2020

The Susceptibility of Deep Neural Networks to Natural Perturbations

Mesut Ozdag
University of Central Florida

THE SUSCEPTIBILITY OF DEEP NEURAL NETWORKS TO NATURAL PERTURBATIONS

by

MESUT OZDAG
M.S. University of Central Florida, 2015
B.S. Istanbul Kültür University, 2010

A dissertation submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy
in the Department of Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Spring Term
2020

Major Professor: Sumit Kumar Jha

ABSTRACT

Deep learning systems have achieved great success in various types of applications in recent years. They are increasingly being adopted for safety-critical tasks such as face recognition, surveillance systems, speech recognition, and autonomous driving. On the other hand, it has been found that deep neural networks (DNNs) can easily be fooled by adversarial input samples. These imperceptible perturbations on images can lead any machine learning system to misclassify the objects with high confidence. Furthermore, they can be almost indistinguishable to a human observer. These systems can also be exposed to adverse weather conditions such as fog, rain, and snow. This vulnerability raises major concerns in security-sensitive environments. Therefore, vulnerability of deep learning systems to synthetic adversarial attacks has been extensively studied and demonstrated, but the impact of natural weather conditions on these systems has not been studied in detail.

The main contribution of this thesis is exploring the effects of fog on classification accuracy of the popular Inception deep learning model. We use stereo images from the Cityscapes dataset and computer graphics techniques to mimic realistic naturally occurring fog. We show that the Inception deep learning model is vulnerable to the addition of fog in images.

We also review the types of adversarial attacks and defenses, describe the state-of-the-art methods for each group, and compare their results.

Adversarial images can be used to generate targeted attacks or non-targeted attacks. Targeted attacks misguide the deep learning networks to produce responses from a specific *a priori* determined class. In non-targeted attacks, all images in the dataset are not assigned to a specific class; instead, the output of the deep neural network is arbitrarily wrong. In this thesis, we create non-targeted, iterative, and physical attacks.

ACKNOWLEDGMENTS

First and foremost, my sincerest gratitude goes to my supervisor, Dr. Sumit Kumar Jha, who has supported me throughout my Ph.D. studies with his great patience and knowledge. I would also like to acknowledge Dr. Ulas Bagci, Dr. Murat Yuksel, and Dr. Samiul Hasan for their valuable suggestions and their kind acceptance to be my Ph.D. dissertation committee members. I am also thankful to Sunny Raj, Drs. Steven Fernandes, Alvaro Velasquez, and Laura L. Pullum for their inspired guidance and help during my research studies. I would like to acknowledge my colleagues and my research group, in particular for their warm companionship and encouragement. I convey special thanks to all my friends here in Orlando and back home. Last but not least, I am eternally grateful for my mother Cevahir, father Muhittin, and sister Tugba. Without their love and encouragement, I would never have completed this degree.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

Deep learning algorithms demonstrate great achievements in various pattern recognition applications and image classification problems. With recent advancements in high-performance graphical processing units and the availability of a large number of classified images, deep learning networks have become even better at image recognition tasks than an average human being.

Despite these outstanding success stories, it has been repeatedly shown that deep learning networks produce incorrect responses when the input is perturbed by small but intelligently crafted “adversarial" changes. For example, such adversarial images can easily cause even state-of-the-art deep learning networks to erroneously classify the images [6, 7, 8, 3]. In many cases, the modifications to the input images are so small that the original images are nearly indistinguishable from the adversarial images to an average human eye. Adversarial inputs pose a real challenge to the successful adoption of deep learning in safety-critical applications. Adversarial attacks on deep learning networks can affect fingerprint and face recognition tasks, as well as cause errors in speech recognition systems, and other applications.

Adversarial attacks can also be classified based on the number of times an input is analyzed during the crafting of the adversarial input. One-time attacks utilize only a single access to the inputs to create the adversarial images. Iterative attacks require multiple accesses to the input image as they create and refine the adversarial images. Perturbations used to generate adversarial images can be broadly classified as *digital* and *physical*. Digital attacks are based on modification of the input image in the memory of a computer that may or may not correspond to an image in the real world, while physical attacks are based on images that can be acquired from the physical world.

(a) Original image is correctly classified as a minivan by the Inception-v3 model.

(b) Image with fog is incorrectly classified as a fountain by the Inception-v3 model.

Figure 1.1: The addition of fog to an image causes the Inception-v3 model to incorrectly classify images that would be correctly classified by a human user [1].

In this thesis, we create non-targeted, iterative, and physical attacks. Our experiments demonstrate that the addition of synthetically generated fog to real-world images causes deep learning networks to incorrectly classify images [1]. Unlike adversarial images, our inputs are not crafted maliciously by choosing careful random perturbations. Instead, our inputs are merely generated using the synthetic addition of fog; hence, such images can be expected to occur in the real world. We present a small but essential step towards demonstrating the need to design much more robust machine learning systems for safety-critical applications.

In addition, we demonstrate the solution models, experiments, and results on some of the types of adversarial attacks and defenses in the literature [9]. Their contributions are significant on this era of machine learning and DNNs.

CHAPTER 2: LITERATURE REVIEW

2.1 Adversarial Attacks and Defenses

The typical purpose of an adversarial attack is to add a natural perturbation (i.e. fast gradient sign, fog, and sunlight etc.) on an image so that the target model misclassifies the sample, but it is still correctly classified by the human eye. Three main types of studies in adversarial attacks have been mainstream in this area:

- *Non-targeted adversarial attacks* are developed to fool a machine learning classifier by modifying source image. The neural network does not return a certain class as opposed to targeted attacks. The output can be a random class excluding the original one.
- *Targeted adversarial attacks* are designed to misclassify an image as a specified target class by modifying source image. The output of this neural network is only one certain class. Impersonation can be an example for this type of attacks because an adversarial image can disguise a face as an admin user.
- *Defenses against adversarial attacks* are aimed to build such a robust classifier that it correctly identifies adversarial images.

Recent research studies on the generation of adversarial samples introduce some new methods. In this thesis, the most well-known and state-of-the-art attacks and defenses are presented [9]:

2.1.1 Fast Gradient Sign Method (FGSM)

One practical way of generating adversarial examples was suggested by their linear behavior [10]. It is observed that the models that are linear are easy to be perturbed. Therefore, adversarial training is processed in a fast and simple way with this method. The goal of fast gradient sign method (FGSM) is to obtain a different classification result (i.e. GoogLeNet) from the original one by adding an unnoticeably small vector. The elements of the vector are extracted by taking the sign of the gradient of a loss function associated with that feature vector:

$$\eta = \epsilon \, \text{sign}(\nabla_x J(\theta, x, y)) \, . \tag{2.1}$$

Let $J(\theta, x, y)$ be the cost function for training a neural network, η be the perturbation, θ be the parameters, x be the input, and y be the targets to the model. The cost function can be obtained as linear which causes an optimal perturbation as in the equation 2.1. It was demonstrated that this method reliably misclassifies the model inputs [10].

A gradient descent is an optimization function that can be used to find a local minimum for a differentiable function. First of all, current adversarial samples are described for linear structures. The individual input feature has limited precision in many cases; for instance, the digital images with 8 bits per pixel can dismiss all the information below $\frac{1}{255}$ of the dynamic range. Since there is limited precision for any feature, it is not logical for the classifier to return a different class for an input x from to an adversarial input $x' = x + \eta$, where η is the perturbation of each element less than the precision of features. Technically, for the cases which result from well-separated outputs, the classifier is expected to give the same output to x and x' as long as $\|\eta\|_\infty < \epsilon$, where ϵ is small enough so that it can be dismissed by the sensor or data storage apparatus related to the case. Let

a weight vector be *w*, an adversarial sample *x*, and consider the *dot product* between them:

$$w^T \mathrm{x'} = w^T \mathrm{x} + w^T \eta \ . \tag{2.2}$$

In the equation 2.2, the activation increases linearly by $w^T \eta$ due to the adversarial perturbation. This demonstrates that a basic linear model can provide adversarial samples if there is enough dimension of its inputs:

$$\begin{aligned} \mathrm{g_{t+1}} &= \mu \ . \ \mathrm{g_t} + \frac{J_\theta(x'_\mathrm{t}, y')}{\|\nabla_\mathrm{x} J_\theta(x'_\mathrm{t}, y')\|_1} \ , \\ \mathrm{x'_{t+1}} &= \mathrm{x'_t} - \alpha \ . \ \mathrm{sign}(\mathrm{g_{t+1}}) \ . \end{aligned} \tag{2.3}$$

The gradient and adversarial samples are calculated in the given equation 2.3 according to the FGSM with momentum, where *y'* is the misclassified target class and $\alpha = \epsilon / L$ with L being the total number of iterations. The effectiveness is boosted by this method with using the momentum term into the iterative process.

2.1.2 Projected Gradient Descent (PGD)

A multi-step variant of FGSM proposed by Madry et al. [4] projects gradient descent (PGD) on the negative loss function, and it is a more powerful adversary than FGSM itself:

$$x^{t+1} = \pi_{\mathrm{x+S}} \left(x^t + \alpha \ \mathrm{sign}(\nabla_\mathrm{x} \mathrm{J}(\theta, \mathrm{x}, \mathrm{y})) \right) . \tag{2.4}$$

In the equation 2.4, π is the projection of a point onto the set *x+S*; where *x* is a data point and *S* is a set of allowed perturbations. FGSM does not increase robustness. More specifically, the network might overfit to the adversarial examples generated by FGSM, which is called *label leaking*. These networks do not demonstrate any robustness against PGD adversaries. Therefore, PGD was the strongest attack using the local first-order information about the network [4]. Moreover, FGSM is

Table 2.1: Performance results [4] of the adversarially trained model against each adversaries for $\epsilon = 0.3$ using MNIST (A: the network itself - white-box attack).

Method	**Steps**	**Restarts**	**Source**	**Accuracy**
Natural	-	-	-	98.8%
FGSM	-	-	A	95.6%
PGD	40	1	A	93.2%
PGD	100	1	A	91.8%
PGD	40	20	A	90.4%
PGD	100	20	A	**89.3%**
Targeted	40	1	A	92.7%
CW	40	1	A	94.0%
CW+	40	1	A	93.9%

a one-step approach and there exists some shortcomings for this method. However, PGD is very useful because it works very well for the large scale constrained optimization.

Generally, the PGD technique is restarted from many points in the l_∞ balls around some data points from a given evaluation set to explore a large part of the loss landscape. In the loss function of model parameters, the value of all the local maxima is similar. These values are the same for both type of networks; normal networks and adversarially-trained networks. Robustness against the PGD adversary gives robustness against all the first-order adversaries, which means it is the attack that would depend upon the first-order information only. When the gradients of the loss functions are used in the case of adversary with regard to different input values, it is concluded that this might not find any significantly better local maxima than PGD.

For the results demonstrated in the Table 2.1, a network consisted of a convolutional layer with 32 filters, one 2 x 2 max-pooling, another convolutional layer with 64 filters followed by another 2 x 2 max-pooling, and lastly one fully-connected layer of size 1024 was used. It was observed that PGD was the best attack. As for the Table 2.2, ResNet and its ten times wider variant were used to train the network and PGD was observed as the most effective attack.

Table 2.2: Performance results [4] of the adversarially trained model against each adversaries for $\epsilon = 8$ using CIFAR10 (A: the network itself - white-box attack).

Method	**Steps**	**Source**	**Accuracy**
Natural	-	-	87.3%
FGSM	-	A	56.1%
PGD	7	A	50.0%
PGD	20	A	**45.8%**
CW	30	A	46.8%

2.1.3 Basic Iterative Method (BIM)

Basic Iterative Method (BIM) is an iterative version of FGSM. In other words, adversarial noise is iteratively applied multiple times instead of applying it only once [11]. Then, pixel values are clipped after each step so that they are ensured to be in an ϵ-*neighborhood* of the clean data. The number of iterations can be selected during the experiments:

$$\text{Clip}_{x,\,\epsilon}(x') = \min(255,\ x+\epsilon,\ \max(0,\ x - \epsilon,\ x'))\ , \tag{2.5}$$

In the equation 2.5, $Clip_{x,\,\epsilon}(x')$ is a clipping value in each iteration limited by ϵ. Multiple iterations are used to create the adversarial images:

$$\begin{aligned} x'_0 &= x\ , \\ x'_{n+1} &= \text{Clip}_{x,\,\epsilon}(x'_n + \alpha\ \text{sign}(\nabla_x J(x'_n, y)))\ . \end{aligned} \tag{2.6}$$

BIM was demonstrated to be more effective than the FGSM attack on ImageNet dataset. In addition, adversarial training can increase the robustness of neural networks for one-step attacks (e.g., FGSM) but might not work well under iterative attacks (e.g., BIM).

2.1.4 Carlini-Wagner (CW)

Carlini-Wagner et al. (CW) introduced an approach that is able to overcome defensive distillation. It presented three powerful attacks against this recent promising defense algorithm in [12]: *L_2 attack, L_0 attack,* and *L_∞ attack* [2]. These three attacks demonstrated 100% success rate in finding adversarial samples on defensively distilled and undistilled networks [2].

Defensive distillation was used to improve the robustness of an arbitrary neural network model against adversarial samples. However, CW proposed L_2, L_0, and L_∞ algorithms that generate adversarial samples causing misclassification with the same label; therefore, they beat defensive distillation with 100% probability. It is processed in four steps:

1. Train a neural network,
2. Calculate the softmax in a smoother way and compute the soft training labels by applying the network to each case in the training data set,
3. Then, train the distilled network with soft training labels,
4. Lastly, the distilled network is tested on the new input to classify.

One of the main contributions that CW presented was that *L_0 attack* was the first approach that results in some specified misclassification on ImageNet. Secondly, after all these three attacks were applied to defensive distillation, it was proved that there was little difference in security between undistilled and distilled networks. *L_0 attack* measures the distance based on the number of pixels that have been changed while the *L_2 attack* is used to measure the distance using the *RMS (Root Mean Square)* between x and x'; where x is an input and x' is a new input found that is similar to x but classified as a targeted t. On the other hand, L_∞ is used to measure the maximum changes to any of the coordinates.

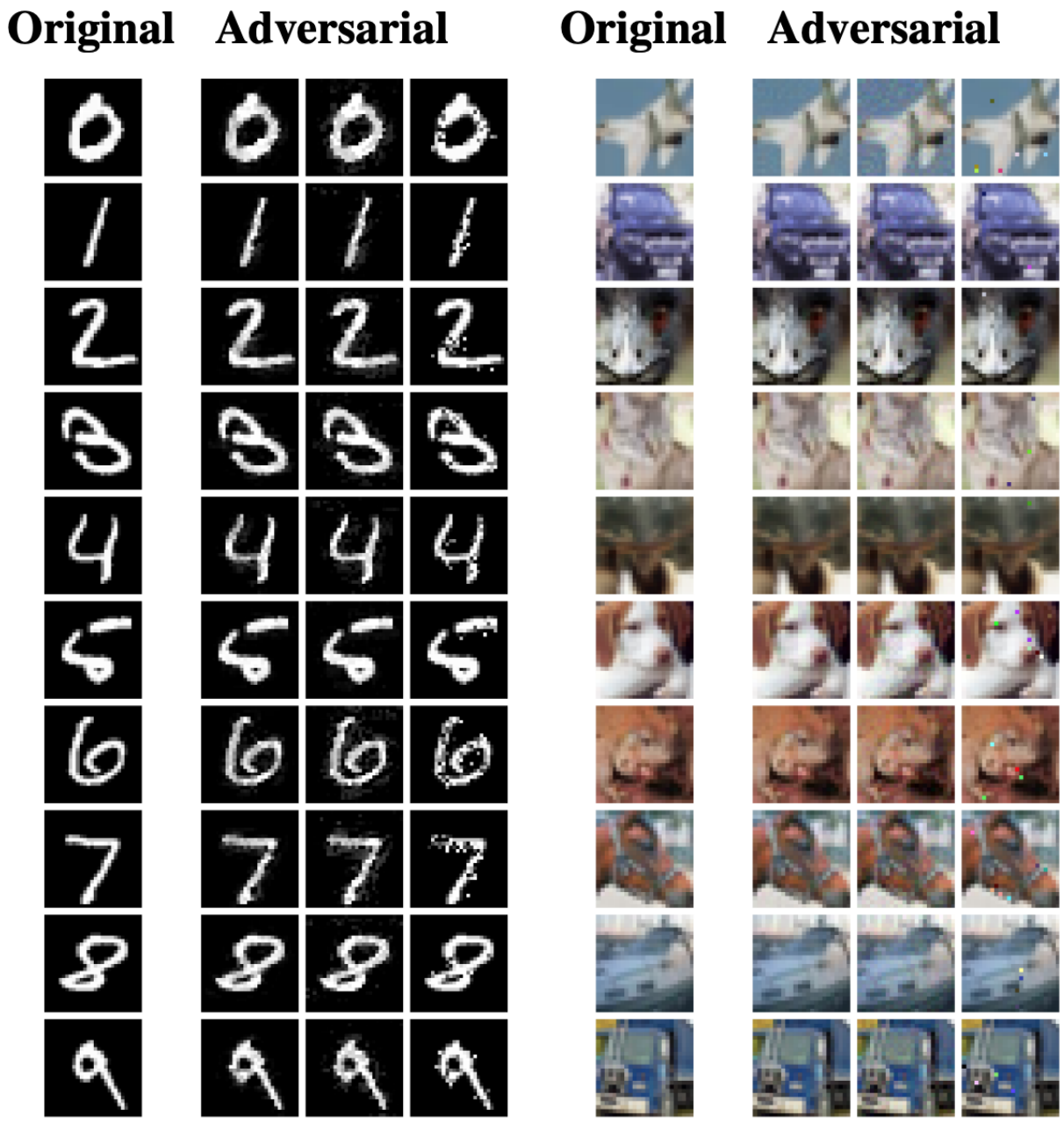


Figure 2.1: A demonstration of the attacks presented in [2] on a defensively distilled network. The leftmost column represents the input image. The next three columns illustrate adversarial images created using L_2, L_∞, and L_0 attacks presented by Carlini-Wagner et al. All the three misclassified examples have the same misclassified label.

It was observed that L_2 *attack* has low distortion while L_0 *attack* is non-differentiable and bad-suited for gradient descent. In addition, L_∞ is not fully differentiable and it is also seen that gradient descent does not give good results.

2.1.5 Jacobian-based Saliency Map Approach (JSMA)

JSMA [3] iteratively perturbs features of input data that has large adversarial saliency scores. This score describes the adversary by taking a sample away from its source class towards a certain target class. All other methods use output variations to find related input perturbations for adversarial samples, whereas JSMA constructs a mapping from input perturbations to output variations. This is called forward derivative, a matrix defined as the *Jacobian* of the function learned by a DNN:

$$\nabla \mathrm{J}(\mathrm{x}) = \partial \mathrm{J}(\mathrm{x}) \,/\, \partial \mathrm{x} = [\partial \mathrm{J_j}(\mathrm{x}) \,/\, \partial \mathrm{x_i}]_{\mathrm{i}\in 1..\mathrm{M},\mathrm{j}\in 1..\mathrm{N}} \,. \tag{2.7}$$

In the given formulation 2.7, x is the input image and $J(x)$ is the classified result by the network defined in [3]; where M is the input dimension or the number of neurons on input layer and N is the output dimension or the number of neurons on output layer. This is the *Jacobian* of the function regarding to what is learned by training the neural network. The smallest possible perturbation is crafted to fool the neural network. Then, two adversarial saliency maps were introduced to manipulate pixels/features in each iteration. 97.10% success rate was achieved to create an adversarial sample that can be misclassified with perturbing only 4.02% of the input features per image.

Generally, a saliency map is known as a visualization tool to explore adversarial examples. This map symbolizes which input features should perturb to affect the changes in the network. As Papernot et al. described in [3], it is a repetitive method for targeted misclassification.

Table 2.3: Results [3] of two average distortion values on three sets (10,000 samples): one considering all images and second considering only successful images.

Original Samples	**Adversary success**	**Average distortion-1**	**Average distortion-2**
Training	97.05%	4.45%	4.03%
Validation	97.19%	4.41%	4.01%
Test	97.05%	4.45%	4.03%

In this classification method, the derivative of DNN is exploited and it tries to find the adversarial perturbation that allows to misclassify the models into the required targets. Let us take the input y and a neural network M; and, the output of the class i is noted by the $M(y)$ for class i. Let us also specify the required class p. $M(y)$ for p would be increased while probabilities $M(y)$ for class i of all other classes that are not p will decrease until the maximum of $M(y)$ is obtained. After defining the adversarial saliency map for the input feature, the pair of features is considered to maximize the saliency map between them and perturbs each feature by some constant offset. This process is repeated until the target misclassification is reached.

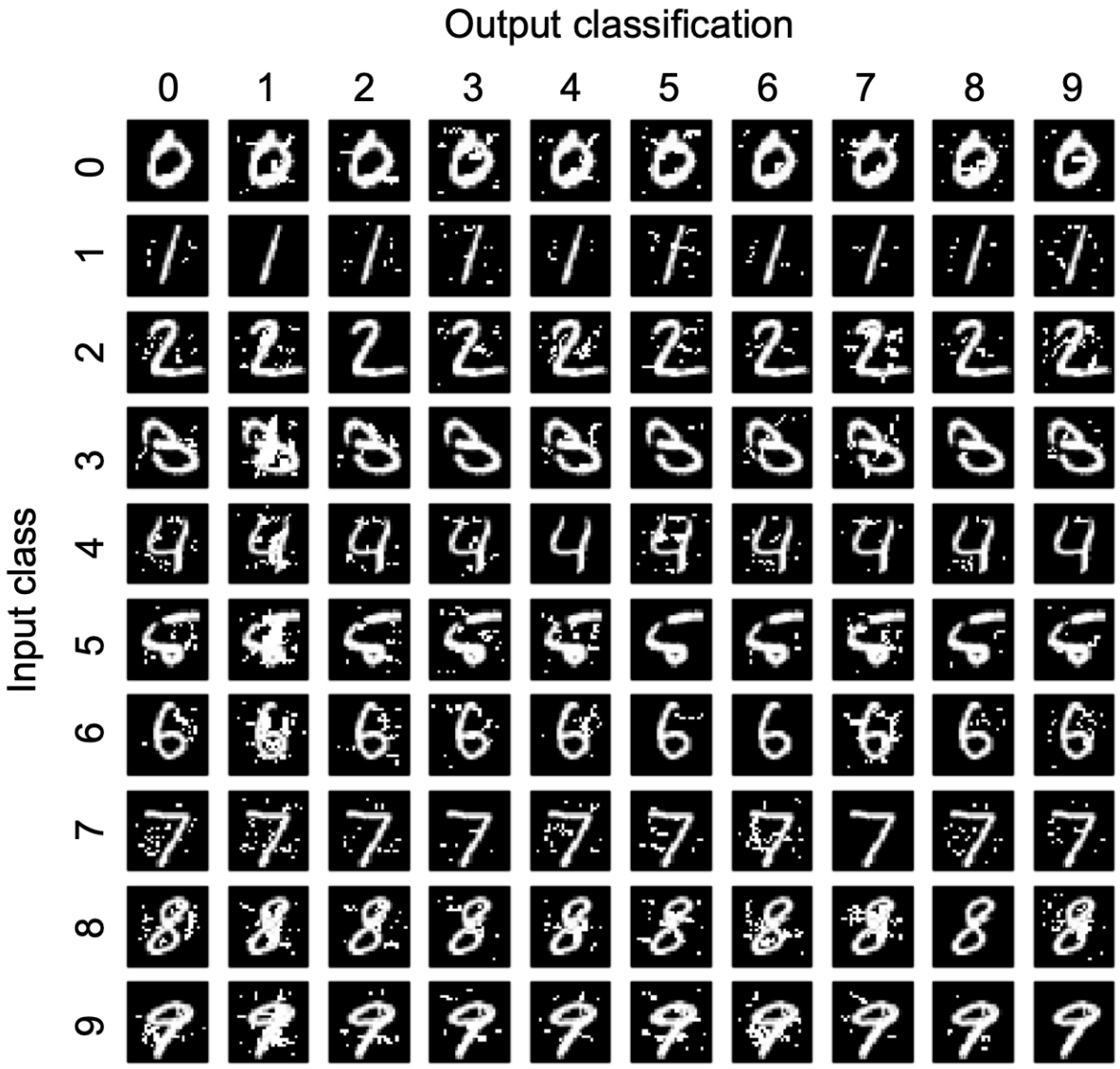


Figure 2.2: Adversarial image generation in [3]. Input images are distorted in order for the DNN to misclassify them (min distortion = 0.26%, max distortion = 13.78%, and average distortion $\epsilon = 4.06\%$).

2.1.6 A Momentum-based Iterative FGSM (MI-FGSM) for Non-targeted and Targeted Attacks

The proposed method in this work [13] was aimed at non-targeted adversarial attack for classification task using ImageNet dataset.

This method was created by adding a momentum term to BIM in order to have more transferable adversarial examples. While one-step methods (i.e. FGSM) can underfit the model, BIM can overfit them; which are mostly untransferable across different models. In order to overcome these problems, a momentum term is integrated into BIM to have more balanced update directions and get rid of poor local optima. In conclusion, having stronger and more transferable attack is the main advantage provided with this approach.

$$\begin{aligned} g_{t+1} &= \mu \cdot g_t + \frac{J_\theta(x'_t, y')}{\|\nabla_x J_\theta(x'_t, y')\|_1}, \\ x'_{t+1} &= \text{Clip}_{[0,1]}\left(x'_t + \alpha \, \text{sign}\left(g_{t+1}\right)\right), \\ f(x) &= \sum_{k=1}^{K} w_k f_k(x). \end{aligned} \tag{2.8}$$

In the given formula 2.8, $g_0 = 0$, $x'_0 = 0$, $\alpha = \epsilon/T$, and T is the number of iterations, g_t is the gradients of *t* number of iterations, μ is the decay factor, x'_t is the perturbed adversarial example. In non-targeted attack, this algorithm was used for ensemble models; $f_k(x)$ is the *k-th* model, w_k is the ensemble weight. The attacked models are listed as follows:

- Normally trained models: Inception-v3 [14], Inception v4 [15], Inception ResNet v2 [15], and ResNet v2-101 [16].
- An adversarially trained model: Inception-$v3_{adv}$ [11].
- Ensemble adversarially trained models: Inc-$v3_{ens3}$, Inc-$v3_{ens4}$, and IncRes-$v2_{ens}$ [17].

The formulation of momentum iterative targeted attack is slightly different from the one used for

non-targeted attacks:

$$g_{t+1} = \mu\, g_t + (\nabla_x J(f(x'_t), y_{target}) \,/\, \mathrm{std}(\nabla_x J(f(x'_t), y_{target}))) \,,$$
$$x'_{t+1} = \mathrm{Clip}_{[0,1]}\,(x'_t - \alpha \mathrm{Clip}_{[-2,2]}\,(\mathrm{round}(g_{t+1}))) \,. \tag{2.9}$$

In the equation 2.9, *std(.)* refers to the standard deviation and *round(.)* is rounding to the nearest integer. Transferability is not considered in targeted attacks. The same models as in the non-targeted track are used for the targeted attacks.

2.1.7 An Iterative FGSM-based Approach for Non-targeted and Targeted Attacks

The proposed solution for non-targeted attack was inspired by the method which uses FGSM with Random Perturbation (RAND+FGSM) [17]. The optimal perturbation is detected under a first-order approximation after a random perturbation is added to an input. The reason why the random perturbation is used is because it skips the non-smooth vicinity of the data point. In [17], RAND+FGSM has a high success rate in white-box setting. However, the transferability of perturbations has low rate. Therefore, a geometric transformation is applied in order to increase transferability. FGSM is a one-shot algorithm which means it only applies a single gradient computation. One-shot attacks provide resiliency for adversarially trained models. However, those models are vulnerable to the attacks which iteratively maximize their loss function. Iterative FGSM (I-FGSM) iteratively applies FGSM k times. It computes multiple gradient updates and lowers error rates more than FGSM. In conclusion, the presented approach in this project outperforms other one-shot methods on all the trained models. The targeted attack has a similar approach with the non-targeted attack as described above. The main differences are as follows:

- The loss function is minimized instead of maximizing.

Table 2.4: The models and their success rate using the Iterative FGSM-based approach for non-targeted attack [5].

Classifier	**Success Rate**
Inception-v3	96.74%
ResNet 50	92.78%
Inception ResNet v2	92.32%
Inception v4	91.69%
EnsAdv Inception ResNet v2	87.36%
Adv Inception-v3	83.73%
Ens-3-Adv Inception-v3	62.76%
Ens-4-Adv Inception-v3	58.11%

- Image augmentation is not used for targeted-attacks due to its lowering the success rate.

2.1.8 High-level Guided Denoiser (HGD) for Defense

The method proposed by Liao et al. [18] was aimed at removing the adversarial noise by training a DNN-based denoiser. 20,000 images from ImageNet composed of 20 images for each class were used in order to train the denoiser. FGSM and Iterative FGSM (I-FGSM) were applied to a variety of models such as Pre-trained Inception-v3, Inception-ResNet-v2, and ResNet50-v2. Denoising U-Net (DUNET) was used in order to remove adversarial perturbation with the following formulation:

$$\begin{aligned} \text{dx'} &= \text{D}_\text{w}(x^{adv}) \,, \\ \text{x'} &= x^{adv} \text{ - dx'} \,. \end{aligned} \tag{2.10}$$

In the given equation 2.10, the clean image is obtained by subtracting the perturbation from the noisy image, where D_w is a denoiser network, *dx'* is estimated perturbation, and x^{adv} is reconstructed noise-free image.

2.2 Digital and Physical Perturbations

Digital perturbations can be classified as individually-tailored or universal. Individually-tailored perturbations generate different perturbations for each of the input images in a dataset [6, 7, 8, 3, 19, 20, 21, 22, 23, 24]. Szegedy et al. [25] was the first to introduce individually-tailored perturbations against deep learning networks in 2014. The adversarial images were generated using the L-BFGS method which uses binary search to obtain an optimal input. The L-BFGS attack was an expensive and time-consuming approach to find an adversarial input. Goodfellow et al. [10] proposed the fast gradient sign method (FGSM). This method performed only a one-step update of the gradient. Rozsa et al. [7] analyzed FGSM and then proposed a new approach, called fast gradient value method. It was obtained by replacing the sign of the gradients with the raw value of the gradients.

Many recent attacks employ individually-tailored perturbations. However, universal perturbations are easier to deploy. They are image-agnostic as they generate a single perturbation for all the images in the dataset [26, 27, 28, 29, 30]. Moosavi-Dezfooli et al. [26] showed that universal perturbations can be generalized across different image classification models. This results in image-agnostic and network-agnostic perturbations. The existence of such general perturbations has been explained by considering the correlation between different image regions of the decision boundary. Mopuri et al. [27] proposed universal perturbations which are quasi-imperceptible to humans but capable of attacking convolutional neural networks (CNNs). This approach is able to attack multiple images from the same target dataset across multiple deep learning networks.

Physical perturbations are generated using real-world objects such as eye glasses or printed stickers that cause an incorrect classification in deep learning models [31, 32, 33]. Kurakin et al. [31] attacked neural networks by applying adversarial images to the physical world by extending FGSM. They made small changes for multiple iterations and for each iteration, the pixel values were clipped to avoid a large change on each pixel. Sharif et al. [32] presented the method of generating

eyeglass frames, which when worn and printed can attack a state-of-the-art deep learning system for face recognition. The perturbations generated are inconspicuous to a human and can be physically acquired via photography in the real world. Lu et al. [33] empirically showed that adversarial perturbations can cause a deep learning network to incorrectly detect a stop sign using physical perturbations when the captured image is taken from a specified range. However, the physical perturbations presented in [31, 32, 33] are not naturally-occurring perturbations, and require the participation of a malicious agent.

In addition, the latest state-of-the-art approach for fog simulation on real scenes was proposed recently by Dai et al. [34]. They used scene semantic annotation as an additional input to their dual-reference cross-bilateral filter on the Cityscapes dataset to obtain Foggy Cityscapes-DBF (Dual-reference cross-Bilateral Filter). They also used a CNN-based approach to estimate fog density.

In this thesis, we propose natural attacks using visibly foggy images to generate input that causes incorrect classification by one of the state-of-the-art image recognition networks, Inception-v3 [14]. Apart from an earlier preliminary work on attacking computer vision algorithms using fog generated via the Perlin noise on two-dimensional images [35], this is the first attempt to attack deep learning classifiers using natural perturbations on stereo images that include depth information and can hence be used to model realistic naturally-occurring fog. As shown in Figure 1.1 and Figure 2.3, our approach of adding fog to images can cause deep neural networks to incorrectly classify input images.

(a) Image with fog incorrectly classified as aircraft by the Inception-v3 model with P_{tf}=0.15, P_{al}=0.6 and PSNR=9.44.

(b) Image with fog incorrectly classified as scooter by the Inception-v3 model with P_{tf}=0.07, P_{al}=0.6 and PSNR=10.77.

Figure 2.3: Images from Cityscapes dataset are incorrectly classified upon the synthetic addition of light fog.

(a) Image with fog incorrectly classified as submarine by the Inception-v3 model with P_{tf}=0.12, P_{al}=0.8 and PSNR=6.74.

(b) Image with fog incorrectly classified as submarine by the Inception-v3 model with P_{tf}=0.12, P_{al}=1 and PSNR=4.36.

Figure 2.4: Images from Cityscapes dataset are incorrectly classified upon the synthetic addition of thick fog.

CHAPTER 3: METHODOLOGY

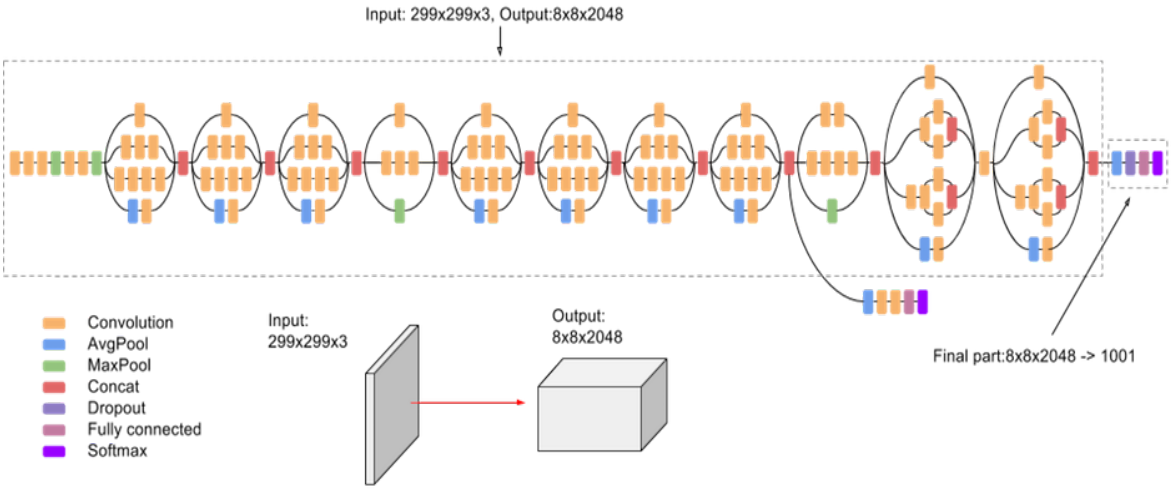


Figure 3.1: Schematic illustration of the Inception-v3 network (Source: `https://git.io/Jez0c`).

3.1 Inception-v3 Model

In the last few years, convolutional neural networks (CNNs) have demonstrated great success in real world applications. Recent studies are aimed to utilize added computation in the most efficient manner using aggressive regularization and factorized convolutions to enhance networks in different ways.

The AlexNet network by Krizhevsky et al. [36] won the ImageNet competition in 2012 and has given successful results in a wide range of computer vision tasks such as video classification, segmentation, and object tracking. It was composed of eight layers; convolutional layers take place in the first five with some max-pooling layers, and the rest of the layers is fully-connected. It utilized the non-saturating Rectified Linear Unit (ReLU) activation function that outperformed

over *tanh* and *sigmoid*. With the inspiration of the AlexNet contributions, the research studies have emerged by aiming to discover new convolutional neural networks with higher quality of performance. Within a couple of years with the use of wider and deeper networks, the quality of network architectures significantly increased.

With the ILSVRC classification challenge in 2014, deep learning models gained higher performance due to the discovery of VGGNet [37] and GoogLeNet [38] networks. One of the advantages of VGGNet is its architectural simplicity, however, it requires a lot of computation. As for GoogLeNet, it was crafted to excel under tough constraints on computation budget as well as memory, however, it owns adaptation difficulties to new use-cases due to its unclear contribution elements.

Inception-v3 [14] is one of the most widely-used image recognition architectures that has been demonstrated to achieve greater than 78.0% accuracy on the ImageNet dataset. As in Figure 3.1, the model is composed of convolutions, average pooling, max-pooling, concatenations, dropouts, and fully-connected layers. The softmax activation is used for the loss function. In our experiments, we selected to use the Inception-v3 network due to its performance with high accuracy, its feasibility in big data scenarios, and its low computational cost.

3.2 ImageNet and Cityscapes Datasets

The model we used, Inception-v3, is pre-trained on the ImageNet dataset [39] that contains 1,331,167 images. They are divided into two sets of images: One is a training set with 1,281,167 images, the second is an evaluation set with 50,000 images. It has 1,000 different object classes such as cats, dogs, cars, bike, traffic light, and many more.

The Cityscapes dataset [40] contains 25,000 stereo images with 30 varied visual theme categories

such as road, sidewalk, person, rider, car, bus, building, bridge, traffic sign, and traffic light. Each stereo image is a pair of images captured from two different cameras. These pairs of images are denoted as left and right images.

3.3 Our Approach

We use the images obtained from the Cityscapes [40] dataset and added fog to attack the Inception-v3 deep learning model. We use these left and right pairs of images from Cityscapes to create a depth mapping of objects in the image. Then, we use the depth information of the objects in the images to synthetically add fog to these images; the presence of depth information allows the synthetically-generated fog to resemble naturally-occurring fog in the image [1].

The typical aim of an adversarial attack test is to add some natural perturbation (e.g. fog, sunlight, visual environmental changes and aberrations, etc.) over an input image in order for the deep learning model to misclassify the image. However, it is still correctly recognized and identified by a regular human visual-eye observer. To corroborate our claims, in this thesis we proceed to generate a conventional, outside fog environment as a naturally-occurring, subtle climate perturbation, in order to provide this foggy image as a qualified difficult adversarial attack against the most advanced, novel deep learning models to date including Inception-v3.

First, we run the Inception-v3 model for an autonomous driving potential application using the weather-clear images in our dataset. Here, we seek to obtain accurate image recognition decision results. Then, we apply generated visual fog conditions onto said baseline images from this dataset, using specific stereo-pair images and disparity mapping techniques. Once this counterintuitive, adversarial image is produced, Peak Signal-to-Noise Ratio (PSNR) value disparities between our initial weather-clear images and their corresponding foggy images are observed.

(a) Original left image classified as traffic light by the Inception-v3 model.

(b) Original right image that forms a stereo pair along with left image.

Figure 3.2: Stereo images from the Cityscapes dataset.

(a) Disparity image generated by the SGBM method using the left and right images. It shows the distance of objects from the observer. Objects closer to the observer appear to be brighter and objects further away from the observer appear to be darker.

(b) Foggy image generated by using the left and disparity images. Image with fog incorrectly classified as scooter by the Inception-v3 model with P_{tf}=0.07 and P_{al}=0.6.

Figure 3.3: Our disparity image generated from the Cityscapes stereo images using semi-global block matching algorithm (SGBM) and its corresponding foggy image.

3.3.1 Semi-Global Block Matching for Depth Information

Developing dense and accurate stereo matching algorithm is challenging caused by blurry object boundaries, occlusions, and fine structures. In order to gain better insight into the concept of stereo matching, we can list four steps that differentiate each up-to-date stereo method: matching cost computation, cost aggregation, disparity computation and optimization, disparity refinement. In literature, the matching cost has been extensively computed by the difference of intensities. Mutual information for faster computation and also image gradients are used to compute matching cost. Another bullet is cost aggregation which links the matching costs inside a certain window of neighborhood based on patches of constant intensity. As for disparity computation, it is selected with the lowest matching error by pixel-wise matching costs for local algorithms or chosen by block-wise smooth disparity selection for global algorithms. Dynamic programming, graph cuts, belief propagation, and layered methods are some of the practical frameworks that have been utilized to calculate the minimum of the global cost. Lastly, disparity refinement is usually processed by peaks removal, consistency control, or gaps interpolation.

We used a variant of stereo processing by semi-global matching (*Stereo SGM*) [41] implemented in the popular OpenCV toolkit. Instead of matching individual pixels, our algorithm matches blocks (*stereo processing by semi-global block matching*) so as to calculate the depth of every pixel in the image. This depth information generated by *stereo SGBM* is called the *disparity* of the image and this block-wise matching is upheld by a global cost function which is actually defined as a smoothness constraint.

Additional depth mapping information is available in the Cityscapes dataset [40] but was not precise enough to generate smooth natural fog. In order to obtain smooth disparity image, some pre- and post- processing filters are implemented such as weighted least squares (WLS) filter, fast bilateral solver filter (FBS), uniqueness check, and speckle filtering. Then, we use the depth value

of each pixel to mimic realistic fog. A higher depth value indicates that the object is further away from the observer and is less visible. An object that has a lower depth value is closer to the observer and is not affected adversely by fog.

3.3.1.1 Matching Cost

We assume that input images have epipolar geometry and epipolar lines turn into hyperbolas by a linear movement. However, non-linear movements turn epipolar lines into general curves and unrectified images.

The matching cost is computed via the intensity I_{bp} of a pixel p of a base image and its corresponding pixel in the match image $q = e_{bm}(p,d)$ with the line parameter d. The epipolar line in the related match image is defined as the function $e_{bm}(p,d)$.

As for block matching, the default window size of 15 is suggested to use on full-sized views.

Window size of 7 is selected on downscaled views. The views are downscaled in order to obtain faster matching process as both left and right disparity images are in need to calculate WLS confidence map. WLS is in form of fast global smoother and uses left-right-consistency-based confidence to improve the results in noisy regions. WLS filtering with confidence computation provides considerably better quality results than filtering without confidence.

We used the measure of pixel dissimilarity by Birchfield and Tomasi et al. [42, 43]. So as to compute the cost $C_{BT}(p,d)$, we applied the absolute minimum difference of intensities of the pixels p and q. We also applied the default window size of 3 for stereo SGBM and force stereo matching on full-sized views to improve the quality of images.

3.3.1.2 Cost Aggregation

Due to the potential inaccurate matches that might have a lower cost caused by perturbations, an additional constraint is utilized that provides smoothness. It aims to penalize neighboring disparity changes. The energy *E(D)* of the disparity image *D* is defined for the cost calculation and the smoothness constraints:

$$\mathrm{E(D)} = \sum_p C(p, D_\mathrm{p}) + \sum_{q \in N_\mathrm{p}} Z_1 T[|D_\mathrm{p} - D_\mathrm{q}| = 1] + \sum_{q \in N_\mathrm{p}} Z_2 T[|D_\mathrm{p} - D_\mathrm{q}| > 1] \,. \tag{3.1}$$

The disparities are used to add all matching costs in the first term. All pixels *q* in neighborhood $\mathrm{N_p}$ are penalized by $\mathrm{Z_1}$ as disparity changes in the second term. Larger penalty is applied for larger disparity changes in the last term. Then, aggregation (smoothness) matching costs in disparity space is calculated by adding up the costs of all minimum cost paths that finish at desired point.

3.3.1.3 Disparity Computation and Optimization

The computation of both $\mathrm{D_b}$ and $\mathrm{D_m}$ enables us to detect inaccurate matches by applying a consistency check, which ensures the uniqueness constraint. A comparison between every disparity $\mathrm{D_b}$ and its related $\mathrm{D_m}$ is performed and the disparity is assigned as invalid $\mathrm{D_{inv}}$ if they are both different:

$$D_\mathrm{p} = \begin{cases} D_\mathrm{bp}, & if |D_\mathrm{bp} - D_\mathrm{mq}| \leq 1, \\ D_\mathrm{inv}, & \text{otherwise.} \end{cases} \tag{3.2}$$

$$\mathrm{q} = \mathrm{e_{bm}(p, D_{bp})}. \tag{3.3}$$

3.3.1.4 Disparity Refinement

In order to obtain more refined disparity images, firstly a simple disparity segmentation and peak filtering are performed for peaks removal. Then, the disparity is selected based on each constant intensity segment in order to overcome the discontinuities within untextured areas. Lastly, different interpolations are performed separately for both the mismatches and occlusions. Mismatches might occur as a result of incorrect smoothing of discontinuities. Therefore, an extrapolation is the essential step for the background. However, holes that was caused by incorrect matches are smoothly interpolated from all neighboring points.

$$D'_{\mathrm{p}} = \begin{cases} secl_{\mathrm{i}} v_{\mathrm{pi}}, & \text{if p is occluded,} \\ med_{\mathrm{i}} v_{\mathrm{pi}}, & \text{if p is mismatched,} \\ D_{\mathrm{p}}, & \text{otherwise.} \end{cases} \tag{3.4}$$

It is ensured that the second lowest value ($secl_i v_{pi}$) is selected for the interpolation of occlusions from the lower background in the first case. All information is utilized in the second case regardless of a selection of background or foreground. Lastly, the final disparity image D'$_{\mathrm{p}}$ is generated by applying median ($med_i v_{pi}$) for the irregularity removal and additional smoothness.

3.3.2 Fog Generation

Besides the depth of a pixel, our synthetically-generated fog requires two additional parameters: the fog thickness (P_{tf}) and the ambient atmospheric light (P_{al}). An example of right, left, disparity and final foggy images is shown in Figure 3.2. Disparity images are stored in such a way that objects closer to the observer are brighter and objects further away from the observer are darker. Examples of fog for various values of P_{tf} and P_{al} values are shown in Figure 2.3.

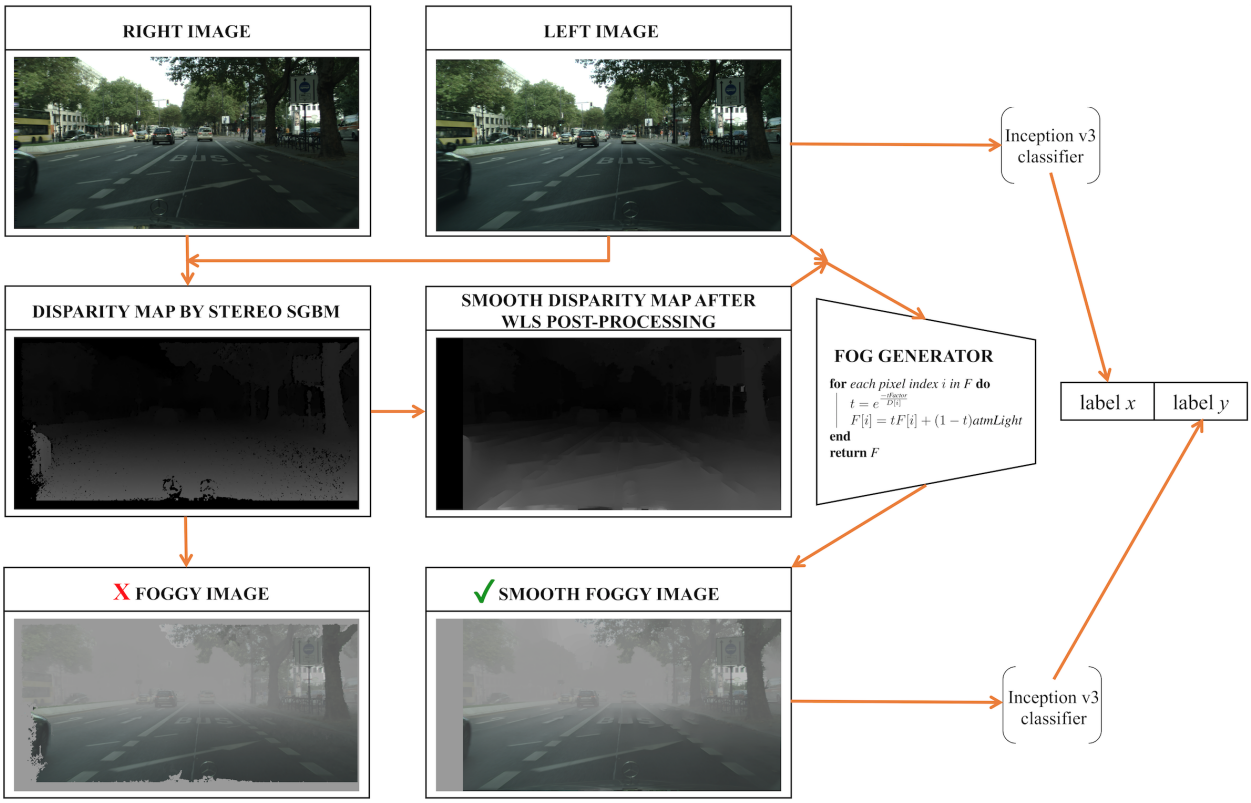


Figure 3.4: Flowchart of the adversarial attack using our fog generator.

The thickness parameter P_{tf} determines the intensity of fog; a thicker fog can occlude objects that are closer to the observer. The atmospheric light P_{al} parameter determines the color and intensity of ambient light; we used white light of varying intensity for our fog. A lower value of P_{al} leads to fog that is darker in color and a higher value of P_{al} leads to a fog that is brighter.

Steps to generate foggy images are presented in Algorithm 1. This algorithm takes a stereo image pair as input: left image (L) and right image (R), thickness factor (P_{tf}) and atmospheric light (P_{al}).

First, we run Inception-v3 classifier [14] on the weather-clear images from the Cityscapes dataset [40] and report the original label as (l). Second, the targeted foggy image (F) is initialized to left image obtained from Cityscapes.

Data: Left Image *L*, Right Image *R*, Thickness factor P_{tf}, Atmospheric Light P_{al}.
Result: Image *F* with synthetically added fog.
begin
  F = *L* `// Initialize to left image.`
  DN = *stereoSGBM (R, L)* `// Calculate noisy disparity image using stereo semi-global block matching algorithm.`
  D = *filter (DN, R, L)* `// Filter noisy disparity image to generate smooth disparity image.`
  for *each pixel index* i *in F* **do**
    $t = e^{\frac{-P_{tf}}{D[i]}}$ `// Compute transmission intensity.`
    $F[i] = tF[i] + (1-t)P_{al}$
  end
  return *F* `// Return foggy image.`
end

Algorithm 1: Algorithm to add fog to a stereo image pair.

Using the left and right image, we find (*D*) as the smooth disparity image generated by the stereo SGBM and some pre- and post- processing filters such as WLS filtering, uniqueness check, and speckle filtering as mentioned in 3.3.1. Third, the transmission intensity is computed for each pixel in the targeted foggy image (*F*) as shown in the following equation:

$$t = e^{\frac{-P_{tf}}{D[i]}} . \tag{3.5}$$

In the equation 3.5, (P_{tf}) refers to the fog thickness which is a combination of fog density and other parameters. Last, the foggy image (*F*) keeps being generated as shown in the following equation by iteratively updating P_{tf} and atmospheric light (P_{al}) until our algorithm finds the lightest foggy image that causes the Inception-v3 model [14] to misclassify the object in the image as a different label from the original one (*l*):

$$F[i] = tF[i] + (1-t)P_{al} . \tag{3.6}$$

CHAPTER 4: RESULTS FROM IMAGES CREATED BY OUR FOG GENERATOR

We test the robustness of the Inception-v3 deep learning classifier [14] on the synthetic images with fog generated by our algorithm 1. In order to create fog on the images from the Cityscapes self-driving car dataset, we use the stereo image pairs. These images are called the left and right images which are used to create disparity map for each pair. Then, we run the classifier on the original left image and note the classification label such as car, bike, or traffic light. Next, we create fog on the weather-clear image using its corresponding disparity map based on our algorithm 1. We then aim to generate the perturbed image with the lightest fog that causes a deep learning model to misclassify the object in the image and run the classifier on the foggy image and note the new classification label. Once the original classification label is different from the classification label generated from the foggy image, we have exposed a potential safety error in the deep learning classifier:

$$\mathit{PSNR} = 20 \log_{10} \left(\frac{D}{\mathit{RMSE}} \right). \tag{4.1}$$

An ideal test of the robustness of the deep learning system will have foggy images that look similar to the original image. We measure the similarity between the original and the foggy image using the peak signal to noise ratio (*PSNR*) value. *PSNR* value can be calculated using the equation 4.1, where D is the maximum possible pixel value of the image and *RMSE* is the root mean squared error calculated between the original and the foggy image.

High *PSNR* values indicate a greater similarity between the original and foggy image. In Figure 2.3, we show images with varying *PSNR* values. We observe that images with more visible fog have a lower *PSNR* value indicating lower similarity between the original and foggy images.

In general, fog generated with higher P_{tf} and P_{al} values have lower *PSNR* values. We generate multiple foggy images by varying the values for P_{tf} and P_{al}. Then, we run the classification model on these images and select the image with the highest *PSNR* value that is able to fool the deep learning system. In Figure 2.3, Image *b* has the highest *PSNR* among all generated foggy images that is incorrectly classified by the Inception-v3 deep learning model.

In our experiments, we aim to attack a deep learning model, Inception-v3, by adding fog using our fog generator (*Algorithm 1*) on the Cityscapes dataset. Tables 4.2, 4.3, and 4.4 demonstrate the PSNR value between an original left image and its corresponding foggy image that we find as adversarial. First, we run the Inception-v3 model on the Cityscapes images and we classify them based on their labels (e.g., *car, traffic light, bike*). Second, the algorithm runs our fog generator to create the foggy images. The first step of the fog generation on weather-clear images is to create the smooth disparity map using the stereo-SGBM method. This method requires to use the left and right images; which are taken from inside a car with two cameras at different angles targeting at the same object at the same time. Then, our synthetic foggy image is generated using the formula in algorithm 1. Then, we report the largest *PSNR* value between the original left image and foggy image that has a different label from the original one classified by Inception-v3. The higher PSNR value, the more similar weather-clear and foggy images are; therefore, we find the lightest fog effect on a weather-clear image that leads Inception-v3 model to misclassify the object in the foggy image. In addition to the largest PSNR value, the model also returns the label of the perturbed image with the corresponding P_{tf} and P_{al} values.

From our experiments, one may conclude that:

- The bounded *PSNR* value for the car images is found to be from 8.59 to 21.35. The adversarial foggy images of cars are observed to be classified as different labels (e.g., *park bench, parking meter, fountain, stage, bubble, washbasin*).

- The bounded *PSNR* value is also observed to be from 9.42 to 19.69 for the traffic light images. The generated adversarial foggy images of traffic light have different labels such as *spotlight*, *wing*, *fountain*, *umbrella*, *locomotive* and *parking meter*.

- The bounded *PSNR* value varies from 8.60 to 20.76 for the bike images. The adversarial images of bikes are classified as *lakeshore, scuba diver, aircraft carrier, bubble, fountain, maze, spotlight, washbasin, wing, and submarine*.

- Overall, we see that the decision boundary between the weather-clear images and their corresponding foggy adversarial images to vary from 8.59 to 21.35 *PSNR*.

- The higher PSNR value there is between a weather-clear image and foggy image, the lighter fog effect there is on the original image.

- It may also be observed that the minimum P_{tf} and P_{al} values that result in an adversarial foggy image are 0.07 and 0.60, respectively.

- It may also be seen that the maximum *PSNR* values that are found are considerably close to each other for the same labels of adversarial images. For example, the maximum *PSNR* values for almost all the adversarial images that are classified as *parking meter* vary from 9.24 to 12.26.

- These perturbed classes crucially affect the decision mechanism of any system that works with deep learning classifiers.

Apart from Inception-v3, we have tested the classification results using 6 other image recognition models on our fog images. They are Inception-ResNet-v2 [44], MobileNet [45], ResNet-50 [46], Xception [47], VGG-16 and VGG-19 [48]. Tables 4.2, 4.3, and 4.4 demonstrate the classification results using these deep learning models on our fog images.

Table 4.1: The table demonstrates the performance results of the models we use and some architectural differences among each other such as number of parameters and depth. The top-1 and top-5 accuracy results are based on the ImageNet validation dataset. We obtain the similar outcomes such that the Xception model is observed to be the least vulnerable among the others on our foggy images. Depth refers to the topological depth of the network that includes some layers such as activation layers and batch normalization layers.

Model	**Top-1 Accuracy**	**Top-5 Accuracy**	**Parameters**	**Depth**
Xception	0.790	0.945	22,910,480	126
VGG-16	0.713	0.901	138,357,544	23
VGG-19	0.713	0.900	143,667,240	26
ResNet-50	0.749	0.921	25,636,712	-
Inception-v3	0.779	0.937	23,851,784	159
Inception-ResNet-v2	0.803	0.953	55,873,736	572
MobileNet	0.704	0.895	4,253,864	88

These synthetic foggy images are initially found as adversary by the Inception-v3 model and they are tested by other abovementioned deep learning models afterwards. From our experiments in the Tables 4.2, 4.3, and 4.4, we obtain the similar results as in the Table 4.1. As we report their classified labels, we notice that some of the fog images that contain car are still classified as *car* even though they are classified as different classes by Inception-v3. For instance in the Table 4.2, the perturbed images of *berlin012, 015, 027, 151, 154, 155, 183, 437* are classified as other objects by Inception-v3, whereas they are still correctly classified as *car* by Xception. Another example in the Table 4.4, the perturbed images of *berlin079, 202, 206, 316, 326, 336, 384* are classified as other objects by Inception-v3; however, they are still correctly classified as *bike* by Xception. We also observe that there are some cases where the Inception-ResNet-v2 model is able to correctly classify the object on the foggy image although the Inception-v3 model fails such as the perturbed images of *berlin094, 156* in the Table 4.3.

In conclusion, one may observe that the Xception model is more robust on foggy images compared to the other six models we used for our experiments. From the Table 4.1, it may be observed that Xception has similar model size with Inception-v3. However, the reason why the Xception model

has higher accuracy than the Inception model is due to its Depthwise Separable Convolution architecture and its more efficient use of model parameters. Depthwise Separable Convolution is a spatial convolution performed separately for every channel followed by a pointwise convolution, which is a 1x1 convolution across channels. We may consider this as performing correlations across 2D and then 1D space. The 2D + 1D mapping is easily learned more than a full 3D mapping and it outperforms Inception-v3 on the ImageNet and our foggy image datasets. MobileNet has similar architecture with Xception and is optimized for mobile applications. The difference is that MobileNet has two parameters that causes to have less number of operations. One of them is the width multiplier which operates as the thinner of number of channels. The other one is the resolution multiplier which is able to resize the input image. As for VGG networks, they have the basic architecture of CNN; a number of convolutional layers, max-pooling, activation layers, and fully-connected layers. The 16 and 19 specify the number of layers and these were considered very deep in 2014. However, the ResNet50 model was successfully trained with 50 weight layers in 2015. This "Residual Network" proposes learning the difference between the identity x and its underlying mapping H(x), which is called "deep residual learning". It is observed that residual mapping is easier to optimize than the original unreferenced mapping. We obtained different classification results on our synthetic foggy images based on all these architectural differences among the models we used for our experiments in the Tables 4.2, 4.3, 4.4.

Table 4.2: Images from Cityscapes classified as car by Inception-v3 and their corresponding foggy image we found as adversarial. We also report the classification results of these foggy images using Inception-ResNet-v2, MobileNet, ResNet-50, VGG16, VGG19, and Xception. We add fog on the original left image with the parameters P_{tf} (thickness factor) and P_{al} (atmospheric light) to obtain an incorrect class using the model. We also report the maximum PSNR value defined in the equation 4.1 to keep track of image similarity between the weather-clear image and the foggy image with the lightest fog that causes Inception-v3 to misclassify.

Image	**P_{tf}&P_{al}**	**PSNR**	**InceptionV3**	**ResNetV2**	**MobileNet**	**ResNet50**	**VGG16**	**VGG19**	**Xception**
berlin000	0.12&1.00	10.30	park bench	traffic light	net	tripod	net	net	traffic l.
berlin009	0.10&1.00	12.26	parking meter	parking meter	**car**	parking meter	**car**	**car**	parking m.
berlin012	0.10&1.00	9.63	fountain	**car**	breakwater	net	fireboat	**car**	**car**
berlin015	0.10&0.80	15.44	bubble	bubble	net	fountain	monitor	bubble	**car**
berlin027	0.07&0.80	17.22	fountain	bubble	**car**	fountain	monitor	monitor	**car**
berlin070	0.07&0.80	17.93	stage	aircraft carrier	submarine	submarine	**car**	**car**	aircraft c.
berlin072	0.10&0.80	17.20	bubble	**car**	umbrella	tripod	wing	wing	bubble
berlin090	0.10&1.00	10.19	washbasin	dining table	net	piano	fireboat	net	net
berlin144	0.10&1.00	11.44	parking meter	parking meter	volcano	television	snowplow	fireboat	parking m.
berlin151	0.10&1.00	11.92	parking meter	parking meter	**car**	snowplow	snowplow	fountain	**car**
berlin154	0.10&0.60	21.35	spotlight	locomotive	net	bridge	wing	wing	**car**
berlin155	0.12&0.60	20.97	bullet	**car**	net	television	wing	wing	**car**
berlin160	0.12&0.60	19.81	spotlight	bathtub	tent	spider web	net	net	mirror
berlin164	0.15&1.00	8.59	fountain	washbasin	volcano	fountain	net	net	bubble
berlin172	0.10&1.00	9.92	mailbox	bubble	net	television	net	curtain	bubble
berlin180	0.15&1.00	8.73	ship	snowplow	net	bridge	net	curtain	bridge
berlin182	0.12&0.80	14.51	stage	bannister	stupa	television	net	net	cote
berlin183	0.15&1.00	9.12	locomotive	snowplow	projector	fireboat	monitor	fireboat	**car**
berlin352	0.10&1.00	12.16	spotlight	tub	net	net	net	net	volcano
berlin437	0.15&1.00	9.31	parking meter	parking meter	net	fountain	net	net	**car**

(a) Weather-clear image.

(b) Smooth disparity image generated by the SGBM method.

(c) Foggy image obtained from using the weather-clear and disparity images.

Figure 4.1: An image of *car* with fog incorrectly classified as *fountain* by the Inception-v3 model with P_{tf}=0.07, P_{al}=0.8 and PSNR=17.22.

(a) Weather-clear image.

(b) Smooth disparity image generated by the SGBM method.

(c) Foggy image obtained from using the weather-clear and disparity images.

Figure 4.2: An image of *car* with fog incorrectly classified as *stage* by the Inception-v3 model with P_{tf}=0.07, P_{al}=0.8 and PSNR=17.93.

(a) Weather-clear image.

(b) Smooth disparity image generated by the SGBM method.

(c) Foggy image obtained from using the weather-clear and disparity images.

Figure 4.3: An image of *car* with fog incorrectly classified as *spotlight* by the Inception-v3 model with P_{tf}=0.1, P_{al}=0.6 and PSNR=21.35.

(a) Weather-clear image.

(b) Smooth disparity image generated by the SGBM method.

(c) Foggy image obtained from using the weather-clear and disparity images.

Figure 4.4: An image of *car* with fog incorrectly classified as *locomotive* by the Inception-v3 model with P_{tf}=0.15, P_{al}=1 and PSNR=9.12.

Table 4.3: Images from Cityscapes classified as traffic light by Inception-v3 and their corresponding foggy image we found as adversary. We also report the classification results of these foggy images using Inception-ResNet-v2, MobileNet, ResNet-50, VGG16, VGG19, and Xception. We add fog on the original left image with the parameters P_{tf} (thickness factor) and P_{al} (atmospheric light) to obtain incorrect class using the model. We also report the maximum PSNR value defined in the equation 4.1 to keep track of image similarity between the weather-clear image and the foggy image with the lightest fog that causes Inception-v3 to misclassify.

Image	**P_{tf}&P_{al}**	**PSNR**	**InceptionV3**	**ResNetV2**	**MobileNet**	**ResNet50**	**VGG16**	**VGG19**	**Xception**
berlin014	0.10&0.80	15.41	spotlight	fire engine	net	fire engine	net	net	car
berlin016	0.10&0.80	15.29	fountain	locomotive	fireboat	television	snowplow	bubble	car
berlin032	0.10&1.00	19.69	parking meter	parking m.	car	car	car	car	car
berlin039	0.10&1.00	10.30	wing	laptop	volcano	tripod	monitor	net	bannister
berlin048	0.15&1.00	12.46	minivan	minivan	amphibian	fireboat	snowplow	fireboat	car
berlin063	0.15&1.00	17.93	spotlight	shopping cart	umbrella	missile	net	wing	spotlight
berlin094	0.10&1.00	12.58	groom	**traffic light**	bubble	guillotine	snowplow	missile	dumbbell
berlin123	0.10&1.00	9.66	wing	umbrella	dome	wing	net	umbrella	screen
berlin147	0.10&1.00	11.10	spotlight	door	umbrella	fountain	net	umbrella	washbasin
berlin153	0.10&1.00	9.64	vault	tractor	maze	abaya	net	umbrella	tractor
berlin156	0.10&1.00	11.92	umbrella	**traffic light**	wing	fountain	wing	wing	turnstile
berlin159	0.10&1.00	11.22	stage	bike	net	book jacket	monitor	door	car
berlin161	0.12&0.80	14.31	spotlight	washbasin	wing	wing	net	net	parking m.
berlin162	0.12&0.80	14.13	aircraft carrier	television	tub	television	net	umbrella	aircraft c.
berlin226	0.10&1.00	11.80	parking m.	washbasin	washbasin	curtain	monitor	safe	washbasin
berlin268	0.10&1.00	11.79	locomotive	locomotive	wing	television	fountain	fountain	locomotive
berlin298	0.10&1.00	10.54	wing	locomotive	dime	fountain	net	net	car mirror
berlin327	0.10&0.80	16.06	stage	car	fireboat	fireboat	aircraft c.	fireboat	fireboat
berlin358	0.12&1.00	10.34	fountain	bubble	vault	net	net	net	bike
berlin430	0.12&1.00	9.42	parking m.	car mirror	car	fireboat	car	car	car
berlin483	0.10&1.00	11.01	car	car	car	fire engine	car	projector	snowplow
berlin484	0.10&1.00	11.51	locomotive	locomotive	car	fountain	fountain	fountain	locomotive

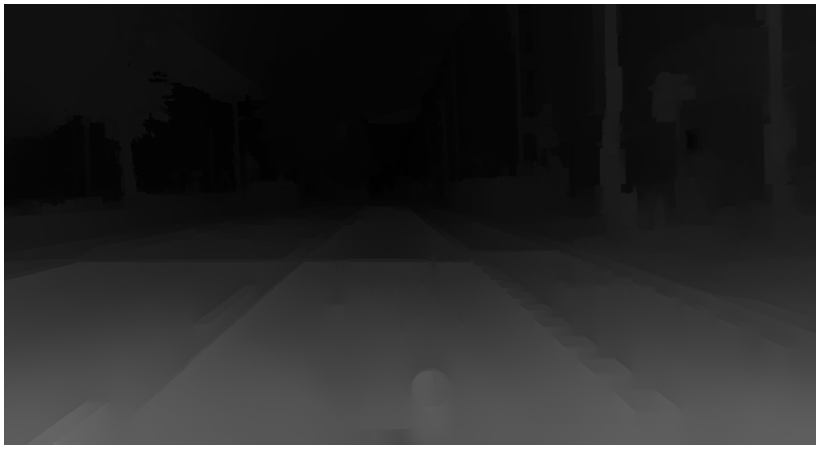
(a) Weather-clear image.

(b) Smooth disparity image generated by the SGBM method.

(c) Foggy image obtained from using the weather-clear and disparity images.

Figure 4.5: An image of *traffic light* with fog incorrectly classified as *spotlight* by the Inception-v3 model with P_{tf}=0.15, P_{al}=1 and PSNR=17.93.

(a) Weather-clear image.



(b) Smooth disparity image generated by the SGBM method.

(c) Foggy image obtained from using the weather-clear and disparity images.

Figure 4.6: An image of *traffic light* with fog incorrectly classified as *spotlight* by the Inception-v3 model with P_{tf}=0.1, P_{al}=1 and PSNR=11.10.

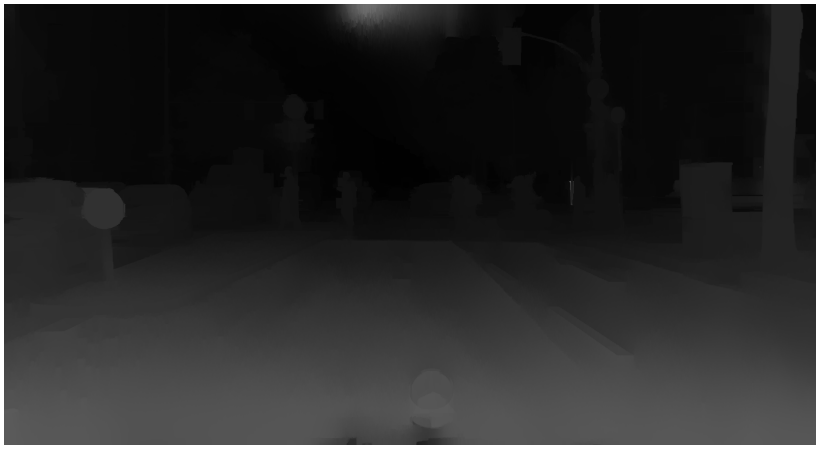
(a) Weather-clear image.

(b) Smooth disparity image generated by the SGBM method.

(c) Foggy image obtained from using the weather-clear and disparity images.

Figure 4.7: An image of *traffic light* with fog incorrectly classified as *vault* by the Inception-v3 model with P_{tf}=0.1, P_{al}=1 and PSNR=9.64.

(a) Weather-clear image.



(b) Smooth disparity image generated by the SGBM method.

(c) Foggy image obtained from using the weather-clear and disparity images.

Figure 4.8: An image of *traffic light* with fog incorrectly classified as *parking meter* by the Inception-v3 model with P_{tf}=0.1, P_{al}=1 and PSNR=11.80.

Table 4.4: Images from Cityscapes classified as bike by Inception-v3 and their corresponding foggy image we found as adversary. We also report the classification results of these foggy images using Inception-ResNet-v2, MobileNet, ResNet-50, VGG16, VGG19, and Xception. We add fog on the original left image with the parameters P_{tf} (thickness factor) and P_{al} (atmospheric light) to obtain incorrect class using the model. We also report the maximum PSNR value defined in the equation 4.1 to keep track of image similarity between the weather-clear image and the foggy image with the lightest fog that causes Inception-v3 to misclassify.

Image	**P_{tf}&P_{al}**	**PSNR**	**InceptionV3**	**ResNetV2**	**MobileNet**	**ResNet50**	**VGG16**	**VGG19**	**Xception**
berlin079	0.15&1.00	8.60	lakeshore	**bike**	wing	fountain	fireboat	fireboat	**bike**
berlin100	0.15&1.00	9.59	scuba diver	car	amphibian	car	amphibian	amphibian	car
berlin176	0.10&0.80	15.53	submarine	bannister	umbrella	television	snowplow	fireboat	car
berlin202	0.10&1.00	11.12	chair	**bike**	missile	missile	net	car	**bike**
berlin206	0.10&0.60	18.04	aircraft carrier	breakwater	aircraft c.	wreck	airship	fireboat	**bike**
berlin216	0.10&1.00	9.94	scuba diver	**bike**	geyser	fountain	fireboat	curtain	jinrikisha
berlin300	0.10&1.00	10.26	bubble	locomotive	net	fireboat	net	net	car
berlin302	0.15&1.00	9.80	spark bench	spotlight	dome	tripod	net	net	car
berlin303	0.15&1.00	9.53	wing	bannister	wing	wing	wing	volcano	car
berlin306	0.10&1.00	11.16	bubble	bubble	net	television	net	curtain	parking m.
berlin316	0.15&1.00	9.85	aircraft carrier	traffic light	wing	platform	monitor	wing	**bike**
berlin326	0.10&0.80	15.41	fountain	tank	aircraft c.	submarine	fireboat	submarine	**bike**
berlin336	0.12&1.00	11.08	maze	**bike**	**bike**	**bike**	traffic light	missile	**bike**
berlin359	0.10&1.00	11.43	parking meter	bubble	vault	abaya	net	curtain	parking m.
berlin367	0.10&0.80	16.55	fountain	car	volcano	television	snowplow	fireboat	car
berlin384	0.12&0.80	15.59	spotlight	**bike**	**bike**	curtain	net	fountain	**bike**
berlin404	0.15&1.00	9.24	parking meter	car mirror	warplane	fireboat	fireboat	fireboat	car
berlin408	0.12&0.60	20.76	aircraft carrier	**bike**	wing	fountain	wing	wing	traffic light
berlin412	0.12&0.60	20.73	bubble	bubble	fire engine	curtain	snowplow	snowplow	car
berlin417	0.15&1.00	9.85	washbasin	snowplow	curtain	fountain	. curtain	curtain	triumphal arch

(a) Weather-clear image.

(b) Smooth disparity image generated by the SGBM method.

(c) Foggy image obtained from using the weather-clear and disparity images.

Figure 4.9: An image of *bike* with fog incorrectly classified as *submarine* by the Inception-v3 model with P_{tf}=0.1, P_{al}=0.8 and PSNR=15.53.

(a) Weather-clear image.

(b) Smooth disparity image generated by the SGBM method.

(c) Foggy image obtained from using the weather-clear and disparity images.

Figure 4.10: An image of *bike* with fog incorrectly classified as *scuba diver* by the Inception-v3 model with P_{tf}=0.1, P_{al}=1 and PSNR=9.94.

(a) Weather-clear image.

(b) Smooth disparity image generated by the SGBM method.

(c) Foggy image obtained from using the weather-clear and disparity images.

Figure 4.11: An image of *bike* with fog incorrectly classified as *parking meter* by the Inception-v3 model with P_{tf}=0.1, P_{al}=1 and PSNR=11.43.

(a) Weather-clear image.

(b) Smooth disparity image generated by the SGBM method.

(c) Foggy image obtained from using the weather-clear and disparity images.

Figure 4.12: An image of *bike* with fog incorrectly classified as *fountain* by the Inception-v3 model with P_{tf}=0.1, P_{al}=0.8 and PSNR=16.55.

CHAPTER 5: IMPACT OF REAL FOG ON DEEP LEARNING SYSTEM

(a) Car misclassified as aircraft.

(b) Car misclassified as aircraft.

(c) Traffic light misclassified as fountain.

(d) Car misclassified as geyser.

Figure 5.1: Classification results obtained by Inception-v3 on real fog images of a traffic light and car.

In addition to doing analysis using our synthetically generated foggy images on the robustness of Inception-v3, one of the state-of-the-art deep learning classifiers, we also tested it with real foggy images. Our experiments demonstrate that the distance of the object from the camera has a significant effect on the classification result with the naturally-occurring fog in the images 5.7. Furthermore, traffic light is the subject that is correctly classified with most of the real fog cases as in the figure 5.8. We also see some rare cases where the first and third images are correctly classified whereas the second one is misclassified as in the figure 5.5.

(a) Real fog image-1 incorrectly classified as *monitor* by Inception-v3.

(b) Real fog image-2 incorrectly classified as *church* by Inception-v3.

(c) Real fog image-3 incorrectly classified as *fountain* by Inception-v3.

Figure 5.2: Real fog experiment - 1: Classification results of 3 consecutive frames that contain a *car* with real fog.

(a) Real fog image-1 incorrectly classified as *aircraft* by Inception-v3.

(b) Real fog image-2 incorrectly classified as *aircraft* by Inception-v3.

(c) Real fog image-3 correctly classified as *car* by Inception-v3.

Figure 5.3: Real fog experiment - 2: Classification results of 3 consecutive frames that contain a *car* with real fog. The third image is relatively closer than the other first two images and Inception-v3 returns the correct label as car.

(a) Real fog image-1 incorrectly classified as *swing* by Inception-v3.

(b) Real fog image-2 incorrectly classified as *aircraft* by Inception-v3.

(c) Real fog image-3 correctly classified as *traffic light* by Inception-v3.

Figure 5.4: Real fog experiment - 3: Classification results of 3 consecutive frames that contain a *traffic light* with real fog. The third image is relatively closer than the other first two images and Inception-v3 returns the correct label as traffic light.

(a) Real fog image-1 correctly classified as *traffic light* by Inception-v3.

(b) Real fog image-2 incorrectly classified as *aircraft* by Inception-v3.

(c) Real fog image-3 correctly classified as *traffic light* by Inception-v3.

Figure 5.5: Real fog experiment - 4: Classification results of 3 consecutive frames that contain a *traffic light* with real fog. Only the second image is misclassified as aircraft.

(a) Real fog image-1 correctly classified as *traffic light* by Inception-v3.

(b) Real fog image-2 correctly classified as *traffic light* by Inception-v3.

(c) Real fog image-3 correctly classified as *traffic light* by Inception-v3.

Figure 5.6: Real fog experiment - 5: Classification results of 3 consecutive frames that contain a *traffic light* with real fog. They are all classified as traffic light by Inception-v3.

(a) Real fog image-1 incorrectly classified as *tripod* by Inception-v3.

(b) Real fog image-2 correctly classified as *traffic light* by Inception-v3.

(c) Real fog image-3 correctly classified as *traffic light* by Inception-v3.

Figure 5.7: Real fog experiment - 6: Classification results of 3 consecutive frames that contain a *traffic light* with real fog. The first one (the farthest one from the camera) is classified as tripod.

(a) (b) (c) (d) (e) (f) (g) (h)

Figure 5.8: Real fog experiment - 7: They are all correctly classified as traffic light by Inception-v3. a, c, e, g are four consecutive frames. b, d, f, h are another four consecutive frames.

CHAPTER 6: CONCLUSION

The overall goal of this thesis was to do analysis on accuracy results and robustness of deep learning models. To achieve this goal, we propose to perturb images naturally and then test deep learning models with them. We introduce natural attacks caused by our synthetically-generated fog added on the images from the Cityscapes self-driving car dataset. Our approach uses computer graphics techniques such as stereo semi-global block matching (stereo SGBM) algorithm, in order to create disparity map using the left and right image from Cityscapes. They are called as stereo images which is a pair of images captured from two different cameras. It also uses weighted least squares (WLS) post-filtering for smoother disparity map and the results of this post-processing provided us with significantly better quality of depth images. Next, our algorithm generates fog effect by determining the density of fog with two parameters: thickness factor and atmospheric light. It increases the density by incrementing the parameter values to find the image with the lightest fog that causes the deep learning network to misclassify the outdoor object in the image.

We observed that these images with synthetically-generated fog were able to fool one of the current state-of-the-art deep learning systems, Inception-v3. Our synthetic foggy images were also tested by other most well-known deep learning models. From our experiments, our dataset with synthetic foggy images were observed to obtain the similar accuracy results with the ImageNet dataset on the 7 models we used; Inception-v3, Inception-ResNet-v2, MobileNet, ResNet-50, Xception, VGG-16, and VGG-19. For instance, the Xception model is more robust on our synthetic foggy images compared to the other six models we used for our experiments. As we reported the classified labels, we noticed that some of the fog images that contain car are still classified as *car* by Xception even though they are classified as different classes by Inception-v3. It is known that Xception has similar model size with Inception-v3. However, the reason why the Xception model has higher accuracy than the Inception model is due to its Depthwise Separable Convolution architecture and

its more efficient use of model parameters. MobileNet has similar architecture with Xception and is optimized for mobile applications. The difference is that MobileNet has two parameters that causes to have less number of operations: the width multiplier and the resolution multiplier. As for VGG networks, they have the basic architecture of Convolutional Neural Networks (CNNs); a number of convolutional layers, max-pooling, activation layers, and fully-connected layers. The 16 and 19 specify the number of layers and these were considered very deep in 2014. However, the ResNet50 model was successfully trained with 50 weight layers in 2015. This "Residual Network" proposes learning the difference between the identity x and its underlying mapping H(x), which is called "deep residual learning". It is observed that residual mapping is easier to optimize than the original unreferenced mapping. We see different classification results on our synthetic foggy images based on all these architectural differences among the models we use for our experiments.

We also tested Inception-v3 on real fog images to compare the results with the ones we obtained from our synthetic fog images. We observed the model returns similar results of misclassification for the real fog images. Hence, existing deep learning systems are vulnerable not only to digital and physical adversarial attacks, but they produce incorrect answers even when faced with benign natural perturbations. To the best of our knowledge, this is the first study on attacking deep learning classifiers with naturally perturbed images obtained from stereo dataset and depth information.

Several interesting directions for future work remain open. *First*, we want to explore the effects of other naturally occurring conditions such as rain, hail and snow not only on the deep learning image classification systems, but also on the DL models for image segmentation. *Second*, we will test the robustness of systems designed specifically for outdoor functionality such as autonomous driving systems. Adversarial learning has been playing important role in improving the performance of DL models; therefore, we aim at training or re-training models with the synthetic perturbed images found as adversary. *Third*, we will explore the design of defense algorithms that can permit deep neural networks to reason correctly about images with fog and other natural perturbations.

LIST OF REFERENCES

[1] M. Ozdag, S. Raj, S. Fernandes, L. Pullum, A. Velasquez, and S. Jha, "On the susceptibility of deep neural networks to natural perturbations," *AISafety 2019 (held in conjunction with IJCAI 2019 - International Joint Conference on Artificial Intelligence) - Macao, China*.

[2] N. Carlini and D. A. Wagner, "Towards evaluating the robustness of neural networks," *CoRR*, vol. abs/1608.04644, 2016.

[3] N. Papernot, P. D. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," *CoRR*, vol. abs/1511.07528, 2015.

[4] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards Deep Learning Models Resistant to Adversarial Attacks," *arXiv e-prints*, p. arXiv:1706.06083, Jun 2017.

[5] A. Kurakin, I. Goodfellow, S. Bengio, Y. Dong, F. Liao, M. Liang, T. Pang, J. Zhu, X. Hu, C. Xie, and et al., "Adversarial attacks and defences competition," *The Springer Series on Challenges in Machine Learning*, pp. 195–231, 2018.

[6] J. Tarel, N. Hautiere, L. Caraffa, A. Cord, H. Halmaoui, and D. Gruyer, "Vision enhancement in homogeneous and heterogeneous fog," *IEEE Intelligent Transportation Systems Magazine*, vol. 4, pp. 6–20, Summer 2012.

[7] A. Rozsa, E. M. Rudd, and T. E. Boult, "Adversarial diversity and hard positive generation," *CoRR*, vol. abs/1605.01775, 2016.

[8] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and Harnessing Adversarial Examples," *arXiv e-prints*, p. arXiv:1412.6572, Dec 2014.

[9] M. Ozdag, "Adversarial attacks and defenses against deep neural networks: A survey," *Procedia Computer Science*, vol. 140, pp. 152 – 161, 2018. Cyber Physical Systems and Deep Learning Chicago, Illinois November 5-7, 2018.

[10] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and Harnessing Adversarial Examples," *arXiv e-prints*, p. arXiv:1412.6572, Dec 2014.

[11] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," *CoRR*, vol. abs/1611.01236, 2016.

[12] N. Papernot, P. D. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," *CoRR*, vol. abs/1511.04508, 2015.

[13] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li, "Boosting adversarial attacks with momentum," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun 2018.

[14] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *CoRR*, vol. abs/1512.00567, 2015.

[15] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," 2016.

[16] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," *Lecture Notes in Computer Science*, pp. 630–645, 2016.

[17] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble Adversarial Training: Attacks and Defenses," *arXiv e-prints*, p. arXiv:1705.07204, May 2017.

[18] F. Liao, M. Liang, Y. Dong, T. Pang, J. Zhu, and X. Hu, "Defense against adversarial attacks using high-level representation guided denoiser," *CoRR*, vol. abs/1712.02976, 2017.

[19] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh, "ZOO: Zeroth Order Optimization based Black-box Attacks to Deep Neural Networks without Training Substitute Models," *arXiv e-prints*, p. arXiv:1708.03999, Aug 2017.

[20] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *CoRR*, vol. abs/1710.08864, 2017.

[21] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adversarial examples and black-box attacks," *CoRR*, vol. abs/1611.02770, 2016.

[22] N. Carlini, G. Katz, C. Barrett, and D. L. Dill, "Ground-truth adversarial examples," *CoRR*, vol. abs/1709.10207, 2017.

[23] A. M. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," *CoRR*, vol. abs/1412.1897, 2014.

[24] S. Sabour, Y. Cao, F. Faghri, and D. J. Fleet, "Adversarial manipulation of deep representations," *CoRR*, vol. abs/1511.05122, 2015.

[25] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *CoRR*, vol. abs/1312.6199, 2013.

[26] S. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," *CoRR*, vol. abs/1610.08401, 2016.

[27] K. R. Mopuri, U. Garg, and R. V. Babu, "Fast feature fool: A data independent approach to universal adversarial perturbations," *CoRR*, vol. abs/1707.05572, 2017.

[28] S. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, P. Frossard, and S. Soatto, "Analysis of universal adversarial perturbations," *CoRR*, vol. abs/1705.09554, 2017.

[29] T. B. Brown, D. Mané, A. Roy, M. Abadi, and J. Gilmer, "Adversarial patch," *CoRR*, vol. abs/1712.09665, 2017.

[30] J. Hendrik Metzen, M. Chaithanya Kumar, T. Brox, and V. Fischer, "Universal Adversarial Perturbations Against Semantic Image Segmentation," *arXiv e-prints*, p. arXiv:1704.05712, Apr 2017.

[31] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *CoRR*, vol. abs/1607.02533, 2016.

[32] M. Sharif, S. Bhagavatula, L. Bauer, and M. Reiter, "Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition," pp. 1528–1540, 10 2016.

[33] J. Lu, H. Sibai, E. Fabry, and D. A. Forsyth, "NO need to worry about adversarial examples in object detection in autonomous vehicles," *CoRR*, vol. abs/1707.03501, 2017.

[34] D. Dai, C. Sakaridis, S. Hecker, and L. V. Gool, "Curriculum model adaptation with synthetic and real data for semantic foggy scene understanding," *CoRR*, vol. abs/1901.01415, 2019.

[35] A. Ramanathan, L. Pullum, Z. Husein, S. Raj, N. Torosdagli, S. Pattanaik, and S. K. Jha, "Adversarial attacks on computer vision algorithms using natural perturbations," in *2017 Tenth International Conference on Contemporary Computing (IC3)*, pp. 1–6, Aug 2017.

[36] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012.

[37] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[38] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *CoRR*, vol. abs/1409.4842, 2014.

[39] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, June 2009.

[40] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," *CoRR*, vol. abs/1604.01685, 2016.

[41] H. Hirschmuller, "Stereo processing by semiglobal matching and mutual information," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, pp. 328–341, Feb 2008.

[42] S. Birchfield and C. Tomasi, "A pixel dissimilarity measure that is insensitive to image sampling," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, pp. 401–406, April 1998.

[43] S. Birchfield and C. Tomasi, "Depth discontinuities by pixel-to-pixel stereo," in *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, pp. 1073–1080, Jan 1998.

[44] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," 2016.

[45] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017.

[46] S. Xie, R. Girshick, P. Dollar, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul 2017.

[47] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul 2017.

[48] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014.