AdvAttackVis: An Adversarial Attack Visualization System for Deep Neural Networks

DING Wei-jie[1]*, Shen Xuchen[2], Yuan Ying[3], MAO Ting-yun[4], SUN Guo-dao[5], CHEN Li-li[6], CHEN bing-ting[7]
Department of Computer and Information Security, Zhejiang Police College, Hangzhou 310053 China[1, 3]
Key Laboratory of Public Security Information Application Based on Big-data Architecture, Ministry of Public Security, Hangzhou 310053 China[1]
Xiaoshan District branch of Hangzhou Public Security Bureau, Hangzhou 310053 China[2]
Zhejiang Dahua Technology Co., Ltd. Hangzhou 310053, China[4, 6]
College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou 310023 China[5]
College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016 China[7]

Abstract—**Deep learning has been widely used in various scenarios such as image classification, natural language processing, and speech recognition. However, deep neural networks are vulnerable to adversarial attacks, resulting in incorrect predictions. Adversarial attacks involve generating adversarial examples and attacking a target model. The generation mechanism of adversarial examples and the prediction principle of the target model for adversarial examples are complicated, which makes it difficult for deep learning users to understand adversarial attacks. In this paper, we present an adversarial attack visualization system called AdvAttackVis to assist users in learning, understanding, and exploring adversarial attacks. Based on the designed interactive visualization interface, the system enables users to train and analyze adversarial attack models, understand the principles of adversarial attacks, analyze the results of attacks on the target model, and explore the prediction mechanism of the target model for adversarial examples. Through real case studies on adversarial attacks, we demonstrate the usability and effectiveness of the proposed visualization system.**

Keywords—Deep learning; deep neural networks; adversarial attacks; adversarial examples; interactive visualization

I. INTRODUCTION

With the development of artificial intelligence, deep neural networks are widely applied to various scenarios such as image classification, natural language processing and speech recognition. However, More and more studies reveal that deep neural networks are vulnerable to adversarial attacks [1]. Adversarial attacks are to generate adversarial examples by adding weak adversarial perturbations to the input examples. The generated adversarial examples can interfere with the decision-making of deep neural networks, resulting in incorrect predictions. Thus, adversarial attacks pose a serious threat to the application of deep learning. Delving into adversarial attacks helps to reveal the weaknesses of deep learning models, which in turn motivates researchers to design effective defense strategies to improve the robustness of neural networks.

The goal of adversarial attacks is to generate adversarial examples which can induce deep learning models to make incorrect predictions. Szegedy et al. [1] discovered early that deep neural networks are vulnerable to adversarial examples in the field of image classification. Specifically, the input images with adversarial noise (i.e., adversarial examples) can induce image classification models to produce incorrect classification results. Since then, a number of adversarial attack methods have been proposed. Goodfellow et al. [2] proposed a classical gradient-based attack method namely FGSM. The method adds the gradients that increases the loss function to the original image to generate an adversarial example. The idea based on gradient derives many gradient-based attack methods such as PGD [3], BIM [4] and ILCM [4]. To obtain adversarial examples with high perceptual quality, Carlin and Wagner [12] proposed an optimization-based attack method which produces adversarial examples with small perturbations and high attack success rates by minimizing the objective function. The above attack methods require access to the target model (i.e., the attacked model) when generating adversarial examples, resulting in low efficiency in generating adversarial examples. To obtain adversarial examples quickly, Xiao et al. [5] presented an attack method based on generative adversarial networks (GAN). This method can directly generate adversarial examples with a high attack success rate without accessing the target model again after the generator is trained.

Adversarial attacks involve generating adversarial examples and attacking target models. However, the generation mechanism of adversarial examples and the prediction principle of the target model for adversarial examples are complicated, which makes it difficult for deep learning users especially beginners to understand adversarial attacks. To address this problem, we present an adversarial attack visualization system called AdvAttackVis, which can assist users to learn, understand and explore adversarial attacks. The visualization system is implemented based on B/S architecture, which is easy for deep learning users to use. Based on the underlying models and the designed interactive visualizations, the system enables users to train and explore adversarial attack models, understand the principles of adversarial attacks, analyze the results of adversarial attacks, and explore the prediction mechanism of the target model for adversarial examples. Through a case study where AdvGAN (attack model) attacks an MNIST classifier (target model), we demonstrate the usability and effectiveness of our system. AdvAttackVis provides end-to-end support for adversarial attack analysis. We have also specifically designed visualization views from different analysis perspectives, providing noval insights for analyzing adversarial attacks. This enables users to explore and understand the attack process comprehensively, whereas existing tools often only support post-analysis of existing attacks.

Our contributions can be summarized as follows:

- A visual analysis scheme for end-to-end interpretability of adversarial attacks. The scheme combines interactive visualizations with underlying algorithms to help users delve into adversarial attacks.
- An adversarial attack visualization system for deep neural networks. The system offers powerful interactive visualization views to assist users in exploring adversarial attacks.
- We demonstrate the usability and effectiveness of the visualization system via a real case study base on the MNIST handwritten digital image dataset.

II. RELATED WORK

A. Adversarial Attacks

Adversarial attacks involve generating adversarial examples and attacking the target model. Szegedy et al. [1] first proposed the concept of adversarial examples. Specifically, an adversarial example is generated by adding weak adversarial interference to the original image. It could induce the target model to output wrong predictions with high confidence, which reveals the vulnerability of deep neural networks and raises concerns about the security of deep learning models. Goodfellow et al. [2] argued that this vulnerability is caused by the local linearity of neural networks, especially when a linear activation function (e.g., ReLU [1]) is used in the model. Arpit et al. [7] analyzed the memory ability of neural networks for training data and found that models with high memory levels are more susceptible to adversarial examples. Gilmer et al. [10] believed that adversarial examples are attributed to the high-dimensional geometric structure of data manifold and then analyzed argued the relationship between adversarial examples and the high-dimensional geometry of data manifold.

From the perspective of the attack environment, adversarial attacks can be divided into white-box attacks and black-box attacks [7]. In white-box attacks, the attacker can obtain information about the target model such as training data, model structure, model parameters and model output. Therefore, the attacker can directly utilize the gradient of the loss function and classification hyperplane of the neural network to calculate the required perturbations, thereby generating adversarial examples. FGSM [2] is a classical white-box attack method which obtains adversarial examples by adding perturbations to the input images. Madry et al [3] improved FGSM and proposed PGD to further improve the attack success rate. To generate the adversarial examples with high perceptual quality, Carlini and Wagner [12] proposed an optimization-based attack method which obtains the smallest adversarial perturbation by iteratively optimizing the objective function. However, in practical applications, attackers usually have no access to the detailed information of the target model, which makes it impossible to implement white-box attacks. To overcome this limitation, researchers have proposed black-box attacks where the attacker only needs the output labels or probability vectors of the target model. Single-pixel attack [14] is a classical black-box attack method which obtains an adversarial example by changing the value of the selected pixel in the original image. Sarkar [15] proposed two black-box attack methods namely UPSET and ANGRI. UPSET generates general perturbations for all output categories while ANGRI produces specific perturbations for different images. Their generated adversarial examples can induce the image classification model to output specific categories. Xiao et al. [5] proposed an attack method based on generating adversarial networks (GAN). The method first applies knowledge distillation to obtain an agent model with similar performance to the target model and then generates adversarial examples that are effective against the target model.

In the white-box or black-box attack scenario, adversarial attacks can be further divided into targeted attacks and untargeted attacks [7-9]. Targeted attacks induce the deep learning model to recognize adversarial examples as a specific class while untargeted attacks simply require the model to output the incorrect class. The white-box attack method FGSM [2] can support both targeted and untargeted attacks. For black-box attacks, AdvGAN [5] can achieve targeted and untargeted attacks by designing the loss function of the target model.

However, existing work on interpretability analysis of adversarial attacks is still relatively lacking. They mostly focus on the surface effects of attacks, such as success rates, perturbation sizes, etc., while rarely delving into the causes of attacks, characteristics of attack samples, and the impact of attacks on model behavior.

B. Visualization of Deep Learning Models

Deep learning models are regarded as black-box models because their decision-making mechanisms are not clear to human cognition. In recent years, researchers have proposed different visualization techniques [16] [29] [30] to reveal the inner workings of deep learning models and assist users to understand, analyze and learn deep learning. TensorBord is a visualization tool launched by the TensorFlow deep learning framework, which can visualize computational graph and training logs. However, TensorBord lacks an interactive visual design that goes deep into the model, so users are unable to gain a deep understanding of the internal mechanism of the model. Liu et al. [19] proposed an interactive visualization technology namely CNNVis to help users understand, diagnose and improve convolutional neural networks (CNN). This technology visualizes the convolutional neural network as a directed acyclic graph where neurons in the convolutional layers are clustered and connections between neurons are bundled to reduce visual clutter. It allows users to interactively inspect the details of the model (e.g., activation values and weights). Recently, Wang et al [20] proposed CNN Explainer, a convolutional neural network visualization system that explains convolution and pooling operations via rich interactive visualizations. It can effectively help deep learning beginners quickly understand complex convolutional neural networks.

Researchers have made great progress in visualizing neural networks. Some of these works attempts to introduce visualization techniques to assist in the analysis of adversarial attacks, they mostly provide limited, static visualization views, lacking rich interactive features and flexible exploration mechanisms. Users find it difficult to dynamically adjust views, filter data, and locate interesting attack patterns according to their analysis needs. In contrast, AdvAttackVis designs multiple visualization views, supporting flexible interactive operations and view coordination, empowering users with full autonomy and flexibility to explore the overview and details of attacks from different perspectives and granularities.

III. Visualization System Overview

In this section, we introduce the overall framework of the AdvAttackVis system. As shown in Fig. 1, the system is designed based on B/S (Browser-Server) architecture. It consists of the front-end and back-end. The two parts are described below.

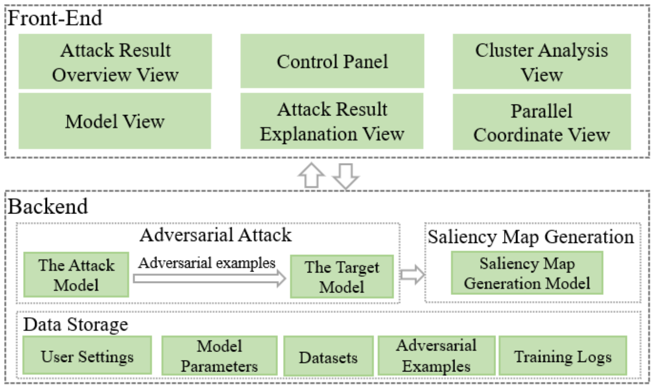


Fig. 1. Framework of AdvAttackVis.

The front-end provides a visual interface to help users understand, analyze, and explore adversarial attacks. It is mainly composed of six visualization views: the attack result overview view, the control panel, the cluster analysis view, the model view, the attack result explanation view, and the parallel coordinate view. The attack result overview view is used to analyze the adversarial attack results including the attack success rate and the distribution of attack results. This view provides a visual analysis of the attack effects from an overall perspective, allowing users to quickly understand the most prominent categories of model vulnerabilities to guide the next step of in-depth analysis. The control panel provides parameter settings for model training and adversarial attacks so that the user can train a satisfactory adversarial attack model. The cluster analysis view is designed to compare the clusters of the original examples and the adversarial examples, which helps users qualitatively analyze the performance changes of the target model. The model view displays the main modules of an adversarial attack model to help users understand the model architecture and working flow. The attack result explanation view offers saliency maps [27] of adversarial examples, which provides a visual explanation of the classification results of the adversarial examples. In the parallel coordinate view, users can analyze the changes in confidence scores of the original example and the adversarial example. In addition, information collaboration is achieved through link interaction between multiple views. Users can select points of interest in one view, and other views automatically update to display the characteristics of these points.

The backend mainly consists of three parts: the adversarial attack module, the saliency map generation module and the data storage module. As the core of the backend, the adversarial attack module contains an attack model and a target model. The former generates adversarial examples to attack the latter. The saliency map generation module generates a saliency map for each adversarial example to explain why the adversarial attack succeeds or fails. The data storage module is used to store public datasets (e.g., training and testing sets), model parameters, adversarial examples, training logs and user settings.

For simplicity, we illustrate our system using the MNIST dataset [28] which is a classic handwritten digit classification dataset. Specifically, we construct a MNIST classification model as the target model and employ AdvGAN as the attack model. In the following sections, we first describe the underlying models of our system, including the target model, attack model and saliency map generation model. Then, we introduce the visual interface of the system. Finally, we demonstrate the effectiveness of our system through case studies.

IV. Underlying Model

In this section, we introduce three main underlying models of the visualization system: the target model, the attack model, and the salient map generation model.

A. Target Model

We construct a MNIST classification model based on convolutional neural network as the target model. The MNIST dataset is very classic and easy to understand. The model structure used in this dataset is also relatively simple, but includes the main components of modern neural networks, making it very suitable for demonstrating the characteristics of deep learning. Therefore, using this dataset helps beginners understand how the model works. Specifically, the MNIST dataset [21] contains ten classes of handwritten digital grayscale images from 0 to 9. It contains 60,000 training images and 10,000 testing images. An MNIST classification model is constructed to recognize handwritten digital images. The model mainly consists of four convolutional layers, two pooling layers, two fully connected layers and a Softmax layer. It employs ReLU [1] as the activation function. Its overall architecture is shown in Table I. We train the MNIST classification model by the Adam optimizer [22] for 30 epochs with the batch size of 128. The model with the highest test accuracy (99.1%) is selected as the target model.

B. Attack Model

We employ AdvGAN [5] as the attack model. Compared to traditional gradient-based attack methods, AdvGAN not only generates more realistic and diverse attack samples more stably and efficiently but also employs a black-box attack approach. This enhances the generalization performance of AdvAttackVis across different attack models. AdvGAN produces adversarial samples using generators of generate adversarial networks (GANs). As shown in Fig. 2, AdvGAN contains three components: a generator G, a discriminator D and a target model f (i.e., the trained MNIST classification model). The generator and discriminator form a GAN. The generator takes the original example x as input. The generated perturbation $G(x)$ is added to the original example x to form an adversarial example $x+G(x)$.

TABLE I. NETWORK ARCHITECTURE OF THE MNIST CLASSIFICATION MODEL

Layer	Parameter
Convolution + ReLU	3×3×32
Convolution + ReLU	3×3×32
Max Pooling	2×2
Convolution + ReLU	3×3×64
Convolution + ReLU	3×3×64
Max Pooling	2×2
Fully Connected + ReLU	200
Fully Connected + ReLU	200
Softmax	10

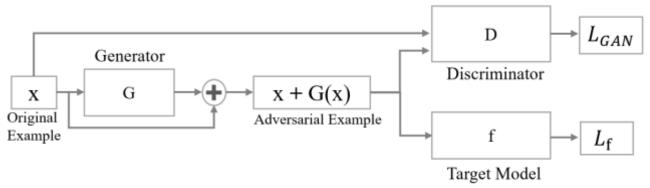


Fig. 2. Framework of the AdvGAN model.

During training AdvGAN, the generator aims to generate fake examples that can fool the discriminator, while the discriminator aims to correctly discriminate fake examples from real examples. The generator and the discriminator confront each other, forcing the generator to eventually generate realistic adversarial examples. To be specific, GAN is trained based on the following adversarial loss:

$$L_{GAN} = E_x logD(x) + E_x\, log\left(1 - D\left(x + G(x)\right)\right) \quad (1)$$

where $D(.)$ represents the probability (i.e., confidence) which the discriminator discriminates the input as a real example. Moreover, AdvGAN needs to constrain the output of the generator. That is, the generated adversarial examples should induce the target model to output wrong predictions. For targeted attacks, the corresponding loss function is as follows:

$$L_{adv}^{f} = E_x l_f(x + G(x), t) \quad (2)$$

where t denotes the target class and l_f represents the cross-entropy loss function. However, for untargeted attacks, t represents the ground truth. The adversarial example can be misclassified into other classes by maximizing the distance between the prediction and the ground truth. To limit the magnitude of the perturbation $G(x)$, AdvGAN usually adds a soft hinge loss to constrain the training process:

$$L_{hinge} = E_x max(0, \|G(x)\|_2 - c) \quad (3)$$

where c indicates the user-specified upper bound and L_{hinge} can stabilize the training process of GAN. Finally, the objective function of AdvGAN is as follows:

$$L = L_{adv}^{f} + \alpha L_{GAN} + \beta L_{hinge} \quad (4)$$

where α and β control the importance of each component, L_{GAN} makes the perturbed example $x+G(x)$ similar to the original example x, and L_{adv}^{f} improves the attack success rate of adversarial examples. We train the generator and the discriminator by solving the min-max game $arg\, min_G max_D\, L$. The trained generator can quickly generate adversarial examples for the original examples.

C. Saliency Map Generation Model

A saliency map [31] can reflect which regions of the input image are important for the decision-making of the target model. That is, the saliency map of an adversarial example can explain why the attack on the target model succeeds or fails from a visual perspective. We construct the saliency map generation model based on Grad-CAM [23]. The saliency map generation algorithm for adversarial examples is shown in Algorithm 1.

***Algorithm 1** Saliency map generation algorithm for antithetical examples*

```
Input: The original example $x$, the generator $G$, the target model $f$
Output: A saliency map of the adversarial example
1. $x' = x + G(x)$ ; // Generate an adversarial example
2. $c = argmax(f(x'))$ ; // Predict the class of the adversarial example.
3. $s = L_{Grad-CAM}^{c}(x')$ ; // Generate a saliency map.
4. $s' = \frac{s - \min(s)}{\max(s) - \min(s)}$ ;
5. return $s'$;
```

Grad-CAM generates saliency maps based on the feature maps of the last convolutional layer of the target model. Specifically, the weight w_k^c of the k-th feature map A^k for category c is the mean of gradients:

$$w_k^c = \frac{1}{z}\sum_i \sum_j \frac{\partial y^c}{\partial A_{ij}^k} \quad (5)$$

where y^c denotes the output of the target model for category c, i and j is the horizontal and vertical coordinates of the feature map A^k respectively, and z denotes the size of the feature map A^k. Then, the weighted feature maps are merged into a saliency map:

$$L_{Grad-CAM}^{c} = ReLU\left(\sum_k w_k^c A^k\right) \quad (6)$$

where ReLU [1] is used to remove negative values of the saliency map.

V. VISUAL INTERFACE

In this section, we introduce the visual interface of our system. As shown in Fig. 3, it mainly includes six visualization views: (A) the attack result overview view, (B) the control panel,

(C) the cluster analysis view, (D) the model view, (E) the attack result interpretation view, and (F) the parallel coordinate view.

A. *The Attack Result Overview View*

This view is used to analyze the attack success rate of the adversarial examples of each category (e.g., categories 0 to 9 of MNIST) and the distribution of the attack results. As shown in Fig. 3(A), a histogram is used to display the attack success rate for each category. The scatter plots show the distribution of attack results for each category, allowing users to inspect which category each adversarial example is identified as. For each scatter plot, we take category as the horizontal coordinate and confidence as the vertical coordinate. Each scatter represents an adversarial example and its color encodes the ground truth of the adversarial example. When users click on a scatter, the scatter is highlighted in black. Meanwhile, the attack result explanation view and the parallel coordinate view present the explanation result and the parallel coordinate plot of the adversarial example, respectively.

B. *The Control Panel*

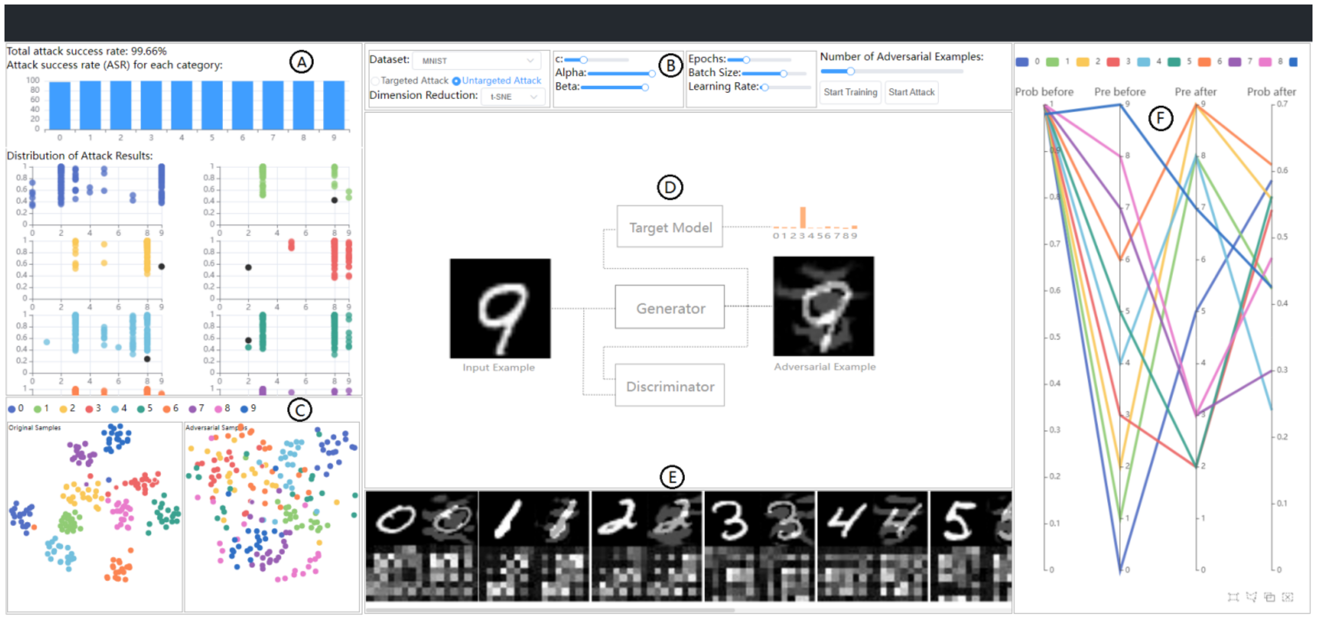
In the control panel, users can adjust the parameters of the attack model and select the dimensionality reduction algorithm for generating the cluster analysis view. When users click the "Start Training" button, the system starts training the attack model and saves the model with the highest success rate. After the attack model is trained, users can adjust the "Number of Adversarial Examples" item to control the number of adversarial examples. When users click the "Start Attack" button, the system generates a specified number of adversarial examples to attack the target model. The attack results are shown in the attack result overview view. The system provides three dimensionality reduction algorithms as options for the cluster analysis view, namely Principal Component Analysis (PCA) [24], Multi-Dimensional Scaling (MDS) [25] and t-distributed Stochastic Neighbor Embedding (t-SNE) [26]. When users select a dimensionality reduction algorithm, the system generates a cluster analysis view by projecting the high-dimensional feature vectors of examples to 2D plane.

C. *The Cluster Analysis View*

This view is designed to compare the clusters of the original examples and the adversarial examples, which allows users to qualitatively analyze the degree of damage to the performance of the target model. As shown in Fig. 3(C), the view consists of two parts. The left shows the clusters of the original examples while the right presents the clustering results of the adversarial examples. Each dot denotes an example and its color encodes the ground truth of the example. In general, the left obtains clear clusters due to the good classification performance of the target model for the original examples. However, the clusters in the right are difficult to be distinguished, which suggests that the adversarial examples interfere with the performance of the target model. Moreover, the view offers interactives (i.e., selection, zooming and highlighting) to facilitate user comparison and analysis of examples.

D. *The Model View*

The model view presents an overall overview of the attack model to help users understand the overall architecture and data flow of the model. For the attack model AdvGAN, we design different rectangular blocks to represents the Generator, Discriminator and target model (e.g., the trained MNIST classification model), respectively. Users can click on a rectangular block to inspect the internal structure of the corresponding module more closely. The small histogram on the right side of the target model displays the classification results of the target model for adversarial examples. During the training phase of the attack model, this view provides the change curve of the attack success rate to help users observe the real-time training situation of the model. When the attack model is trained, users can click on the "Input Example" rectangle to load examples to explore adversarial attacks.


Total attack success rate: 99.66%

Dataset: MNIST

Untargeted Attack

Dimension Reduction: t-SNE

Fig. 3. The visual interface of the AdvAttackVis system.

E. *The Attack Result Interpretation View*

The view can provide explanations for the classification results of the adversarial examples. Specifically, the view generates a saliency map for each adversarial example (see Algorithm 1). The saliency maps reveal the contribution of different regions and pixels of the input image to the model's decision. Intuitively, adversarial perturbations should target those key pixels that influence the model's decision, in order to deceive the model more effectively. Therefore, by visualizing the differences between adversarial and original samples on saliency maps, users can intuitively understand the mechanism of adversarial perturbations, namely how attackers manipulate pixel values subtly to deceive the model. As shown in Fig. 3(D), this view offers the saliency maps of the original example and the adversarial example for comparative analysis. Taking a MNIST image as an example, its saliency map is similar to a mosaic map where the lighter color of a pixel indicates that the target model pays more attention to that pixel. Compared with the saliency map of the original image, that of the adversarial example usually changes dramatically, which suggests that the adversarial example interferes with the decision-making of the target model.

F. *The Parallel Coordinate View*

In the parallel coordinate view, a parallel coordinate plot is designed to help users analyze the classification results and the confidence changes of the adversarial examples. As shown in Fig. 3(E), the plot contains four vertical parallel lines from left to right. They represent the classification confidence of the original example, the classification result of the original examples, the classification result of the adversarial example and the classification confidence of the adversarial example, respectively. A curve in the parallel coordinate plot indicates an adversarial example and its color encodes the class of the adversarial example. As illustrated in Fig. 4, the curve corresponding to the handwritten digit "6" has the values "1.00", "6", "9" and "9" from left to right. This means that the original example is identified as "6" by the MNIST classification model with confidence "100%" but the adversarial example (i.e., the original example with an adversarial perturbation) is misidentified as "9" with confidence "61%".

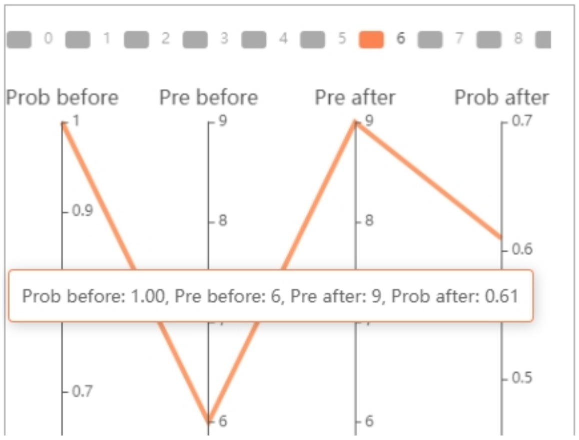


Fig. 4. The curve corresponding to the handwritten digital image "6" in the parallel coordinate view.

VI. Case Study

This section verifies the usability and effectiveness of the visualization system through a case study based on the MNIST dataset.

To train the AdvGAN model, we select the MNIST dataset as the training data in the control panel (Fig. 3(A)) and click on the "Untargeted Attack" item to set the adversarial attack as an untargeted attack. Then we set the parameters of its objective function (i.e., c=0.3, α=1 and β=1) and the parameters of the model training (i.e., epochs=30, batch size=128 and learning rate=0.001). Finally, we click on the "Start Training" item to start model training. In addition, we select the t-SNE dimensionality reduction algorithm to generate the cluster analysis view.

After the AdvGAN model is trained, we set the value of the "Number of Adversarial Examples" item in the control panel to 2075 and click the "Start Attack" button to start the adversarial attack. Specifically, the system first randomly selects 2075 images from the MNIST testing set that can be correctly classified by the MNIST classification model (i.e., the target model), and then inputs them into the trained generator to generate 2075 adversarial examples to attack the MNIST classification model. As shown in Fig. 5, the total attack success rate is 99.6%. The histogram shows that only the adversarial examples with class 0 and class 6 fail against the target model. In other words, the MNIST classification model can correctly predict the category of these adversarial examples. As shown in the red dashed box in Fig. 6, there are six failed examples in the adversarial examples with class 0, while there is only one failed example in the adversarial examples with class 6. Although these seven adversarial examples do not cause the MNIST classification model to output wrong results, they indeed decrease the classification confidence of the MNIST classification model to a certain extent. Specifically, as shown in Fig. 7, the MNIST classification model predicts the classes of the seven failed samples with 100% confidence before adding the adversarial interferences to the original examples. However, when the adversarial interferences are added to the original examples (i.e., generating the adversarial examples), the MNIST classification model correctly classifies the adversarial examples with a maximum of 73% confidence. This means that the addition of the adversarial interferences can interfere with the model's judgment to some extent even if the attack fails.

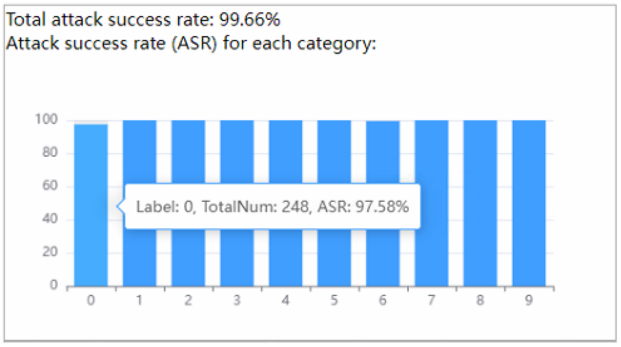


Fig. 5. The total attack success rate of 2075 adversarial examples and the attack success rate for each category.

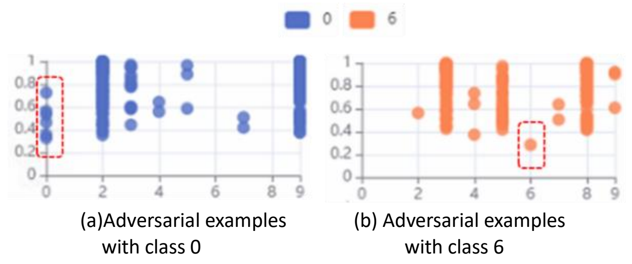


Fig. 6. Seven failed examples in the adversarial examples with class 0 and class 6.

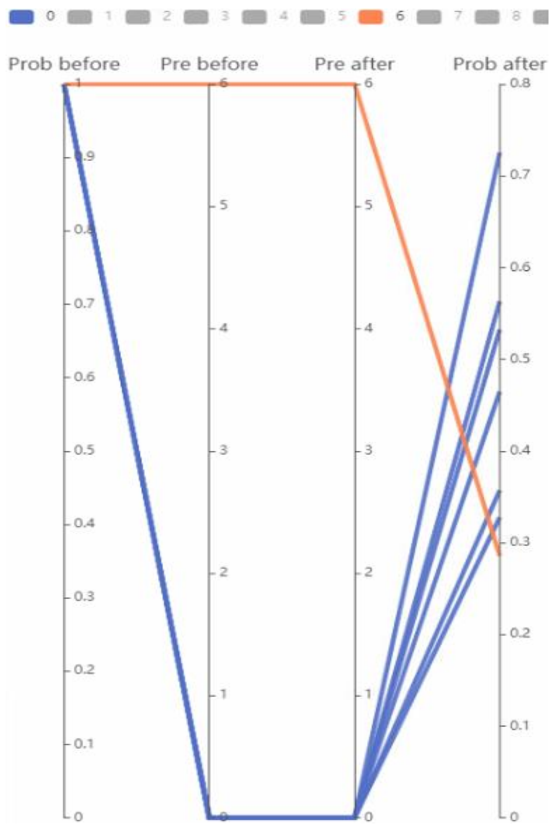


Fig. 7. Confidence change of the seven failed examples before and after the adversarial attack.

The degree of interference suffered by the target model can be analyzed through the cluster analysis view, as shown in Fig. 8. In this case, the left view shows that there are highly distinguishable clusters in the original examples, which shows that the target model is capable of distinguishing classes well before being attacked. However, when the original examples become adversarial examples against the target model, the cluster results of the adversarial examples become cluttered, as shown in the right view of Fig. 8. This indicates that the adversarial attack has a great effect on the category discrimination ability of the target model. For example, the two examples of class “5” (i.e., “2248” and “2250”) in the left view have similar features and thus are close to each other. When these two examples become adversarial examples, their positions in the right view are far apart. This means that the addition of adversarial perturbations has a significant impact on their features, thus causing them to be misclassified by the target model.

From the scatter plots in the attack result overview view (Fig. 3(A)), we can find that most of the adversarial examples are misclassified by the MNIST classification model with high confidence (i.e., large values on the vertical coordinate). In order to explore the successful adversarial examples, we select a successful adversarial example with low confidence (i.e., a small value on the vertical coordinate) from each category (i.e., each scatter plot) for further analysis. Fig. 9 shows the confidence changes of the selected 10 adversarial examples before and after adding adversarial disturbances. Specifically, each example is correctly classified by the MNIST classification model with a confidence level close to or equal to 100% before adding the adversarial interference. However, they are incorrectly classified into other categories by the model with a lower confidence level after adding the adversarial interferences to them. For example, the adversarial example with category 4 is classified as 9 with 24% confidence. For successful untargeted attacks, the low confidence level indicates that the making decision of the MNIST classification model is influenced by the adversarial disturbances. To further explore the reasons why the selected 10 adversarial examples (see Fig. 9) are misclassified, we analyze the saliency maps of these examples in the attack result interpretation view. As shown in Fig. 10, there are a total of 10 pictures. Each picture is composed of 4 small pictures. They are the original image (row 1, column 1), the adversarial example (row 1, column 2), the saliency map of the original image (row 1, column 2), and the saliency map of the adversarial example (row 1, column 2). For each saliency map, the lighter the color of a pixel is, the more attention the MNIST classification model pays to the pixel (i.e., the more important the pixel is to the final classification result). Comparing the saliency map of the original image with that of the adversarial examples, we can find that when adversarial perturbations are added to the original image (i.e., producing an adversarial example), the attention area of the MNIST classification model changes, resulting in the predictions of the model to be disturbed. For example, for the original image "0", the pixel color of the contour area of digital "0" in its saliency map is lighter, which reveals that the MNIST classification model mainly focuses on the contour area of digital "0". However, when the original image becomes an adversarial example, the white pixels in the upper-left region of digital "0" in the adversarial example’s saliency map become denser. This reveals that the MNIST classification model shifts its attention to the upper-left region of digital "0", resulting in the incorrect prediction.

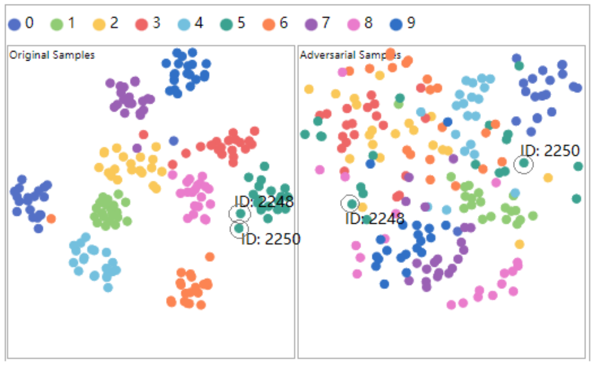


Fig. 8. Cluster results of the original examples and the adversarial examples.

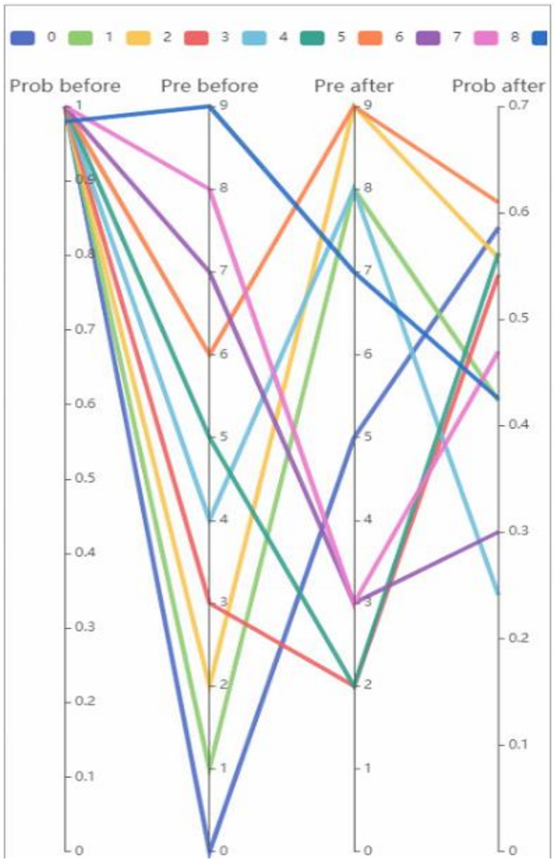


Fig. 9. The parallel coordinate plot of the selected 10 successful adversarial examples.

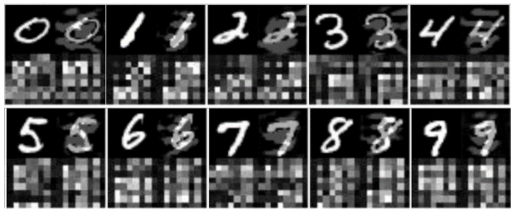


Fig. 10. Interpretation results of 10 successful attack samples.

VII. Conclusion and Future Work

In this paper, we present an adversarial attack visualization system called AdvAttackVis which can effectively assist users in learning, understanding, and exploring adversarial attacks. The visualization system offers interactive visualization views to help users train and analyze the adversarial attack models, understand the principles of adversarial attacks, analyze the success rate of adversarial attacks and the distribution of attack results, and explore the prediction mechanism of the target model for adversarial examples. It facilitates end-to-end analysis of adversarial attacks. Our multi-view visual analysis environment enables users to gain a deep understanding of the effects and impacts of adversarial attacks. Additionally, the system incorporates interpretability analysis techniques, such as saliency analysis, which can assist users in inferring the reasons behind adversarial attacks. Through the case study base on the MNIST handwritten digital image dataset, we demonstrate the usability and effectiveness of the system.

In the future, we will implement more interactive visualizations to help users explore the internal states of the target model. For example, by analyzing the changes in feature maps of each convolutional layer layer by layer, users can track how misclassifications caused by attacks accumulate and propagate within the model, deepening their understanding of the underlying mechanisms of adversarial attacks and enhancing their analytical abilities. Through interactive visualization of the internal states of the model, users can gain insights into the intermediate processes of model inference, identify anomalous behaviors introduced by attacks, and infer their causes and impacts.

Acknowledgment

This research work was partly supported by Humanities and Social Sciences Research Project of the Ministry of Education(Grant No.22YJA840004) and Natural Science Foundation of Zhejiang Province(Grant No. LGF19G010001) and Basic Project of Strengthening Police by Science and Technology of the Ministry of Public Security (Grant No.2020GABJC35).

References

[1] Szegedy C , Zaremba W , Sutskever I , et al. Intriguing properties of neural networks[J]. Computer Science, 2013.

[2] Goodfellow I J , Shlens J , Szegedy C . Explaining and Harnessing Adversarial Examples[J]. Computer Science, 2014.

[3] MADRY A, MAKELOV A, SCHMIDT L, et al. Towards deep learning models resistant to adversarial attacks[J]. arXiv: 1706. 06083, 2017.

[4] KURAKIN A, GOODFELLOW I, BENGIO S. Adversarial examples in the physical world[C]. ICLR(workshop). 2017.

[5] XIAO C, LI B, ZHUJ Y, et al. Generating Adversarial Examples with Adversarial Networks[C]. Proceedings of the 27-th Inter-national Joint Conference on Artificial Intelligence, 2018: 3905-3911.

[6] GLOROT X, BORDES A, BENGIO Y. Deep Sparse Rectifier Neural Networks[J]. Journal of Machine Learning Research, 2011, 15: 315-323.

[7] ARPIT D, JASTRZEBSKI S, BALLAS N, et al. A closer look at memorization in deep networks[C]. Proceedings of the 34th International Conference on Machine Learning Volume 70. JMLR. org, 2017: 233-242.

[8] AKHTAR N, MIAN A. Threat of Adversarial Attacks on Deep Learning in Computer Vision: A Survey[J]. IEEE Access, 2018, 6:14410-14430.

[9] ZhOU Y, HAN M, LIU L, et al. The Adversarial Attacks Threats on Computer Vision: A Survey[C]. Proceedings of The 16th IEEE International Conference on Mobile AdHoc and Smart Systems. IEEE, 2019.

[10] GILMER J, METZ L, FAGHRI F, et al. Adversarial Spheres[J]. arXiv preprint arXiv: 1801.02774, 2018.

[11] GILMER J, METZ L, FAGHRI F, et al. The relationship between high dimensional geometry and adversarial examples[J]. arXiv preprint arXiv: 1801.02774, 2018.

[12] CARLINI N, WAGNER D. Towards evaluating the robustness of neural networks[C]. IEEE Symposium on Security and Privacy (SP). IEEE, 2017: 39-57.

[13] KURAKIN A, GOODFELLOWI, BENGIO S, et al. Adversarial examples in the physical world[C]. International Conference on Learning Representations, 2017.

[14] Su J, VARGAS D V, Kouichi S. One pixel attack for fooling deep neural networks [J]. IEEE Transactions on Evolutionary Computation, 2017.

[15] SARKAR S, BANSAL A, MAHBUB U, et al. UPSET and ANGRI: Breaking High Performance Image Classifiers [J]. 2017.

[16] JIAWEI Z, Yang W, et al. Manifold: A Model-Agnostic Framework for Interpretation and Diagnosis of Machine Learning Models.[J]. IEEE Transactions on Visualization & Computer Graphics, 2018.

[17] KAHNG M, THORAT N, CHAU D H, et al. GAN Lab: Understanding Complex Deep Generative Models using Interactive Visual Experimentation[J]. IEEE Transactions on Visualization & Computer Graphics, 2018.

[18] KWON B C, CHOI M J, KIM J T, et al. RetainVis: Visual Analytics with Interpretable and Interactive Recurrent Neural Networks on Electronic Medical Records[J]. IEEE Transactions on Visualization & Computer Graphics, 2018:1-1.

[19] Liu M, Shi J, Zhen L, et al. Towards Better Analysis of Deep Convolutional Neural Networks[J]. IEEE Transactions on Visualization & Computer Graphics, 2017, 23(1):91-100.

[20] Wang Z J, Turko R, Shaikh O, et al. CN-N Explainer: Learning Convolutional Neural Networks with Interactive Visualization [J]. 2020.

[21] Lecun Y, Bottou L. Gradient-based learning applied to document recognition[J]. Proceedings of the IEEE, 1998, 86(11):2278-2324.

[22] KINGMA D, BA J. Adam: A Method for Stochastic Optimization[J]. Computer Science, 2014.

[23] SELVARAJU R R, COGSWELL M, DASA, et al. Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization[J]. International Journal of Computer Vision, 2020, 128(2):336-359.

[24] WOLD, S., ESBENSEN, K.; GELADI, P. Principal component analysis. Chemom. Intell. Lab. Syst. 1987, 2, 37–52.

[25] KRUSKAL, J. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. Psychometrika 1964, 29, 1–27.

[26] MAATEER, L.V.D.; HINTON, G.E. Visualizing Data using t-SNE. J. Mach. Learn. Res. 2008, 9, 2579–2605.

[27] Arun N, Gaw N, Singh P, et al. Assessing the (Un)Trustworthiness of Saliency Maps for Localizing Abnormalities in Medical Imaging:, 10.1101/2020.07.28.20163899[P]. 2020.

[28] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner: Gradient-Based Learning Applied to Document Recognition, Proceedings of the IEEE, 86(11):2278-2324, November 1998.

[29] Matthew H F, Minsuk K, Robert P, et al. Visual Analytics in Deep Learning: An Interrogative Survey for the Next Frontiers: IEEE, 10.1109/TVCG.2018.2843369[P]. 2018.

[30] Yuan J, Chen C,　Yang W, et al. A survey of visual analytics techniques for machine learning[J]. Computational Visual Media, 2021, 7(1):3-36.

[31] Itti, Laurent, Koch, et al. A Model of Saliency-Based Visual Attention for Rapid Scene Analysis.[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 1998.