CS 590 Final Project: Gradient Ascent Adversarial

Frank Yin, Xiaoxuan Zhang, Yunfan Zhang

April 20, 2019

1 Introduction

For this final project, we will be investigating the impacts of gradient ascent neural network on computer security. Specifically, we will analyze and demonstrate how a fooling image can be generated from an authentic image to mislead the trained model into false categorization with only minimal changes to the original image. We will demonstrate its functionality by building a website that allows users to upload images, select target classes and generate fooling images that misguide the model into a false category.

Before diving into the details and procedures of the gradient ascent applications, let us take a closer look at the mechanisms under the hood for better understanding. Lots of the image categorization programs use deep learning to train and recognize the objects in pictures. Deep learning is part of a broader family of machine learning methods based on learning data representations, and it can be supervised, semi-supervised or unsupervised. Deep learning architectures such as deep neural networks, deep belief networks and recurrent neural networks have been applied to fields including computer vision, speech recognition, natural language processing, audio recognition, social network filtering, machine translation, bioinformatics, drug design, medical image analysis, material inspection and board game programs, where they have produced results comparable to and in some cases superior to human experts. In deep learning, each level learns to transform its input data into a slightly more abstract and composite representation. In an image recognition application, the raw input may be a matrix of pixels; the first representational layer may abstract the pixels and encode edges; the second layer may compose and encode arrangements of edges; the third layer may encode a nose and eyes; and the fourth layer may recognize that the image contains a face. Importantly, a deep learning process can learn which features to optimally place in which level on its own.

The model we use in our website is "squeezenet1_1", a computer vision model that categorizes images into different classes implemented with pytorch. This is the target model that we have access to on our website. Based on the characteristics and features of this trained model, we would operate gradient ascent on the images (chosen by users) to fool the model and to output false result.

2 Theory

Although neural networks perform very well for image recognition, the delicate nature of deep neural networks makes them very easily fooled by adversarial attacks. Scholars were able to generate "minimimum distortions" of images for which the neural networks output different results in as early as 2014[1]. The procedure to produce a "minimally distorted" image that confuses the neural network is called gradient ascent, and is very similar to the gradient descent steps used to train neural networks. The only difference is that in gradient descent, the input image is fixed and we update parameters, while in gradient ascent we fix the input parameters and update the image. Consider a neural work model:

$$S_\theta(I)$$

where θ is the parameters of the neural network and I is the input image. The classes we want the neural network to learn are indexed as integers starting from 0. This model outputs a vector containing the raw scores of classes corresponding to the indices, before the soft-max layer. The higher the score, the more the neural network thinks the image is from a certain class. Suppose we want the neural network to classify an image into a target class of our choice, c. Because we want The score for this class to be as large as possible, we want to find:

$$\arg\max_I S^c_\theta(I)$$

where S^c is the score that the neural network outputs for class c. To find the target image I that will confuse the neural network, we do iterative gradient ascent on the image. We calculate the gradient of the input image pixels with respect to the target class score, and add a multiple of that gradient to the image:

$$I = I + \lambda \cdot \frac{\partial S^c_\theta(I)}{\partial I}$$

where λ is the learning rate. We repeat this step until the image is correctly classified by the neural network.

This is very easy to do in practice, and in our experiment we are usually able to find an image that tricks the neural network in a few steps. In Pytorch, we simply set the **requires_grad_()** property of the input image to be true. This enables Pytorch to track the gradients of the image. The image goes through the model, and the model outputs a score vector **out**. To find the gradient of the image with respect to the c th output, we simply call **out[c].backward()**.

For implementation, we used a pretrained neural network model **squeezenet1_1** from **torchvision**. This is a small network that runs very fast, so gradient ascent doesn't take too much time on a laptop. We used image loading, preprocessing and deprocessing methods from Stanford's CS231n assignment 2 for convenience.

[1] Szegedy et al, "Intriguing properties of neural networks", 2013

3 Procedure

View and Flask are used to build the website where the gradient ascent algorithm and the trained deep learning model run on.

The front end of the website is implemented with View and consists of the following sections with descriptions on their functionalities:

-

The back end of the website is implemented with Flask and consists of the following sections with descriptions on their functionalities:

-

The implementation details of the image confusion algorithm are described below. It starts from an arbitrary image, picks an arbitrary class, modify the image to maximize the class and repeat the process until the network is fooled.

The following steps are taken to make the image modification on the website:

- Choose an image to upload to the website from the local machine (cat, dog, fish, etc.)
- A classification/categorization is outputted to the user with a confidence level
- Choose a target class that the user intends to fool the network to categorize the modified image as
- A false classification/categorization is outputted to the user with a confidence level, as well as the modified image with no human-conceivable differences
- Save the modified image to the local machine and re-submit it on the website to confirm that the network is fooled

4 Results

In this section, we will be displaying some of the results generated by this gradient ascent algorithm and their corresponding outputs based on the network.

Hay or Black Swan?

As shown below in Figure 1.1, the original picture of hay is successfully classified by the network to be hay with a confidence level of 0.99012256. However, the fooling image (Figure 1.2) looks just like the original picture but gets classified as black swan with a confidence level of 0.8915755. The pixel difference graph is illustrated in Figure 1.3 but displays no obvious changes. For better visualizations, the pixel difference graph with 10x contrast is illustrated in Figure 1.4 to display the minor changes in the modified picture.

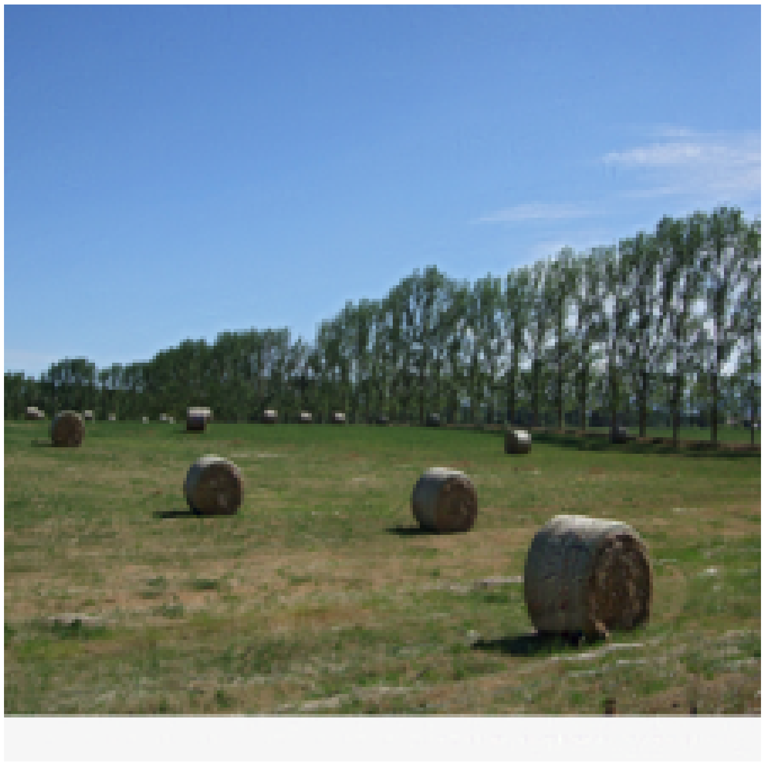

Figure 1.1. Hay

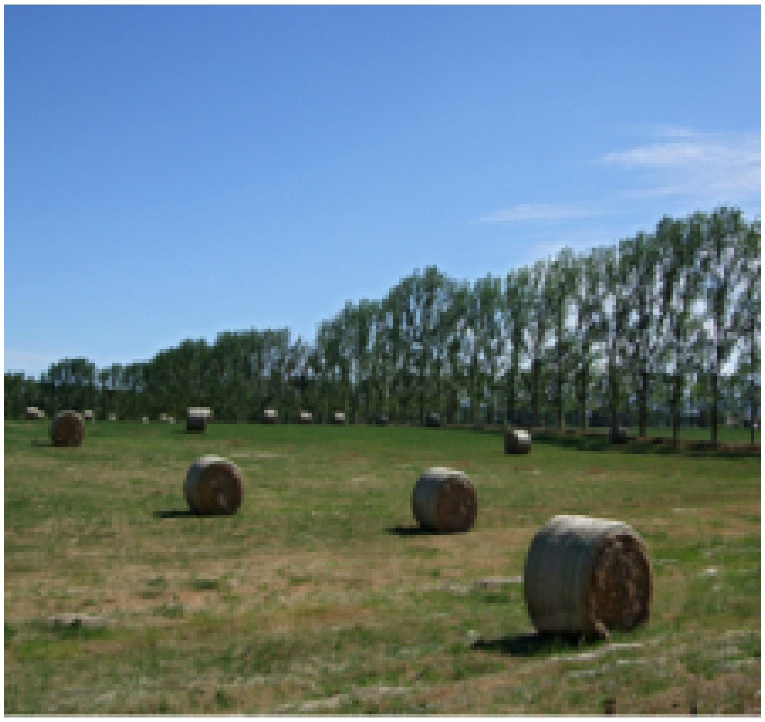

Figure 1.2. Black Swan

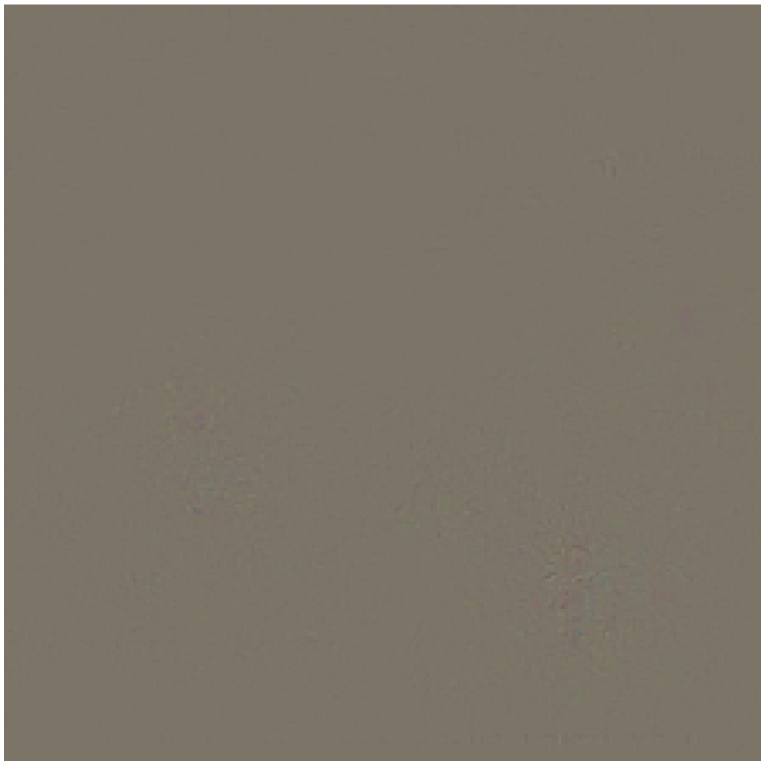

Figure 1.3. Hay vs. Black Swan Difference Graph

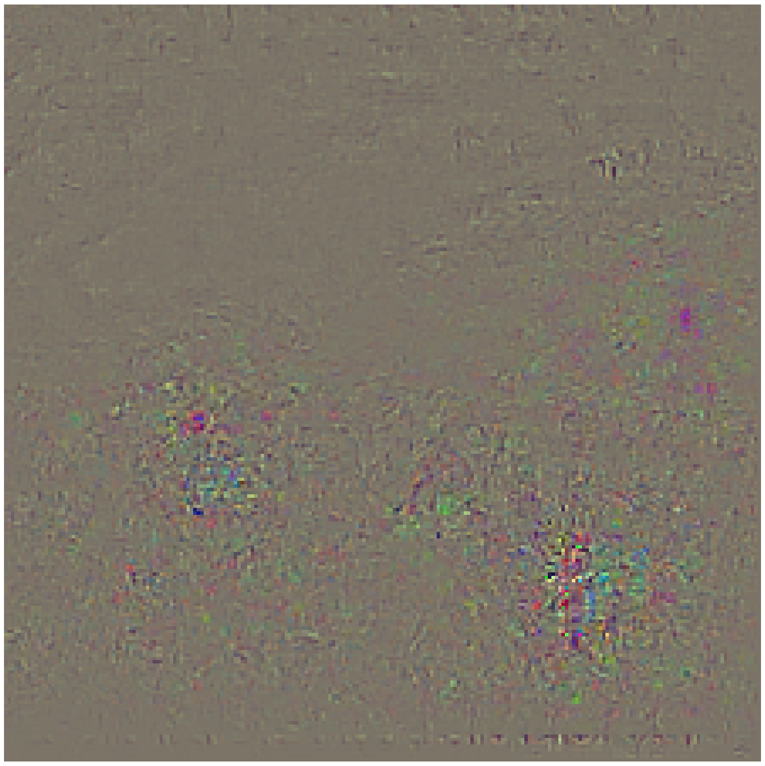
Figure 1.4. Hay vs. Black Swan Difference Graph with 10x Contrast

Quail or Assault Rifle?

As shown below in Figure 2.1, the original picture of quail is successfully classified by the network to be quail with a confidence level of 0.70830715. However, the fooling image (Figure 2.2) looks just like the original picture but gets classified as assault rifle with a confidence level of 0.9930503. The pixel difference graph is illustrated in Figure 1.3 but displays no obvious changes. For better visualizations, the pixel difference graph with 10x contrast is illustrated in Figure 1.4 to display the minor changes in the modified picture.

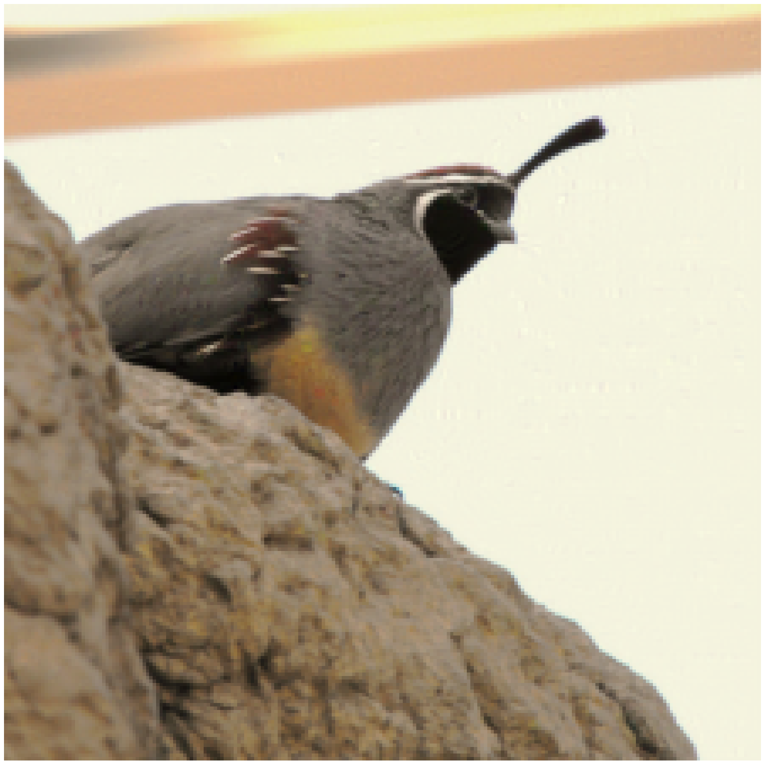

Figure 2.1. Quail

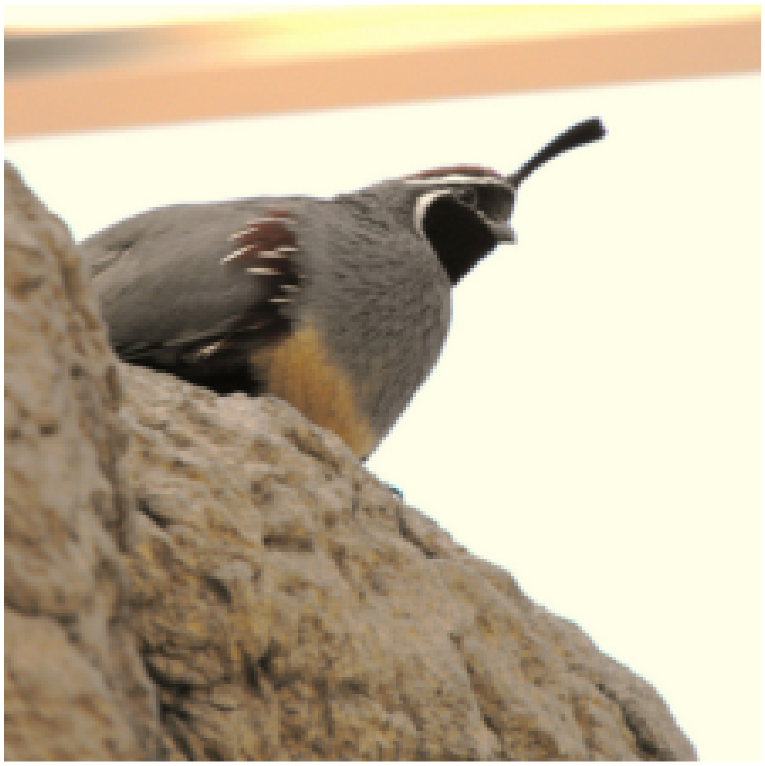
Figure 2.2. Assault Rifle

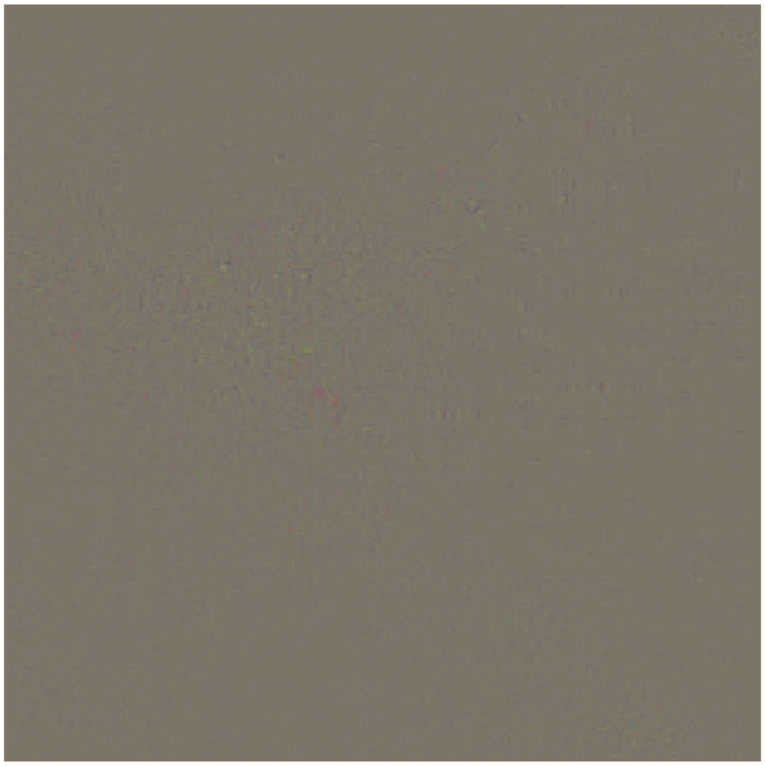
Figure 2.3. Quail vs. Assault Rifle Difference Graph

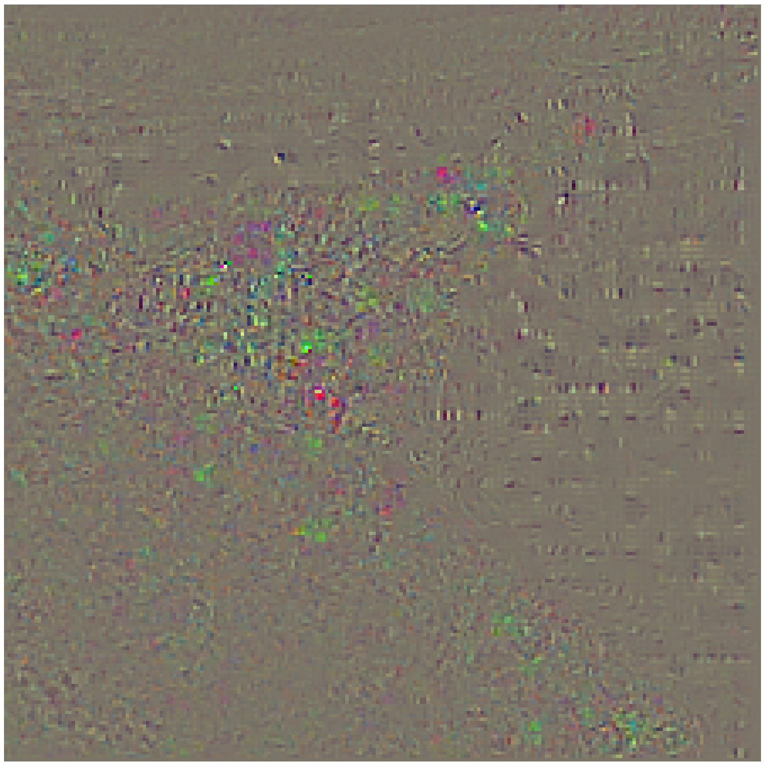
Figure 2.4. Quail vs. Assault Rifle Difference Graph with 10x Contrast

Poncho or Toilet Paper?

As shown below in Figure 3.1, the original picture of poncho is successfully classified by the network to be poncho with a confidence level of 0.47271538. However, the fooling image (Figure 3.2) looks just like the original picture but gets classified as toilet paper with a confidence level of 0.95665723. The pixel difference graph is illustrated in Figure 3.3 but displays no obvious changes. For better visualizations, the pixel difference graph with 10x contrast is illustrated in Figure 3.4 to display the minor changes in the modified picture.

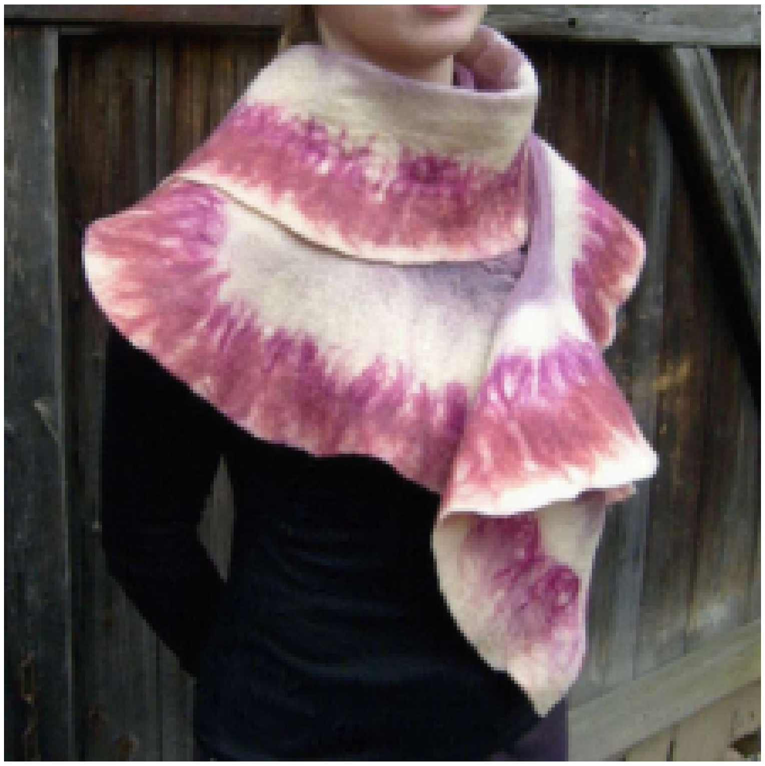

Figure 3.1. Poncho

Figure 3.2. Toilet Paper

Figure 3.3. Poncho vs. Toilet Paper Difference Graph

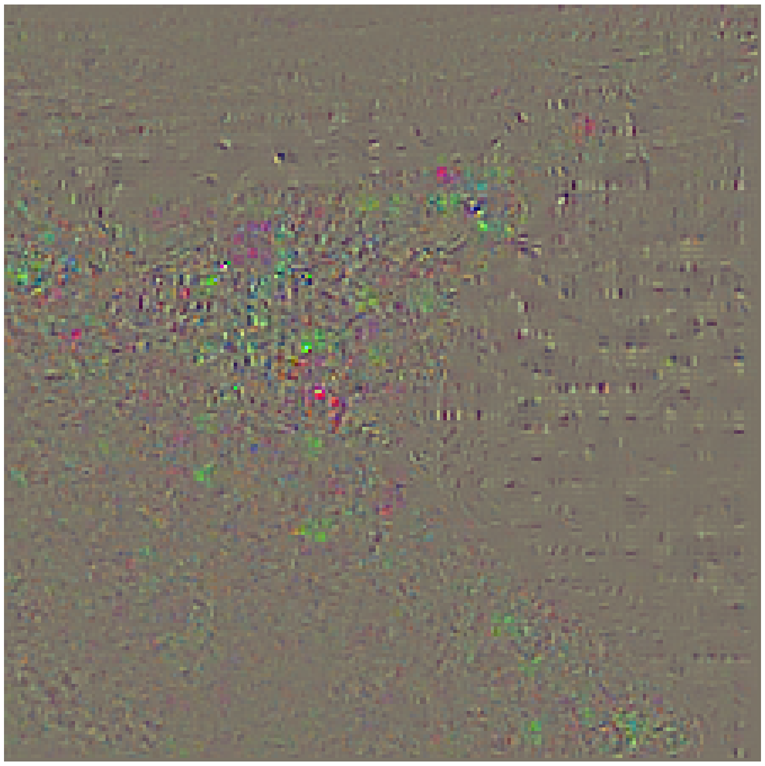

Figure 3.4. Poncho vs. Toilet Paper Difference Graph with 10x Contrast

5 Discussion

As illustrated above, it is easily observed that neural network is a non-interpretable black box and can have unexpected, manipulated behaviors. Given the parameters of neural network models, the output of modified inputs can be changed and designed to a false categorization.

Attacking a neural network can have serious real-world consequences. A lab from Tencent has demonstrated that they can process an image digitally to trick Tesla's autopilot into thinking that it is raining[1]. They used an algorithm called Particle Swarm Optimization, and similar to our experiment, the difference between the original and processed images are indistinguishable by humans. They also displayed the noises calculated by the Particle Swarm Optimization algorithm on a LED screen in the real world, and also successfully tricks the machine learning algorithm into thinking that it is raining. Although tricking autopilot into thinking it is raining is benign, attackers could trick the autopilot to crash into a house or a passenger.

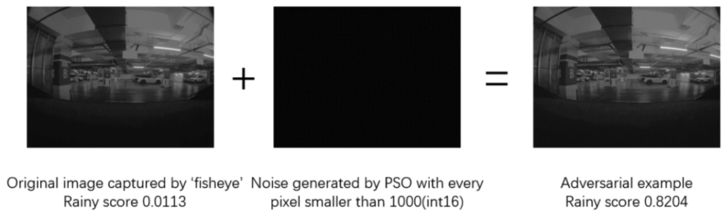
+
=
Original image captured by 'fisheye'
Rainy score 0.0113
Noise generated by PSO with every
pixel smaller than 1000(int16)
Adversarial example
Rainy score 0.8204

Figure 3.5. Original v.s. Adverserial Images by Tencet

Because neural network is a non-intepretable black box and is vulnerable to adversarial attacks, under no circumstances should we fully trust the decisions made by the neural network. Moreover, our experiment demonstrated that it is critical not to expose neural network parameters to public because it is fairly straightforward to perform adversarial attack if an attacker knows the parameters of a neural network. The cautiousness in protecting private keys should also be put in protecting neural network model parameters.

- [1] https://keenlab.tencent.com/en/whitepapers/Experimental_Security_Research_of_Tesla_Autopilot.pdf