A Study of Adversarial Attacks on Image Classification DNNs

Sumedh Vijay Naik
Georgia Institute of Technology
Atlanta, USA
snaik65@gatech.edu

Ahmed Aqeel Shaikh
Georgia Institute of Technology
Atlanta, USA
ashaikh65@gatech.edu

Amogh Jayant Dabholkar
Georgia Institute of Technology
Atlanta, USA
adabholkar6@gatech.edu

Brahmi Dwivedi
Georgia Institute of Technology
Atlanta, USA
bdwivedi6@gatech.edu

Abstract

Deep Neural Networks (DNNs) have attained state-of-the-art performances on most computer vision problems in today's world. They have beaten a lot of traditional machine learning approaches at tasks and continue to surpass their own levels as the hardware acceleration for learning keeps getting better. However, despite their scalability and efficiency, most of the DNNs deployed for computer vision tasks are vulnerable to adversarial attacks. Since it was first discovered back in 2014, there has been a lot of work in this domain of generating attacks to fool these DNNs. This project aims to implement, compare and analyze the effects of popular adversarial attacks on DNNs that are used for the image classification task. It will also use the common visualization technique - GradCAM to visualize where the different neurons in the pre-final layer of these DNNs 'look at' before predicting the class.

1. Introduction

Over the last decade, researchers have made tremendous strides using Deep Neural Networks (DNNs), an array of algorithms and architectures have been designed to solve important problems when it comes to image and text data. However, these deep learning models are vulnerable to adversarial attacks applied to input data. Such adversarial attacks manipulate the data in a way that leads to models incorrectly classifying the data during evaluation.

Adversarial attacks can be categorized into white-box and black-box attacks on the basis of the level of accessibility that the attacker has to the model and its predictions. In the case of white-box attacks, the attacker has access to the model architecture and model parameters, while in the case of black-box attacks, the attacker has access to the predictions made by the model but does not have access to the model itself. Figure 1 shows this classification.

Attacks can be categorized into poisoning attacks and evasion attacks on the basis of when they were launched. In poisoning attacks, the training data or its labels are tampered with. This may be implemented when the model is being trained for the first time or when the model has been deployed but needs to be re-trained with new data. Evasion attacks are implemented when the model has been deployed. In this, the input data is maliciously modified to fool the model which has been trained.

In some cases, the perturbations made by adversarial attacks in images are so subtle that they cannot be noticed by a human, yet the DNN classifier fails to correctly classify the image. Figure 2 shows such an example. The adversarial image is not distinguishable from the original, but the model is unable to identify the image correctly. Adversarial attacks pose a threat to security because they could be used by unauthorised personnel to attack deep learning models[7] [1]. This is a matter of concern in safety-critical applications such as healthcare, autonomous vehicles, flight control and so on.

In this project, we have studied adversarial attacks such as the Fast Gradient Sign Method (FGSM)[12] [5] attack, the Basic Iterative Method (BIM) [3] attack, and the DeepFool attack[9]. These attacks fall into the white-box and evasion attack categories. We study the effect of the attacks on Deep Neural Networks trained to classify images on the MNIST and CIFAR-10 datasets. Furthermore, we have trained models on the CIFAR-10 dataset and studied the transferability of adversarial attacks by using examples generated for one model for evaluation by another. Lastly, we use Grad-CAM[10], a visualisation technique that allows us to see where a deep learning model focuses on in an image during evaluation. For visualisations, we make use of models trained on the ImageNet dataset and the FGSM attack. We use the ImageNet dataset[2] for visualisations because MNIST[8] and CIFAR-10[6] contain low-resolution images which do not allow for intelligible differences when we use Grad-CAM.

In this paper, Section 2 talks about the methodology in which, 2.1 contains details about the datasets that were used and how they were used. 2.2 to 2.4 elaborates on each of the adversarial attacks that were used in this work. 2.5 talks about the Grad-CAM technique which has been used to visualize the pre-final layer neurons. That is followed by Section 3 which contains all the details of experiments conducted and the corresponding results. Section 4 concludes this work and is followed by References.

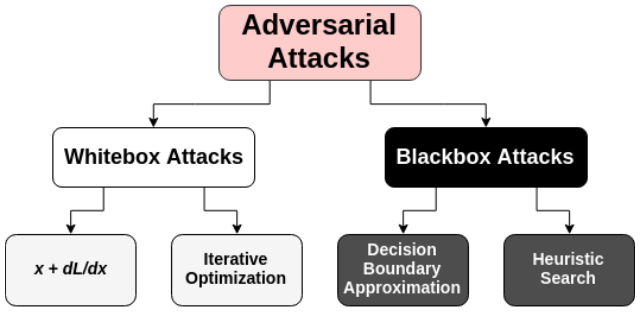


Figure 1. Classification on Accessibility

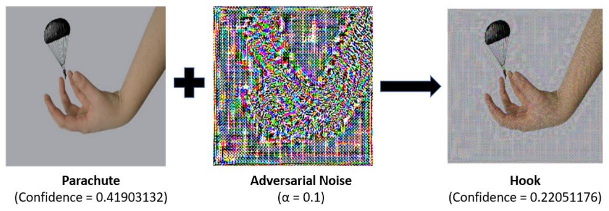


Figure 2. Adversarial Example

2. Methodology

In this section, we will be discussing the datasets used in this project as well as the approaches undertaken for the generation of adversarial images.

2.1. Datasets used for the experiments

- **CIFAR-10** - This dataset has 60000 images of size 32x32x3, each image belonging to one of 10 classes. The classes include birds, airplanes, ships, trucks, dogs etc. The CIFAR-10 dataset has low-resolution images which allow fast and easy training for testing a neural network algorithm. CIFAR-10 is one of the most commonly used benchmark datasets when it comes to computer vision and deep learning research.
- **MNIST** - This dataset has 60000 training images and 10000 test images of handwritten digits from 0-9. Several researchers have used the MNIST dataset and achieved human-like performance using CNNs and other Deep Learning architectures. The creators of the database had used Support Vector Machines for their original paper and achieved an error rate of just 0.8%.
- **ImageNet (Subset)** - ImageNet is a widely-used open source dataset consisting of more than 14 million images organized into more than 20000 different classes. While it is extensively used in image processing and image classification applications, we use a subset of the ImageNet for our experiments. We tested our attack models and obtained results on images from 10 classes.

2.2. FGSM (Fast Gradient Sign Method) Attack

Fast Gradient Sign Method (FGSM) is a very effective untargeted attack method to generate adversarial images to attack datasets. It uses the gradients of a loss function and then messes with its sign in such a way that it maximizes the loss instead of minimizing it. Thus it creates an image that a human eye may perceive to be identical, but forces the neural network into making a wrong prediction. This new image can be called an adversarial image. The following expression can be used to summarize the FGSM attack:

$$adv\_x = x + \epsilon * \text{sign}\left(\nabla_x J(\theta, x, y)\right)$$

where,

- adv_x : Adversarial image.
- x : Original input image.
- y : The ground-truth input label.
- ϵ : The tiny value multiplied with the gradient to ensure smaller perturbation such that the human eye cannot detect it.
- θ : DNN parameters.
- J : Loss function.

Essentially, the FGSM attack consists of 3 steps in the following order:

- Evaluate the forward propagation loss.
- Evaluate the gradient wrt the image pixels.
- Image pixels are nudged lightly in the direction of the evaluated gradients which maximize the loss.

Figure 3 explains the concept in a simpler fashion. In the figure, the equation on the left is the usual training update rule, whereas FGSM uses the equation mentioned on the right.

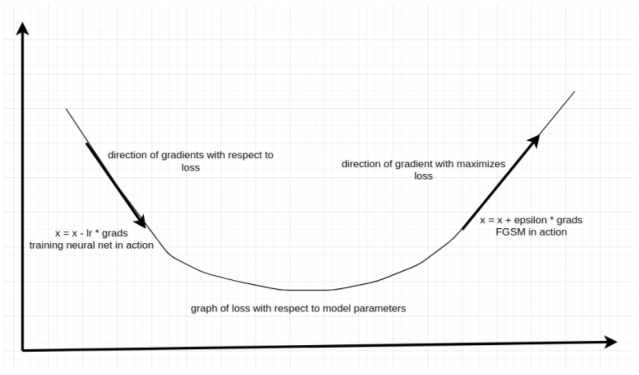

direction of gradients with respect to loss
direction of gradient with maximizes loss
x = x - lr * grads
training neural net in action
x = x + epsilon * grads
FGSM in action
graph of loss with respect to model parameters

Figure 3. Working of FGSM

2.3. BIM (Basic Iterative Method)

Basic Iterative Method (BIM) [3] is an extension of the previously discussed FGSM attack. The idea is to do FGSM iteratively while clipping the output of the perturbation as it steps so that the generated adversarial image still remains within both, the ϵ neighbourhood as well as the input space. Few papers sometimes also call BIM the Iterative FGSM (I-FGSM). This attack allows you more control over the attack. The following update rule can be used to summarize the BIM attack:

$$\boldsymbol{X}_0^{adv} = \boldsymbol{X}$$

$$\boldsymbol{X}_{N+1}^{adv} = \mathrm{Clip}_{X,\epsilon}\left\{\boldsymbol{X}_N^{adv} + \alpha\,\mathrm{sign}\left(\nabla_X J\left(\boldsymbol{X}_N^{adv}, y_{true}\right)\right)\right\}$$

where,

- $\boldsymbol{X}_N^{adv}$: Adversarial image at the i^{th} iteration.
- $\boldsymbol{X}$: Original input image.
- y_{true} : The ground-truth input label.
- ϵ : Tunable parameter.
- α : Step size.
- J : Loss function.

2.4. DeepFool Adversarial Attack

DeepFool is an attack used on classifier models. It iteratively perturbs the image until the attacked image is wrongly classified by the model. The perturbations across the image are then summed up and added to the original image to generate an attacked image.

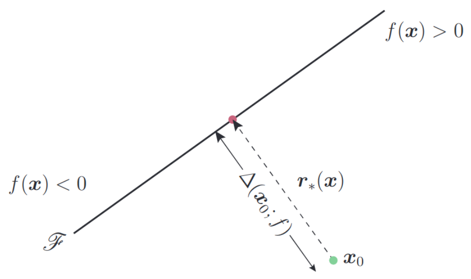


Figure 4. Example of DeepFool on a Linear Classifier

For example, as shown in Figure 4, if an image x_0 is classified correctly into its respective class, the DeepFool model finds the nearest hyper-plane (f(x) < 0) and perturbs the image to move in the direction of it. The iterative process takes place till the image just about reaches the other side of the hyper-plane, thus inducing misclassification. The DeepFool algorithm can be written as shown in Algorithm 1.

Algorithm 1: DeepFool Algorithm

```
Input: Input Image $x$ and its nearest classifier $f$
Output: Perturbations $\hat{r}$
1 $x_0 \leftarrow x$ , $i \leftarrow 0$
2 while $\mathrm{sign}(f(x_i)) = \mathrm{sign}(f(x_0))$ do
3     $r_i \leftarrow \frac{-f(x_i)}{\|\nabla f(x_i)\|_2^2}$
4     $x_{i+1} \leftarrow x_i + r_i$
5     $i \leftarrow i + 1$
6 return $\hat{r} = \sum_i r_i$
7
```

2.5. Grad-CAM

Grad-CAM [10] [11] is a visualization technique that aids the explainability of decision making in Convolutional Neural Networks. It allows us to view the regions of the image the network is looking at for a particular predicted class in the form of a heat map, utilizing the gradients for that particular class in the last convolutional layer. An advantage of using Grad-CAM is that it can be used to perform visualizations on a variety of CNN models without architectural changes or re-training. This visualization is particularly useful to analyze failures of these CNNs- to understand the reasoning behind and diagnose incorrect predictions.

Since the last layer of CNNs retains spatial information of the image and contains high-level semantics as well, Grad-CAM uses gradients in this layer to assign importance to neurons for a particular class for a chosen image. The class-discriminative localization map for a class c is obtained by first calculating the gradient of the class c's score y^c (prior to applying softmax) with respect to the feature map activations A^k, (where we apply k filters in the convolutional layer) of a convolutional layer. The global average pooling of these gradients over the width and height dimensions gives the neuron importance weights α_k^c:

$$\alpha_k^c = \frac{1}{Z}\sum_i\sum_j \frac{\partial y^c}{\partial A_{ij}^k}$$

($Z = u * v =$ the dimensions of the feature maps)

These weights capture the importance of feature map k for the class c are then used to calculate a weighted combination of the forward activation maps, followed by ReLU (ReLU is applied as we are only interested in features having a positive influence on the selected class c), to obtain the heatmap $L^c_{Grad-CAM}$:

$$L^c_{Grad-CAM} = ReLU(\sum_k \alpha^c_k A^k)$$

The heatmap produced has the same dimensions as the feature maps. This heatmap can be upsampled to the input image resolution using bilinear interpolation.

3. Experiments

We have used pre-trained classification models for both MNIST and CIFAR-10 datasets. These pre-trained models are attacked using FGSM, BIM, and DeepFool. We apply these attacks on 100 samples each. We observe the change in the images, and the Test Accuracy and Mean Confidence of the models on these perturbed images. The pre-trained models we have used are a CNN model for MNIST and the DenseNet model[4] for CIFAR-10, the performance of these models on benign images are shown in Table 1 and Table 2 respectively. The performance of the model we have trained on CIFAR-10 can be seen in Table 3.

Table 1. Performance Statistics of CNN Model on Benign MNIST Examples

Test Accuracy	Mean Confidence	Test Time Per Sample (Seconds)
99.43%	99.39%	0.000563

Table 2. Performance Statistics of DenseNet Model on Benign CIFAR-10 Examples

Test Accuracy	Mean Confidence	Test Time Per Sample (Seconds)
94.84%	92.15%	0.002619

Table 3. Performance Statistics of Trained Model on Benign CIFAR-10 Examples

Test Accuracy	Mean Confidence	Test Time Per Sample (Seconds)
86.07%	83.85%	0.002243

3.1. FGSM Attack

We applied the FGSM attack on both pre-trained models, the attack was successful in dramatically reducing the test accuracy of both models. However, FGSM is a one-shot attack algorithm, the DeepFool and BIM attacks are able to fool the model at a higher rate for the MNIST dataset. Since FGSM uses a one-shot approach it's evaluation time is low. Figure 5 shows examples of FGSM applied on MNIST data samples, Figure 6 shows the same for CIFAR-10 data samples. The perturbations are clearly visible in the MNIST samples, but are not noticeable in the CIFAR-10 samples. Table 4 and Table 5 show the statistics of the pretrained models under the FGSM attack.

Table 6 shows how the model we trained performs on adversarial samples generated on the DenseNet model. The FGSM attack has poor transferability as is apparent from the results. The time taken per sample is the same as DenseNet because the evaluation is done together.

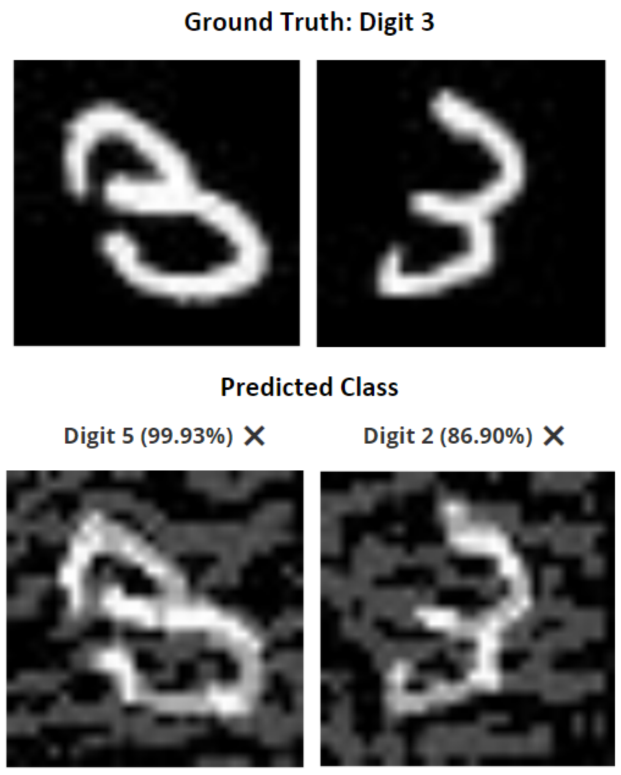


Figure 5. MNIST data under FGSM attack

Table 4. Performance Statistics of CNN Model on Adversarial MNIST Examples generated using FGSM

Test Accuracy	Mean Confidence	Test Time Per Sample (Seconds)
54%	94.98%	0.004986

Table 5. Performance Statistics of DenseNet Model on Adversarial CIFAR-10 Examples generated using FGSM

Test Accuracy	Mean Confidence	Test Time Per Sample (Seconds)
14%	96.9%	0.085511

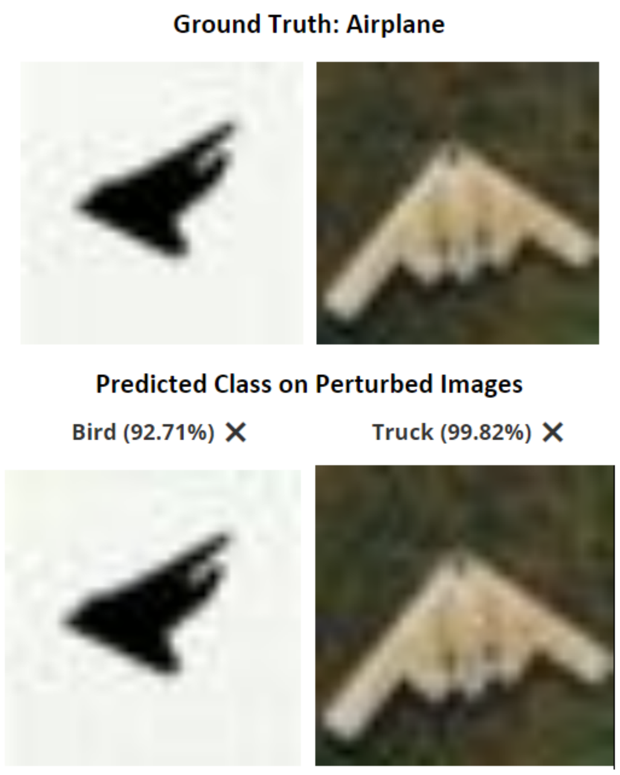


Figure 6. CIFAR-10 data under FGSM attack

Table 6. Performance Statistics of Trained Model on Adversarial CIFAR-10 Examples generated using FGSM for DenseNet Model

Test Accuracy	Mean Confidence	Test Time Per Sample (Seconds)
67%	85.93%	0.085511

3.2. DeepFool Attack

We applied the DeepFool attack on both pretrained models, the attack was successful against both models. DeepFool has a much higher success rate because the DeepFool attack keeps going until it fools the model. The confidence level is lower than seen in FGSM. Figure 7 shows examples of DeepFool applied on MNIST data samples, Figure 8 shows the same for CIFAR-10 data samples. The perturbations are visible in the MNIST samples, but they appear more realistic than the FGSM ones. DeepFool also takes a lot more time per sample. Table 7 and Table 8 show the statistics of the pretrained models under the DeepFool attack.

Table 9 shows how the model we trained performs on adversarial samples generated on the DenseNet model. The DeepFool attack has poor transferability as is apparent from the results. The success of the attack is limited to the model it is attacking.

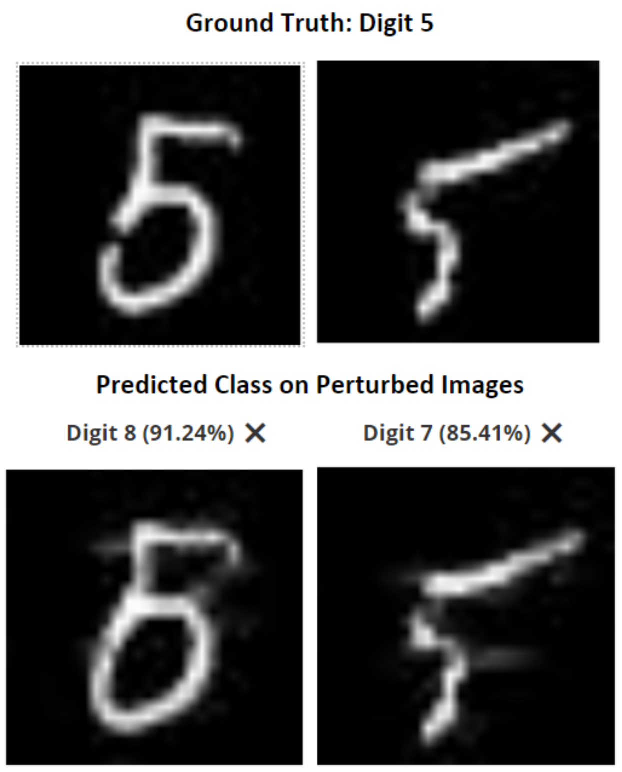
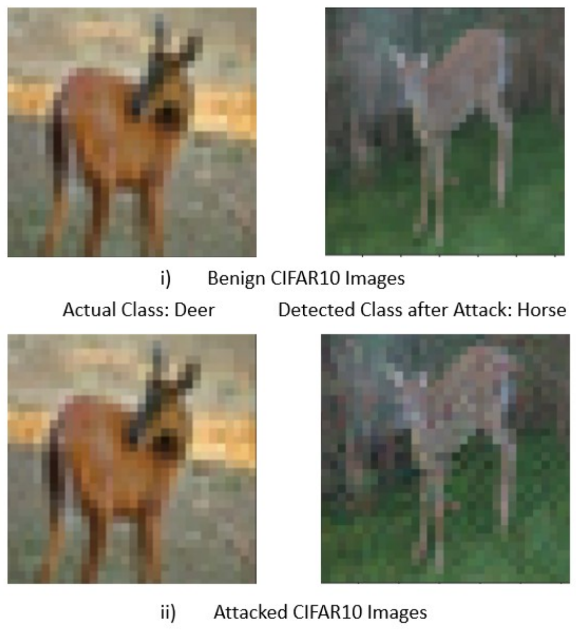


Figure 7. MNIST data under DeepFool attack


i) Benign CIFAR10 Images
Actual Class: Deer
Detected Class after Attack: Horse
ii) Attacked CIFAR10 Images
Figure 8. CIFAR-10 data under DeepFool attack

3.3. BIM Attack

We applied the BIM attack with 10 iterations on both pretrained models, both models were able to achieve an ac-

Table 7. Performance Statistics of CNN Model on Adversarial MNIST Examples generated using DeepFool

Test Accuracy	Mean Confidence	Test Time Per Sample (Seconds)
0%	83.76%	0.222839

Table 8. Performance Statistics of DenseNet Model on Adversarial CIFAR-10 Examples generated using DeepFool

Test Accuracy	Mean Confidence	Test Time Per Sample (Seconds)
0%	85.78%	2.575407

Table 9. Performance Statistics of Trained Model on Adversarial CIFAR-10 Examples generated using DeepFool for DenseNet Model

Test Accuracy	Mean Confidence	Test Time Per Sample (Seconds)
85%	96.04%	2.575407

curacy of just 8%. BIM iteratively applies FGSM and as a result achieves better success. Figure 9 shows examples of BIM applied on MNIST data samples for 3, 6 and 10 iterations of the attack, Figure 10 shows the same for CIFAR-10 data samples. The perturbations become more pronounced as the number of iterations increase. BIM takes more time per sample than FGSM, but is faster than DeepFool. Table 10 and Table 11 show the statistics of the pretrained models under the BIM attack.

Table 12 shows how the model we trained performs on adversarial samples generated on the DenseNet model. The BIM attack has worse transferability than the FGSM attack as our model achieves higher accuracy on the BIM attacked samples.

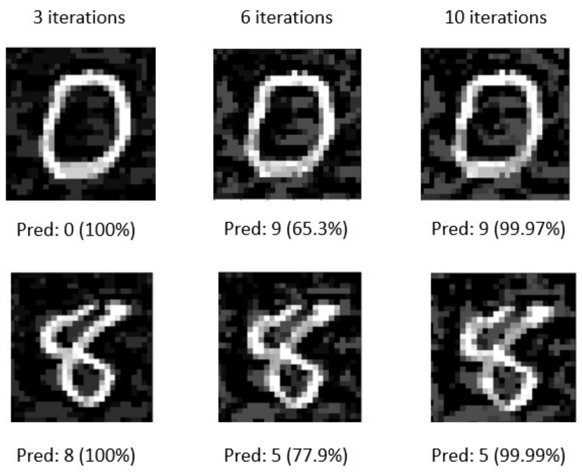


Figure 9. MNIST data under BIM attack

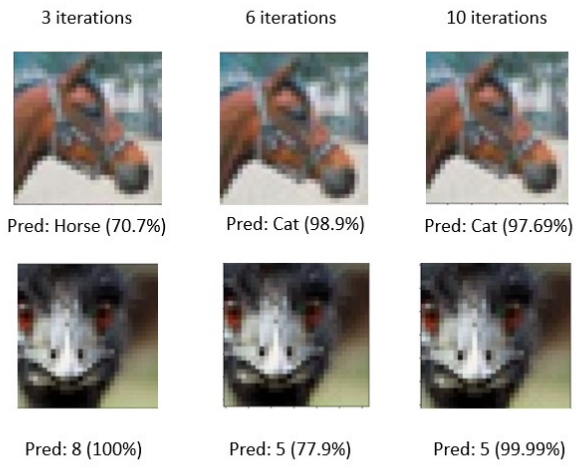


Figure 10. CIFAR10 data under BIM attack

Table 10. Performance Statistics of CNN Model on Adversarial MNIST Examples generated using BIM

Test Accuracy	Mean Confidence	Test Time Per Sample (Seconds)
8%	99.84%	0.060513

Table 11. Performance Statistics of DenseNet Model on Adversarial CIFAR-10 Examples generated using BIM

Test Accuracy	Mean Confidence	Test Time Per Sample (Seconds)
8%	98.85%	1.495455

Table 12. Performance Statistics of Trained Model on Adversarial CIFAR-10 Examples generated using BIM for DenseNet Model

Test Accuracy	Mean Confidence	Test Time Per Sample (Seconds)
81%	85.72%	1.495455

3.4. GradCam on Adversarial Examples

GradCam visualization was applied on the original as well as the adversarially attacked images to observe the shifts in attention, if any. On the MNIST and CIFAR10 images, the model looked at almost similar regions to determine the class of the image. One of the main reasons for the same is the size of the images in the CIFAR10 and MNIST datasets, and hence, we used a model trained on ImageNet to visualise the effect of the FGSM attack.

As shown in Figure 11, the attacked image made the model look at different locations to determine the class of the image. The added impurity (adversarial noise) shifts the attention of the model to different regions of the image explaining the misclassification. Further research into the reasons and possible remedies for the same can potentially be a future scope of this project.

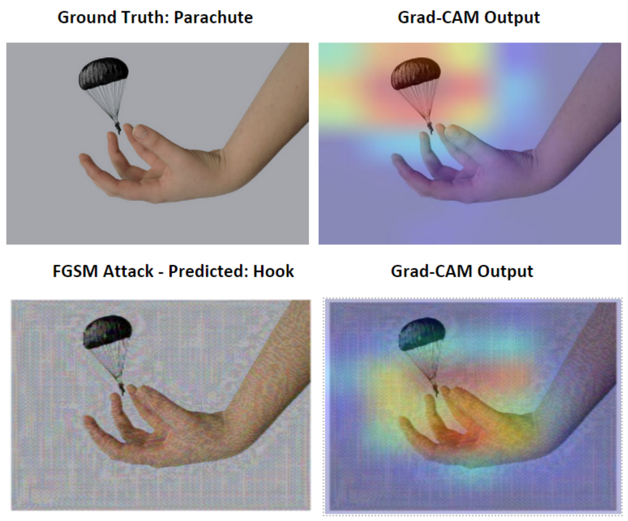


Figure 11. GradCam Visualizations on ImageNet

4. Conclusion

In this project, we studied and implemented three adversarial attacks for image classification, namely the FGSM attack, the DeepFool Attack and the BIM attack. The FGSM attack is the fastest one, however, its success rate is the lowest. DeepFool keeps attacking an image until it succeeds, as a result, it has a 100% success rate on both our models. The BIM is an iterative application of the FGSM attack with a threshold, as we increase the number of iterations the attack is increasingly successful and at the same time, the perturbations become more pronounced. We also trained another model on the CIFAR-10 dataset, we then tested the transferability of the adversarial examples. FGSM was the most successful in fooling the new model, however, the model still achieved high accuracy. Lastly, we used Grad-CAM to visualise the change in the focus area of a DNN when it evaluates an adversarial example.

Adversarial attacks is an active area of research, there are several applications of adversarial attacks, for example, adversarial attacks have been used in a constructive manner to protect privacy. Researchers are trying to incorporate measures during the training to improve the robustness of their models against adversarial attacks and at the same time, new adversarial attacks are devised to beat these measures. This back and forth leads to a continuous improvement in the security of Deep Learning models.

References

[1] Naveed Akhtar and Ajmal Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access*, 6:14410–14430, 2018. 1

[2] 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*, pages 248–255, 2009. 2

[3] R. Feinman. R. Curtin, S. Shintre, A. Gardner,"Detecting Adversarial Samples from Artifacts," arXiv preprint. 1, 3

[4] Z. Liu G. Huang. *K.* Q.Weinberger, and L. Maaten. Densely connected convolutional networks. In CVPR, 2017. 4

[5] I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. arXiv preprint. 1

[6] A. Krizhevsky. *Learning multiple layers of features from tiny images*. Tech Report, 2009. 2

[7] Alexey Kurakin, Ian Goodfellow, Samy Bengio, et al. Adversarial examples in the physical world, 2016. 1

[8] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 86(11), November 1998. 2

[9] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR*, pages 2574–2582, 2016. 1

[10] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *2017 IEEE International Conference on Computer Vision (ICCV*, pages 618–626, 2017. 1, 3

[11] Ramprasaath R Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. Grad-cam: Why did you say that? *arXiv preprint arXiv:1611.07450*, 2016. 3

[12] X. Yuan, P. He, Q. Zhu, and X. Li. Adversarial examples: Attacks and defenses for deep learning. *IEEE Transactions on Neural Networks and Learning Systems*, 30:9, September 2019. 1