature transformations remove adversary features from adversary samples while retaining data features consistent with real samples achieving improved robustness for neural networks.

1.4. Basis and significance of topic selection

Neural networks are an important technology in the field of machine learning. It establishes a simple computational model by abstracting neurons simply and connects them in different ways to form different networks. Thanks to the rapid increase in hardware computing power, neural networks have been widely used in many fields in recent years through powerful learning ability and automatic feature extraction ability, and have achieved superior performance that traditional methods in these fields cannot achieve in many tasks. For example, image classification, face recognition, behavior recognition speech recognition etc. However, while neural networks have achieved great success researchers have found that neural networks are very vulnerable to adversary sample attacks. Adversary samples are false data obtained by adding very slight perturbations on the basis of real data. Although adversary samples are very similar to real data, neural networks cannot correctly identify this kind of false data thus giving wrong output. Therefore, the discovery of adversary samples brings great security risks to many application systems based on neural networks. For example, if adversary samples are applied to face recognition payment systems, if face recognition systems cannot correctly identify user identity it will cause serious economic losses. Researchers have proven that adversary samples are widely present and diverse. Goodfellow proposed a method for quickly calculating adversary samples FGSM in 2014. Moosavi-Dezfooli aimed at the shortcomings of FGSM proposed a method for calculating minimal perturbation adversary samples using this method can make adversary samples closer to real data. Sparse attacks and single-pixel attacks were also subsequently proposed [4] these methods can deceive neural networks by changing only a few pixels. Adversary samples were first discovered in image classification tasks but in subsequent research researchers found that adversary samples can be used to attack various systems based on neural networks. [5] found that adversary samples can deceive image segmentation networks. [6] studied a way of calculating physical adversary samples which can deceive neural networks with physical objects made by 3D printing.

The current neural network-based AI models have gradually moved out of the laboratory environment and entered into practical commercial or civilian fields. Most smartphones have facial recognition function, and users can even use facial recognition for quick payments; many international airports use facial recognition devices to replace manual means of verifying passengers' identities. These fields have extremely high requirements for the robustness and security of artificial intelligence. Therefore, the emergence of adversarial samples limits the further development of artificial intelligence models in these fields, and may even threaten the currently existing artificial intelligence systems, resulting in serious losses. Therefore, research on neural network security has become a current research hotspot. On the one hand, research on neural network security is aimed at improving the security of current neural network models, avoiding their being attacked by adversarial samples and causing economic losses, promoting the further development of neural network models in fields with extremely high security requirements. On the other hand, it is to enable researchers to further understand and comprehend the black-box model of neural networks, to consider the key conditions that

determine their robustness, and to promote the further development of the artificial intelligence field, pushing artificial intelligence to better serve the real world.

2. Literature Review:

2.1. Research status of adversarial sample attack methods

The main purpose of adversarial sample attack is to generate data samples that can make neural networks judge wrongly with high probability by calculation, while requiring data samples to be as similar as possible to real data. Here we briefly introduce some representative attack methods.

L-BFGS Attack

Szegedy [7] first proposed the concept of adversarial samples in 2014, and proposed to use the L-BFGS algorithm to solve the following optimization problem to obtain adversarial samples

$$\min_x c\|\eta\| + J_\theta(x', l'), s. t. x' \in [0, 1]$$

At the same time, they pointed out that adversarial samples have transferability, that is, the adversarial samples calculated by using a known model can effectively attack the sample, and can also attack other models similar to the model, even if the parameters of other models are not used in the process of calculating the adversarial samples.

Single-step attack

L-FBGS attack uses a complex optimization algorithm, especially the linear search method to select the appropriate parameter c , which makes the L-FBGS method unable to efficiently use in large networks, and cannot use the existing neural network framework for acceleration. Therefore, Goodfellow [1] proposed a method that can quickly calculate adversarial samples. By calculating the gradient of each pixel to the classification loss function, the pixel is updated once in the direction of increasing the loss function to obtain the adversarial sample. The adversarial perturbation can be expressed by mathematical formula as

$$\eta = \epsilon \cdot \text{sign}(\nabla_x J_\theta(x, l))$$

Where ϵ represents the perturbation size of the adversarial sample, that is, the similarity degree of the adversarial sample to the real sample, the smaller the ϵ , the closer the adversarial sample is to the real sample. The adversarial sample can be expressed as $x' = x + \eta$. The calculation of this adversarial attack can use the existing neural network framework, such as TensorFlow, Pytorch, etc. for GPU acceleration calculation, which can perform very efficient calculation.

In subsequent research, the FGSM attack method derived a number of attack methods, including [8] directly using the gradient to calculate the adversarial sample $\eta = \epsilon \cdot \nabla_x J_\theta(x, l)$; [9] modified the loss function The target category and direction make the adversarial sample induce the neural network to produce a specific category of output, $\eta = -\epsilon \cdot \nabla_x J_\theta(x, l')$, l' is the specified wrong category.

Iterative attack

At the same time, the single-step FGSM method also derived a method based on iterative FGSM. Kurakin [9] iteratively generated adversarial samples by using the FGSM attack method multiple times. Considering that the format of the input data is strictly restricted in some applications, they used the following iterative formula

$$\begin{aligned}x_0 &= x \\x_{n+1} &= \text{clip}_x \left\{ x_n + \epsilon \cdot \text{sign} \left(\nabla_x J_\theta(x_n, l) \right) \right\} \\ \text{clip}_{x,\xi} \{x'\} &= \min \{1, x + \xi, \max \{0, x - \xi, x'\}\}\end{aligned}$$

By controlling the parameters of the *clip* function, the adversarial samples generated by each step of iteration are strictly restricted within the pixel range of normal image data. Similarly, by modifying the loss function, specific target attacks can also be implemented iteratively.

Sparse attack

The attack methods introduced above usually only consider limiting the infinity norm or the second norm of the adversarial perturbation, while some attack methods implement adversarial attacks that limit the zero norm, that is, the adversarial samples and the real data only have a few pixels. There is a difference. Jacobian-based Saliency Map Attack [10] proposes to calculate the Jacobian matrix of the input sample, and then select a small number of pixels to modify the original data according to the Jacobian matrix to achieve the attack. One Pixel Attack [3] uses an optimization method based on evolutionary algorithms to modify only one pixel of the original image and successfully attack the neural network. Similar attack methods also include sparse attacks [4] and so on.

DeepFool

Moosavi-Dezfooli and colleagues were inspired by the local linearity of neural networks and designed an iterative attack method called DeepFool to generate adversarial examples [2]. This method assumes that real data points are located at linear segments of a neural network's decision function and uses the linear property to calculate the decision function boundary closest to the real data point. Because the actual neural network is not a linear model, an iterative approach is needed to compensate for the gap between the linear assumption and the non-linear model. However, the authors found that usually this type of attack only needs to be iterated fewer than 5 times to complete the attack, which indirectly proves the effectiveness of the linear assumption [13]. They also introduced the concept of Universal Adversarial Perturbations [11], which means that an adversarial perturbation can be added to all images to deceive image recognition networks. They proposed a method to calculate this universal adversarial perturbation by repeatedly using the DeepFool attack on different images.

Universal perturbation attack

Most of the attack methods are to calculate adversarial samples for individual input data, that is, for each image, a calculation needs to be run to generate the corresponding adversarial sample. Moosavi achieved a universal perturbation attack [11], that is, a specific adversarial perturbation can be calculated, which can deceive the neural network with a high probability after adding it to all input data. This kind of adversarial perturbation is generated by using an iterative method. In each iteration, a sample is randomly drawn from the data set, and the adversarial perturbation generated by the previous iteration is used as the initial value to calculate the current sample's

adversarial perturbation using DeepFool attack. This iteration continues until the generated adversarial perturbation can deceive most of the inputs of the neural network. Experiments show that the adversarial samples generated by this method can effectively migrate to many representative deep learning network frameworks.

Non-norm-constrained attacks

In order to address the issue of adversarial examples in more diverse scenarios, a new type of non-norm-constrained adversarial examples has emerged, and researchers have proposed many generation techniques specifically for this type of adversarial examples. Reference [12] proposed an adversarial example generation technique that is not limited to digital inputs, demonstrating that adversarial examples can be generated using printed methods in real-world physical environments. The authors argued that adversarial examples that affect real-world physical environments should take into account factors such as the angle, distance, and resolution at which digital imaging systems capture images of physical adversarial examples. They applied their algorithm to the generation of adversarial examples for road sign recognition systems and tested it on actual roads, achieving a high attack success rate. In this work, they implemented two types of attacks: one was a large-scale perturbation attack, in which the attacker generated a complete fake road sign and placed it on the road, causing the sign recognition system to make errors without pedestrians noticing anything unusual about the sign; the other was a sticker-type attack, in which the attacker generated small adversarial example stickers and pasted them around normal road signs, causing the sign recognition system to make errors while retaining the integrity of the main sign identification. Their adversarial example generation technique only requires the use of an ordinary color printer and does not require other special printing equipment or hardware.

Reference [13] also demonstrated an adversarial example attack method that targets street sign classifiers. They proposed adding maliciously constructed images to training data during the neural network training process, which can make the final street sign classifier exhibit advanced recognition performance on normal data samples, but once the attacker provides a maliciously constructed input, the network's performance will decline significantly. They implemented this special attack method in a real-world scenario by adding special marks to parking signs, causing the street sign classifier to recognize parking signs as speed limit signs.

Rozsa et al. also believed that the search space for adversarial examples should not be limited to the norm ball of real data [14]. They proposed that small image transformations and rotations of real images can also generate adversarial examples. This method does not directly use norm distances to evaluate the distance between adversarial examples and real data, but defines an index called the Psychometric Perceptual Adversarial Similarity Score (PASS), which is a variant of the structural similarity (SSIM) [15], and its calculation involves first aligning the adversarial example and the original sample, then calculating the correlation coefficient between the two samples. Although this method abandons widely used norm constraints, the generated adversarial examples are still imperceptible to the human eye. The authors used experiments to demonstrate that this method generates more diverse adversarial examples than the FGSM method.

Non-norm-constrained adversarial examples have expanded the connotation of adversarial examples and have achieved success in many practical application fields. However, because the diversity of non-norm-constrained adversarial examples is closely related to practical application scenarios, this type of adversarial example generation technique is more complex than gradient descent-based adversarial example generation techniques, typically resulting in higher time complexity for generating adversarial examples.

Other adversarial attacks:

In reference [16], a very extreme adversarial attack method was proposed to deceive image recognition classifiers. The algorithm claimed that only changing one pixel on the image could successfully deceive three different network models on 70% of test images. The algorithm used a differential evolution-based optimization algorithm to generate adversarial examples. For a normal image I_c , 400 candidate vectors were first created, each containing the x, y coordinates and RGB color values of the pixels to be modified. Then, these vectors were randomly modified to create offspring individuals, which competed with their parent individuals in the next iteration. The predicted probability values of the neural network for the modified images represented by these vectors were used as the fitness criterion for individuals. Through this method, the individual that could make the classification prediction value the smallest was found in the iterative evolution process, and the image represented by that individual was used as the final adversarial example. The reference proved that even with a simple evolutionary strategy, adversarial examples could be effectively calculated. It should be noted that differential evolution is a gradient-independent optimization algorithm, so the attack algorithm proposed in the reference is very suitable for deceiving black-box neural networks that cannot obtain gradients directly.

In reference [17], adversarial example generation technology was applied to malicious software production. They found that current popular anti-malware products all use machine learning and deep learning technology to detect malicious software based on file features. Although this method can detect unknown malicious software more effectively than traditional signature-based software authentication methods, it also creates security risks due to the existence of adversarial example generation technology. They used an end-to-end training method to modify the binary files of malicious software, evading detection while keeping the malicious functions unchanged, causing anti-malware detection algorithms to classify the binary files of malicious software as benign.

In addition to traditional heuristic iterative algorithms, white-box attacks also include using network-generated adversarial attacks. Xiao et al. [18] introduced the generative adversarial network [19] into the scene of generating adversarial examples and proposed the AdvGAN attack network. Unlike directly calculating the gradient of the loss function with respect to the input image, the network model was pre-trained to directly transform clean samples into adversarial perturbations, which was more efficient than other computationally generated adversarial example generation algorithms. The pre-trained network can efficiently transform random noise into adversarial perturbations, and can obtain adversarial examples that are more consistent with human perception.

2.2. The current state of research on adversarial sample defense methods

While studying attack methods, researchers are also actively studying methods to defend against adversarial sample attacks. Generally speaking, defense methods can be divided into two categories. One is passive defense methods, which means that after the generation of adversarial samples, methods such as adversarial sample detection or input sample reconstruction are used to make the adversarial samples generated by the attacker lose their attack ability. The other is active defense methods, which generally means that special network structures, training methods or special loss functions are used to train the neural network during training, so that the neural network itself has natural robustness, making it impossible for the attacker to effectively generate adversarial samples, thus achieving the effect of defense.

Defensive distillation

Papernot [20] proposed using network distillation to defend against adversarial sample attacks. Network distillation was originally a technique proposed for network model compression. The output of a large network is used as the label of another small network to train the small network, thus transferring the knowledge contained in the large network to the small network, and then replacing the large network with the small network to achieve the purpose of network compression. In classification neural networks, usually the last layer of the network uses Softmax function to normalize the output and output the classification probability of each category. The output of Softmax can be expressed as

$$q_i = \frac{\exp\left(\frac{z_i}{T}\right)}{\sum_j \exp\left(\frac{z_j}{T}\right)}$$

Where T is the distillation temperature. In general, neural networks, T is usually set to 1. When T is very large, the output of Softmax will tend to converge, that is, the classification probability of each category will tend to $1/m$, where m is the number of categories. And when T is very small, the output corresponding to the element with the largest input value will tend to 1, while the classification probability of other categories will tend to 0. It has been found that most attacks use network sensitivity information to update inputs, thus perturbing inputs in the direction of increasing loss function. By using a higher distillation temperature to distillate network, it can reduce network sensitivity and thus reduce attack success rate of adversarial samples on neural network.

Adversarial retraining

Using adversarial samples instead of real samples to train neural networks is a common method of defending against adversarial attacks. Goodfellow and Huang [21] studied injecting adversarial samples into training set in each iteration of training process and found that this kind of adversarial training can effectively enhance network robustness. It was also found that using adversarial samples to train neural networks can improve network generalization performance and accuracy.

At the same time, research [9] found that on large-scale image classification task Imagenet, using real image data and adversarial samples to train classification network can improve network robustness against single-step adversarial attacks, but not against iterative attacks. Some research [22] also believes that adversarial training can only defend against white-box attacks and does not have enough defense effect against black-box attacks.

Adversarial sample detection

There are also many studies focusing on detecting malicious adversarial samples in input data. Using detection techniques can reject processing of adversarial samples, or reconstruct adversarial samples to restore real samples. Thus, achieving the purpose of defense. [23] and [24] studied and designed classifiers that can classify adversarial samples and real samples. Metzen designed a small detector as an auxiliary structure of normal neural network, which outputs the probability of input sample being an adversarial sample, and the detector itself is a small neural network. SafetyNet extracts the output of neural network ReLU layer as features, and uses RBF-SVM classifier to classify features, judging whether input is an adversarial sample or not. [25] trained a PixelCNN neural network and found that the distribution of adversarial samples is very different from normal samples. They based on p-value of PixelCNN's activation degree for input data, and used p-value as threshold to distinguish between adversarial samples and normal samples. Their experiments show that this method can detect FGSM, BIM, DeepFool and CW attacks.

Input reconstruction

Adversarial samples can be transformed into normal samples using reconstruction methods. Gu and Rigazio [26] proposed a series of auto-encoding networks to reconstruct samples. There are also methods that use denoising auto-encoding networks to remove the adversarial perturbations from the input data and reconstruct the real samples. PixelDefend [25] uses PixelCNN to defend against attacks by remapping the input sample distribution to the training set distribution. PixelDefend also uses adversarial sample detection methods, that is, only remapping data that is detected as adversarial samples.

Robustness provable defense methods

Most of the above methods are based on heuristic defense, while a class of research focuses on provable network robustness computation. Katz [27] proposed a robustness verification method for neural networks using ReLU activation functions, Reluplex. They use satisfiability modulo theory solvers to verify neural networks. Experiments show that this method can prove the non-existence of adversarial samples within a small perturbation range. At the same time, they proved that this method is NP-complete. Therefore, this method can only be applied to small neural networks with only a few hundred nodes.

3. Methods:

3.1. Main theories

Adversarial training: Using adversarial sample generation techniques and training the network with adversarial samples is one of the defense measures to protect neural networks from adversarial sample attacks. Its core principle is to use adversarial sample generation techniques to solve an upper bound of adversarial robustness for neural networks, and use backpropagation to update network parameters to improve the upper bound of network robustness.

Feature denoising: The activation of neural networks caused by adversarial perturbations is highly correlated with network parameters, usually showing sparse activation, which is quite different from the activation of real data. Using traditional filtering denoising techniques can suppress this sparse activation, thereby achieving the purpose of improving neural network

robustness. At the same time, this defense method does not target specific adversarial attack methods, thereby achieving attack-independent defense effects.

3.2. Research methods

- Literature research method: Systematically collect and read relevant literature, study adversarial training and computable robustness techniques and their applications in neural robustness improvement;
- Experimental observation method: Verify and observe the feasibility of related algorithms and improvements through experiments;
- Empirical summarization method: Inductively analyze the specific situations in the experiments to make them systematic and theoretical.

3.3. Technical route and implementation plan

To address the issue of insufficient stealthiness of existing adversarial samples, this study proposes a semantic similarity-based adversarial attack algorithm. The specific implementation plan is as follows:

1. In order to attack an image at the feature level, an encoder is needed to represent the image in the feature space. This process can be achieved by using a pre-trained deep neural network (such as VGG, ResNet, etc.). Firstly, the clean image is input into the encoder, and after a series of operations including convolution and pooling layers, a high-dimensional feature vector is obtained. This feature vector can be regarded as the representation of the image in the feature space;
2. To generate adversarial examples, an image in the same batch that is most dissimilar to the clean image in the feature space is selected. A technique similar to contrastive learning is used to minimize the distance between the adversarial image and the clean image, while increasing the distance between the adversarial image and the normal image;
3. Research has shown that the human visual system is not sensitive to fine details. In the frequency domain, these details are represented as high-frequency components. To further increase the loss and ensure semantic similarity, we maintain consistency between the low-frequency information of the adversarial image and the low-frequency information of the clean image. This is achieved by minimizing the mean squared error between the low-frequency components of the adversarial image and those of the clean image in the Fourier domain. This approach preserves the overall structure and content of the image while perturbing the high-frequency components to generate the adversarial image.
4. And using a similar method as C&W, an optimization approach is adopted to generate the final adversarial samples.

Aiming at problem that defense methods based on adversary training have insufficient transferability, this research intends to propose a defense algorithm based on adversary feature denoising. The specific implementation plan is as follows:

1. Study similarities between activation degree between adversary samples with real data on neural networks;
2. Design of intermediate mapping layers:

For a neural network $F(\cdot)$, where $F_i(\cdot)$ is defined as the output of the i -th layer in the neural network. We need to learn a transformation function $T(\cdot, \cdot)$ that transforms the intermediate layer features of adversarial samples to be as close as possible to those of normal samples, i.e., minimizing the difference between them:

$$\sum_i D\left(F_i(x), T\left(F_i(x')\right)\right)$$

Where x is the normal sample, x' is the corresponding adversarial sample, and $D(\cdot)$ is the metric function that measures the difference between two intermediate layer features.

3. Selection of distance metric function:

In the field of adversarial attack and defense, L_∞ and L_2 are commonly used as distance metric functions to measure the difference between adversarial samples and original samples. However, we chose to use L_1 to measure the difference between features. L_∞ places too little constraint on the intermediate layer features, allowing the adversarial samples to have more freedom to deceive the neural network. Moreover, in our study, we found that the difference between the shallow features of adversarial samples and normal samples is very small, and this difference increases as the layer depth increases. Therefore, if we use L_2 distance metric function to learn multiple feature mapping layers, the deep mapping layer will learn faster than the shallow mapping layer because the gradient of the deep mapping layer is larger. However, we want the shallow mapping layer to learn better because the interference of adversarial samples on the neural network gradually increases from shallow to deep. Therefore, we use L_1 to measure the difference between features.

4. Mapping layer structure:

Because the mapping layer is an additional network to the original neural network, it is necessary to design a simple enough mapping layer to ensure the computational efficiency of the original neural network. We used two convolutional layers with a kernel size of 3 and one convolutional layer with a kernel size of 1, and added ReLU layers between each layer. To improve learning efficiency, we adopted the idea of residual learning and added a shortcut connection between the output and input of the mapping layer, so that the mapping layer learns the residual between the adversarial features and the normal features.

5. Adversarial training methods:

For a trained but unprotected neural network $F(\cdot)$, a robust neural network can be constructed by adding an additional feature transformation layer. A protected neural network model needs to learn the transformation between real samples and adversarial samples through adversarial training. First, real samples are fed into the neural network to calculate the intermediate layer features. Then, a batch of adversarial samples are generated using adversarial sample attack methods and input into the neural network again to extract the intermediate layer features of the adversarial samples. The obtained normal features and adversarial features are used as training data for the feature mapping layer to learn the transformation function between the two.

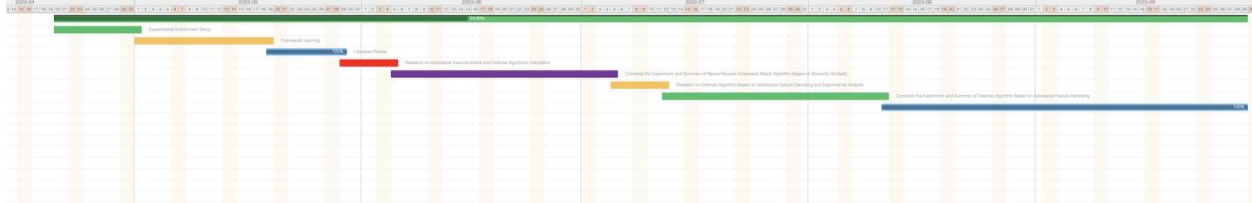
The training method mentioned above ensures that the neural network can correctly classify adversarial samples. However, some studies have shown that this training method can reduce the neural network's accuracy in classifying normal samples.

Therefore, we adopted a mixed batch processing approach for training. Specifically, the loss function of the neural network during training includes both the classification loss for normal data and the classification loss for adversarial samples. Therefore, the total loss function for adversarial training is:

$$L(F_p(x_{adv}), y) + L(F_p(x), y) + \sum_i D(F_i(x), T(F_{pi}(x_{adv})))$$

6. Use adversary sample generation algorithm for attacking calculation on neural networks, verify classification accuracy when facing attacks.

Here is a Gannt chart which illustrates the implement plan and schedule of this project:



4. Results:

In order to demonstrate the effectiveness of the proposed algorithm, iterative network model pruning experiments are expected to be conducted on three classic image datasets: CIFAR-10, MNIST and Fashion-MNIST. This section will focus on introducing these two commonly used datasets.

The CIFAR-10 dataset is a commonly used collection of color image data for validating computer vision algorithms. The dataset contains a total of 50,000 training images and 10,000 testing images, with 10 image categories: frogs, airplanes, horses, cats, trucks, birds, cars, dogs, deer, and ships. Each category contains 5,000 training images and 1,000 testing images, with each image being a 32×32 three-channel color image. Each image corresponds to only one true label, and there are no cases where multiple categories exist in a single image.

The MNIST dataset is a collection of handwritten digit images, consisting of 70,000 images in total. These images were contributed by approximately 250 different experiment subjects and were divided into a training set of 60,000 images and a test set of 10,000 images. Each image is a single-channel grayscale image with a size of 28 x 28 pixels.

The Fashion-MNIST dataset is a variation of the MNIST dataset with the same image format but different content. It contains images of 10 different categories of fashion items, including shirts, pants, handbags, etc. All images in the dataset were provided by Zalando company.

In the specific experiments, adversarial samples will be generated under multiple maximum allowable perturbation settings for optimizing the Euclidean distance and optimizing the Chebyshev distance, respectively. Then, the generated adversarial samples are expected to be classified using an image classification model. The classification accuracy of the image classification model on the adversarial samples will be recorded under different maximum allowable perturbation settings.

5. Evaluation:

Maximum perturbation, attack success rate, and structural similarity are chosen as the final evaluation metrics against attacks and defenses.

Maximum perturbation: In order to make the adversarial samples maintain the human eye visual similarity during the attack, we need the spatial distance between the adversarial samples and the real samples to be less than a certain value, so that the perturbation is kept within a certain range. In this paper, we use the Chebyshev distance and Euclidean distance to measure the distance between each sample and denote the value by ϵ .

Attack Success Rate (ASR): The correct attack rate measures the strength of the adversarial generation algorithm, and the higher the attack success rate, the lower the recognition rate of the target neural network classification model. Assuming that the number of classes of the original data samples is N , X_i^a and X_i^b denote the number of samples of class i that can be correctly identified before and after the attack, respectively, the formula is as follows:

$$ASR(X^a, X^b) = \frac{\sum_{i=1}^n X_i^b}{\sum_{i=1}^n X_i^a} \times 100\%$$

Structural similarity: Structural similarity expresses the similarity of two images. To measure the stealthiness of the adversarial perturbation, we can compare the same image classification models under the same adversarial perturbation strength to damage the image structure. If the attack success rates of the two adversarial generation algorithms are comparable, the adversarial algorithm with greater structural similarity is more similar to the original image and the stealthier the algorithm is.

Reference

- [1] I. J. Goodfellow, J. Shlens and C. Szegedy, "Explaining and Harnessing Adversarial Examples," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.
- [2] S.-M. Moosavi-Dezfooli, A. Fawzi and P. Frossard, "DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, 2016.
- [3] J. Su, D. V. Vargas and K. Sakurai, "One Pixel Attack for Fooling Deep Neural Networks," *IEEE Transactions on Evolutionary Computation*, vol. 23, p. 828–841, 10 2019.
- [4] A. Modas, S.-M. Moosavi-Dezfooli and P. Frossard, "SparseFool: A Few Pixels Make a Big Difference," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, 2019.
- [5] C. Xie, J. Wang, Z. Zhang, Y. Zhou, L. Xie and A. L. Yuille, "Adversarial Examples for Semantic Segmentation and Object Detection," in *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, 2017.
- [6] A. Athalye, L. Engstrom, A. Ilyas and K. Kwok, "Synthesizing Robust Adversarial Examples," in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, 2018.
- [7] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.
- [8] A. Rozsa, E. M. Rudd and T. E. Boult, "Adversarial Diversity and Hard Positive Generation," in *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2016.
- [9] A. Kurakin, I. J. Goodfellow and S. Bengio, "Adversarial Machine Learning at Scale," in *5th International Conference on Learning Representations*, 2017.
- [10] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik and A. Swami, "The Limitations of Deep Learning in Adversarial Settings," in *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, 2016.
- [11] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi and P. Frossard, "Universal Adversarial Perturbations," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [12] Gu T, Dolan-Gavitt B, Garg S. Badnets, "Identifying vulnerabilities in the machine learning modelsupply chain," CoRR, 2017.
- [13] Rozsa A, Rudd E M, Boult T E, "Adversarial diversity and hard positive generation," in *Conference on Computer Vision and Pattern Recognition*, 2016.
- [14] Flynn J R, Ward S, IV J A, et al., "Image quality assessment using the SSIM and the just noticeable difference paradigm," *Engineering Psychology and Cognitive Ergonomics*, 2013.
- [15] thalye A, Engstrom L, Ilyas A, et al. Synthesizing robust adversarial examples," *Proceedings of the 35th International Conference on Machine Learning*, 2018.
- [16] Su J, Vargas D V, Sakurai K. One pixel attack for fooling deep neural networks," *IEEE Transactions on Evolutionary Computation*, 2019.
- [17] Rosenberg I, Shabtai A, Rokach L, et al. Generic black-box end-to-end attack against state of the art API call based malware classifiers," *Research in Attacks, Intrusions, and Defenses*, 2018.
- [18] Papernot N, McDaniel P, Jha S, et al. The limitations of deep learning in adversarial settings," *IEEE European Symposium on Security and Privacy*, 2016.
- [19] Goodfellow I, Pouget-Abadie J, Mirza M, et al, "Generative adversarial networks," *Communications of the ACM*, 2020.
- [20] N. Papernot, P. D. McDaniel, X. Wu, S. Jha and A. Swami, "Distillation as a Defense to Adversarial

- Perturbations Against Deep Neural Networks," in *IEEE Symposium on Security and Privacy*, 2016.
- [21] R. Huang, B. Xu, D. Schuurmans and C. Szepesvári, "Learning with a Strong Adversary," *CoRR*, vol. abs/1511.03034, 2015.
 - [22] F. Tramèr, A. Kurakin, N. Papernot, I. J. Goodfellow, D. Boneh and P. D. McDaniel, "Ensemble Adversarial Training: Attacks and Defenses," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.
 - [23] J. Lu, T. Issaranon and D. Forsyth, "SafetyNet: Detecting and Rejecting Adversarial Examples Robustly," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017.
 - [24] J. H. Metzen, T. Genewein, V. Fischer and B. Bischoff, "On Detecting Adversarial Perturbations," in *5th International Conference on Learning Representations*, 2017.
 - [25] Y. Song, T. Kim, S. Nowozin, S. Ermon and N. Kushman, "PixelDefend: Leveraging Generative Models to Understand and Defend against Adversarial Examples," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.
 - [26] S. Gu and L. Rigazio, "Towards Deep Neural Network Architectures Robust to Adversarial Examples," 11 12 2014.
 - [27] G. Katz, C. Barrett, D. Dill, K. Julian and M. Kochenderfer, "Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks," 32 2017.