Adversarial Machine Learning

A Project Report Submitted
in Partial Fulfillment of the Requirements
for the Degree of

Bachelor of Technology

by

Vaibhav Nagrale
(112001046)

INDIAN INSTITUTE OF TECHNOLOGY PALAKKAD

COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY PALAKKAD

<u>CERTIFICATE</u>

*This is to certify that the work contained in the project entitled "**Adversarial Machine Learning**" is a bonafide work of **Vaibhav Nagrale (Roll No. 112001046)**, carried out in the Department of Computer Science and Engineering, Indian Institute of Technology Palakkad under my guidance and that it has not been submitted elsewhere for a degree.*

Vivek Chaturvedi

Assistant/Associate Professor

Department of Computer Science & Engineering

Indian Institute of Technology Palakkad

Acknowledgements

I would like to take this opportunity to express my deepest gratitude to my mentor, **Vivek Chaturvedi**.

Vaibhav Nagrale

IIT Palakkad

Date: 27/10/2023

Abstract

In the pursuit of my Bachelor's Thesis Project (BTP) in Adversarial Machine Learning, my primary objective was to explore the development of novel attack or defense strategies.

I conducted an extensive review of research papers on various attack methods and applied pre-existing attacks to different datasets and models. Through rigorous experimentation, I systematically tested a range of attacks, each tailored to specific perturbations, and meticulously documented their outcomes. Additionally, I employed graphical representations and the GradCAM visualization technique to provide insightful and visually compelling observations. I also implemented RED Attack, Resource Efficient Decision-based Imperceptible Attack for Machine Learning.

This report encapsulates the comprehensive findings and insights gained from these investigations, contributing to the ever-evolving landscape of adversarial machine learning research.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction to Adversarial Machine Learning

Adversarial machine learning is the study of the attacks on machine learning algorithms, and of the defenses against such attacks. It focuses on exploiting vulnerabilities in machine learning models by crafting adversarial inputs to deceive them.

1.2 Significance and motivation for the BTP

The motivation for this BTP stems from the critical issue of adversarial attacks on artificial intelligence systems. As highlighted in the research paper "Review of Artificial Intelligence Adversarial Attack and Defense Technologies," [1] such attacks limit the potential of AI in crucial security domains. Enhancing AI robustness against adversarial threats is paramount for advancing the field, making our BTP a significant endeavor.

1.3 Research objectives

The primary objective of this study is to explore the development of novel attack or defense strategies, aiming to advance the understanding and capabilities in the field of adversarial machine learning. Additionally, it seeks to contribute to the ongoing efforts to improve the security and robustness of machine learning models in the face of adversarial attacks.

Chapter 2

Literature Review

2.1 Insights into Adversarial Attacks

To gain valuable insights into the realm of adversarial attacks and their implications within the domain of security, I extensively reviewed a selection of research papers on BlackBoxAttacks and WhiteBoxAttacks.

2.1.1 BlackBoxAttacks

- **Decision Based** : Focuses on altering model outputs without access to model internals. [2]
- **Gen Attack** : Evolves adversarial examples using genetic algorithms. [3]
- **Swarm Optimization** : Utilizes swarm intelligence for crafting adversarial inputs. [4]

2.1.2 WhiteBoxAttacks

- **Carlini Wagner** : Generates adversarial examples using an optimization-based approach. [5]
- **FGSM** : Perturbs input data based on gradient information. [6]

- **JSMA** : Manipulates features based on Jacobian saliency maps.

[7]

2.2 Conclusion

I got an overview on ways of attacking and types of attack under them. It has shed light on different strategies employed by researchers and attackers to manipulate machine learning models, exposing vulnerabilities in their decision-making processes.

Chapter 3

Methodology

After getting idea on adversarial attacks by reading research papers I implemented some pre-existing attacks on model. This section has overview of attacks implementation.

3.1 Description of Datasets:

- **ImageNet** : I utilized the ImageNet dataset, an extensive online dataset, for my research. ImageNet comprises a diverse collection of images spanning various categories, making it a widely used resource in computer vision and machine learning.
- **X-ray Pneumonia** : In addition, I incorporated the X-ray Pneumonia dataset, which exclusively contains X-ray images of patients diagnosed with pneumonia. This dataset served as a specific and targeted subset relevant to the medical domain.

3.2 Overview of Model:

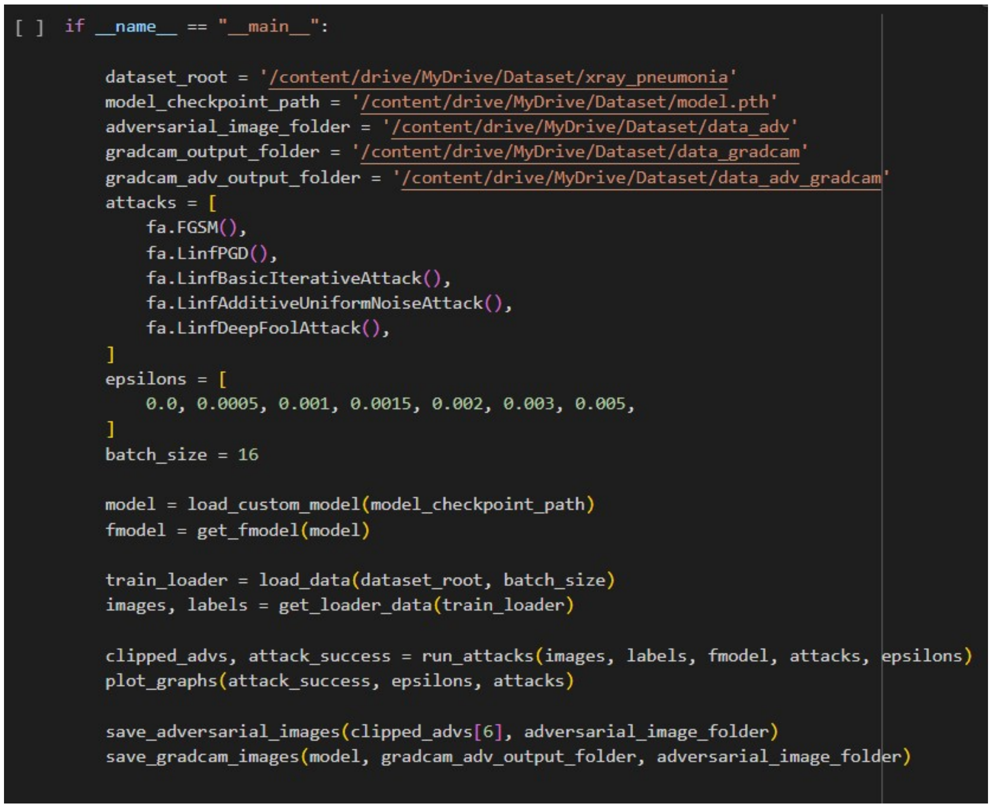
For my experiments, I selected the ResNet-18 model as the backbone. ResNet-18 is a well-established convolutional neural network architecture known for its robust performance in image classification tasks. It offers a suitable balance between accuracy and computational efficiency, making it an ideal choice for this study.

3.3 Attack Strategies Applied:

In this research, I implemented a series of adversarial attack strategies to evaluate the model's robustness against adversarial perturbations. These included:

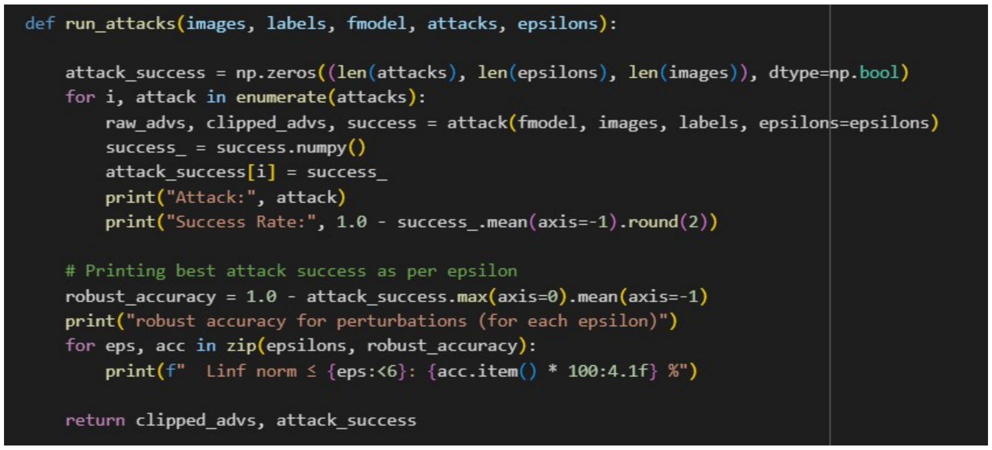
- **Fast Gradient Sign Method (FGSM)** : FGSM is a gradient-based attack method that generates adversarial examples by perturbing input data based on the sign of the gradient with respect to the loss function.
- **Projected Gradient Descent (PGD)** : PGD is an iterative variant of FGSM that applies multiple iterations to produce more potent adversarial examples.
- **Basic Iterative Attack** : This method extends the iterative approach to craft adversarial examples by perturbing the input multiple times.
- **Fast Minimum Norm Attack (FMN)** : The goal of FMNAttack is to generate minimal perturbations that, when applied to input data, lead to misclassifications by the model.
- **DeepFool Attack** : The DeepFool Attack is an iterative method that computes the smallest perturbation needed to mislead the model.

I executed these attacks using the Foolbox library in Python, a versatile and widely-used library for crafting and evaluating adversarial attacks.



```
[ ] if __name__ == "__main__":

        dataset_root = '/content/drive/MyDrive/Dataset/xray_pneumonia'
        model_checkpoint_path = '/content/drive/MyDrive/Dataset/model.pth'
        adversarial_image_folder = '/content/drive/MyDrive/Dataset/data_adv'
        gradcam_output_folder = '/content/drive/MyDrive/Dataset/data_gradcam'
        gradcam_adv_output_folder = '/content/drive/MyDrive/Dataset/data_adv_gradcam'
        attacks = [
            fa.FGSM(),
            fa.LinfPGD(),
            fa.LinfBasicIterativeAttack(),
            fa.LinfAdditiveUniformNoiseAttack(),
            fa.LinfDeepFoolAttack(),
        ]
        epsilons = [
            0.0, 0.0005, 0.001, 0.0015, 0.002, 0.003, 0.005,
        ]
        batch_size = 16

        model = load_custom_model(model_checkpoint_path)
        fmodel = get_fmodel(model)

        train_loader = load_data(dataset_root, batch_size)
        images, labels = get_loader_data(train_loader)

        clipped_advs, attack_success = run_attacks(images, labels, fmodel, attacks, epsilons)
        plot_graphs(attack_success, epsilons, attacks)

        save_adversarial_images(clipped_advs[6], adversarial_image_folder)
        save_gradcam_images(model, gradcam_adv_output_folder, adversarial_image_folder)
```

Fig. 3.1: Code snippet showing five distinct attacks



```
def run_attacks(images, labels, fmodel, attacks, epsilons):

    attack_success = np.zeros((len(attacks), len(epsilons), len(images)), dtype=np.bool)
    for i, attack in enumerate(attacks):
        raw_advs, clipped_advs, success = attack(fmodel, images, labels, epsilons=epsilons)
        success_ = success.numpy()
        attack_success[i] = success_
        print("Attack:", attack)
        print("Success Rate:", 1.0 - success_.mean(axis=-1).round(2))

    # Printing best attack success as per epsilon
    robust_accuracy = 1.0 - attack_success.max(axis=0).mean(axis=-1)
    print("robust accuracy for perturbations (for each epsilon)")
    for eps, acc in zip(epsilons, robust_accuracy):
        print(f"  Linf norm ≤ {eps:<6}: {acc.item() * 100:4.1f} %")

    return clipped_advs, attack_success
```

Fig. 3.2: Code snippet showing attack strategy

This code excerpt highlights the implementation of five distinct adversarial attacks(Fig. 3.1), showcasing their methodologies(Fig. 3.2). Additionally, it generates a graph depicting attack success versus epsilon and saves Grad-CAM visualizations for the adversarial images. The complete code is available here.

Chapter 4

Results and Analysis

This section has the results of the analysis by applying a series of adversarial attacks to the model. The attacks are executed with varying perturbation magnitudes (epsilon) to assess the model's performance under different levels of adversarial influence.
Here are graphical results for different attacks :

4.1 Robustness Analysis using Adversarial Attacks:

In figures below we see decrease in robust accuracy with increase in epsilon depicting success of attack for decreasing accuracy of model.

- **Fast Gradient Sign Method:** (Fig. 4.1)

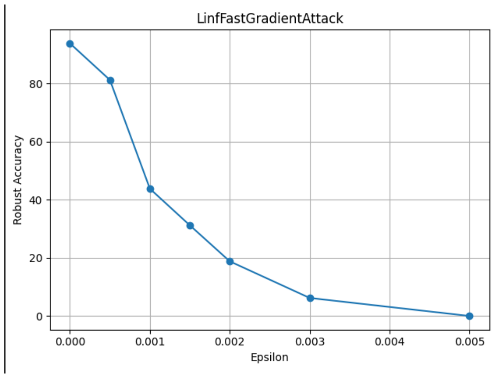


Fig. 4.1: Fast Gradient Sign Attack Graph

- **Projected Gradient Descent:** (Fig. 4.2)

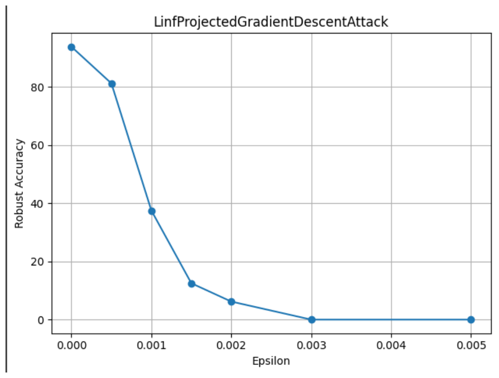


Fig. 4.2: Projected Gradient Descent Attack Graph

- **Basic Iterative Attack:** (Fig. 4.3)

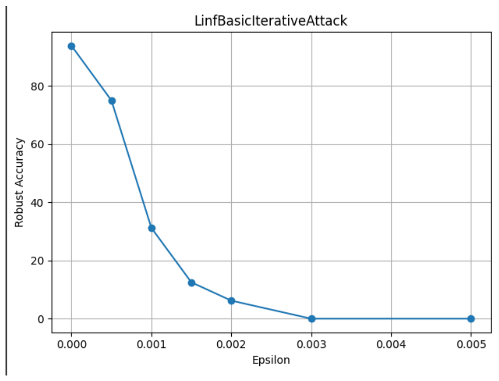


Fig. 4.3: Basic Iterative Attack Graph

- **Fast Minimum Norm Attack:** (Fig. 4.4)

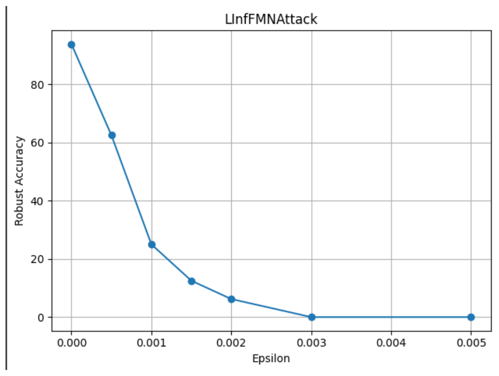


Fig. 4.4: Fast Minimum Norm Attack Graph

- **DeepFool Attack:** (Fig. 4.5)

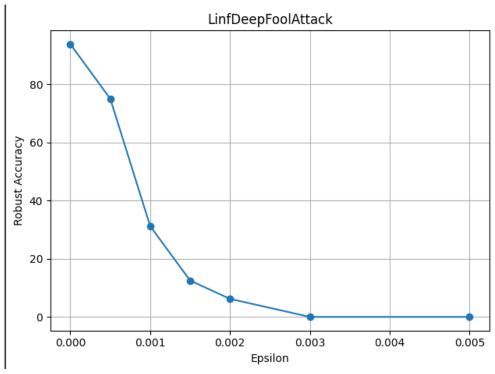


Fig. 4.5: DeepFool Attack Graph

4.2 Visual Insights using GradCAM:

In addition to quantitative analysis, I employed GradCAM (Gradient-weighted Class Activation Mapping) to provide a qualitative perspective on model behavior. GradCAM images for select attack scenarios are included to reveal the regions of the image that the model focuses on during classification decisions.

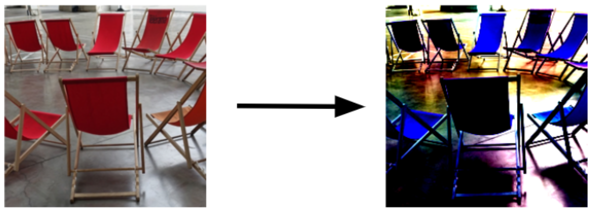


Fig. 4.6: Chair GradCAM image (imagenet dataset)

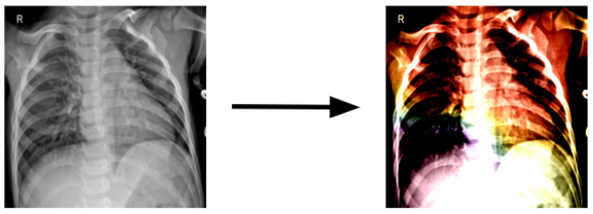


Fig. 4.7: Lungs GradCAM image (xray pneumonia dataset)

In Figure (Fig. 4.6), we observe the GradCAM of an adversarial image depicting a chair.

Notably, the chair appears in blue, signifying a shift in the model's attention away from the chair and consequently leading to incorrect predictions. Similarly, in Figure (Fig. 4.7), the presence of the red region in the upper portion of the X-ray image, due to adversarial perturbations, results in the model focusing on the wrong area, leading to inaccurate predictions.

This structured approach allows for a clear and organized presentation of the results and analysis section, combining quantitative metrics with qualitative insights for a comprehensive evaluation of the model's performance under adversarial conditions.

Chapter 5

Resource Efficient Decision-based Imperceptible Attack

5.1 Introduction

In this research, we introduce a Resource-Efficient Decision-Based methodology tailored for the generation of imperceptible adversarial attacks. [8]
The underlying assumption for the proposed methodology is that it takes the pre-processed image and its corresponding class label from the black-box model. Then it iteratively generates the adversarial image multiple time, computes the corresponding classification label for each iteration and optimizes it by finding the closest adversarial image from input image.

5.2 Motivation

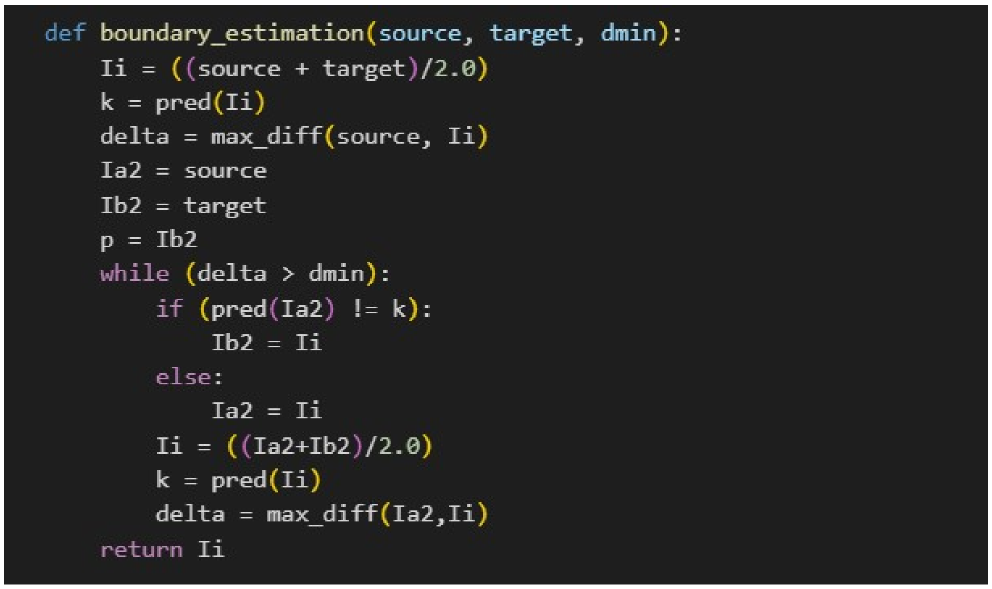
Traditional decision-based attacks require a large number of queries to generate a single untargeted attack image, which is not suitable for resource-constrained systems where real-time performance is crucial. To overcome this limitation, the RED-Attack introduces a resource-efficient methodology to generate imperceptible attacks for black-box models.

5.3 Methodology

The RED-Attack consists of two main steps: classification boundary estimation and adversarial noise optimization.

1. Classification Boundary Estimation:

   - The proposed half-interval search-based algorithm estimates a sample on the classification boundary using a target image and a randomly selected image from another class.



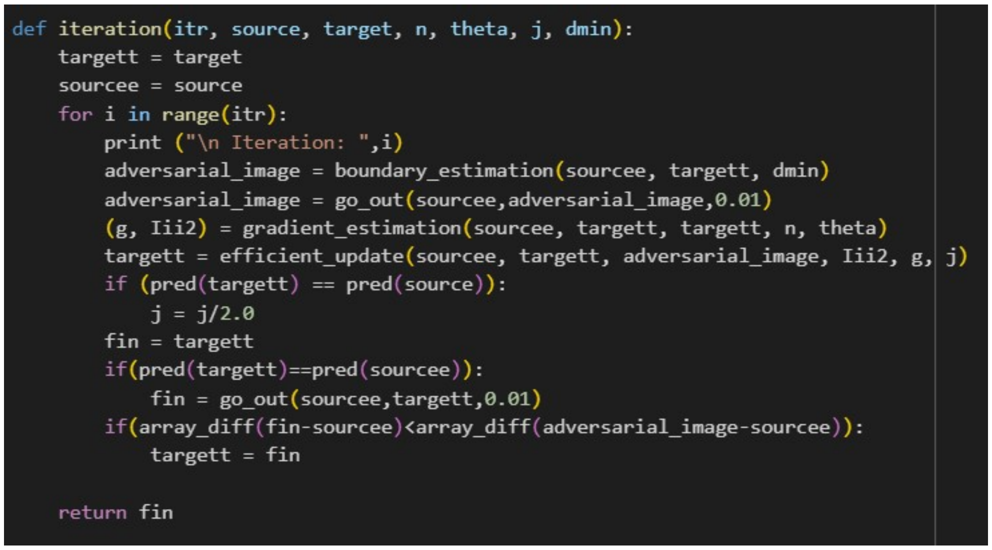
```
def boundary_estimation(source, target, dmin):
    Ii = ((source + target)/2.0)
    k = pred(Ii)
    delta = max_diff(source, Ii)
    Ia2 = source
    Ib2 = target
    p = Ib2
    while (delta > dmin):
        if (pred(Ia2) != k):
            Ib2 = Ii
        else:
            Ia2 = Ii
        Ii = ((Ia2+Ib2)/2.0)
        k = pred(Ii)
        delta = max_diff(Ia2,Ii)
    return Ii
```

Fig. 5.1: Boundary Estimation code snippet

   This code implements the boundary estimation algorithm (Fig. 5.1) used by the RED-Attack. The algorithm works by iteratively finding the midpoint between the source image and the target image until the midpoint is on the classification boundary. The classification boundary is the set of all images that are predicted to belong to the target class with a probability of 50

2. Adversarial Noise Optimization:

- An optimization algorithm introduces small perturbations in randomly selected pixels of the estimated sample.
- To ensure imperceptibility, the algorithm optimizes the distance between the perturbed sample and the target sample.

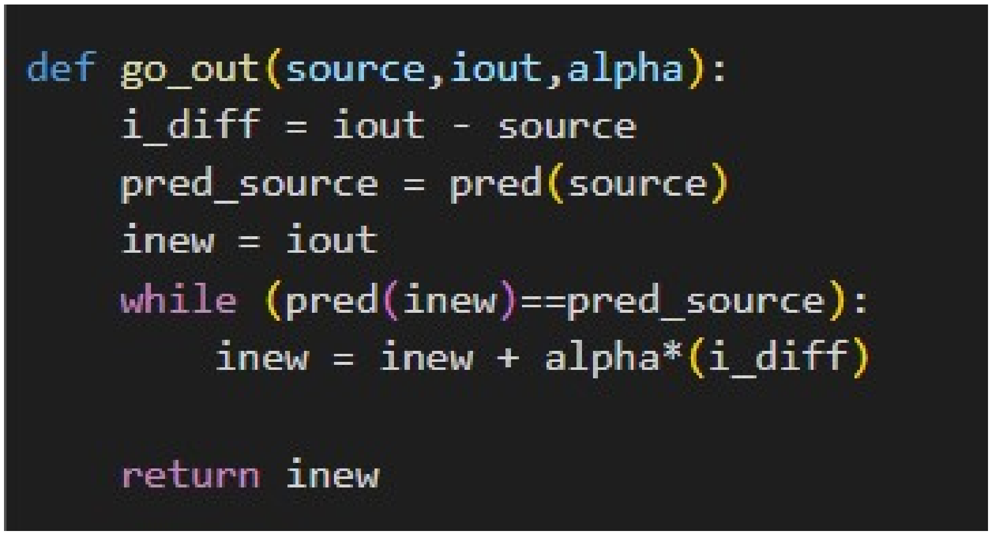


```
def iteration(itr, source, target, n, theta, j, dmin):
    targett = target
    sourcee = source
    for i in range(itr):
        print ("\n Iteration: ",i)
        adversarial_image = boundary_estimation(sourcee, targett, dmin)
        adversarial_image = go_out(sourcee,adversarial_image,0.01)
        (g, Iii2) = gradient_estimation(sourcee, targett, targett, n, theta)
        targett = efficient_update(sourcee, targett, adversarial_image, Iii2, g, j)
        if (pred(targett) == pred(source)):
            j = j/2.0
        fin = targett
        if(pred(targett)==pred(sourcee)):
            fin = go_out(sourcee,targett,0.01)
        if(array_diff(fin-sourcee)<array_diff(adversarial_image-sourcee)):
            targett = fin

    return fin
```

Fig. 5.2: Iteration code snippet

The iteration function (Fig. 5.2) in RED-Attack iteratively refines the target image to be an adversarial image, which is indistinguishable from the original image but classified differently by the model.

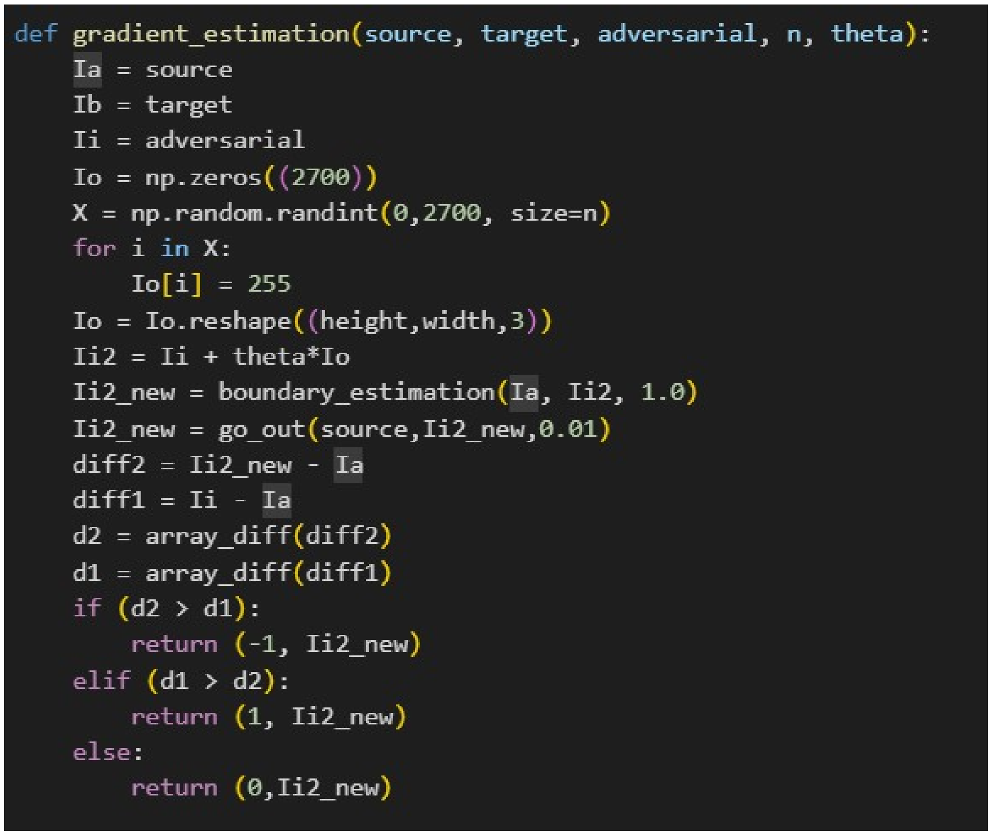
Estimates the classification boundary: It uses the boundary_estimation function (Fig. 5.1) to find an image on the boundary between the source and target class.



```python
def go_out(source,iout,alpha):
    i_diff = iout - source
    pred_source = pred(source)
    inew = iout
    while (pred(inew)==pred_source):
        inew = inew + alpha*(i_diff)

    return inew
```

Fig. 5.3: Go Out code snippet

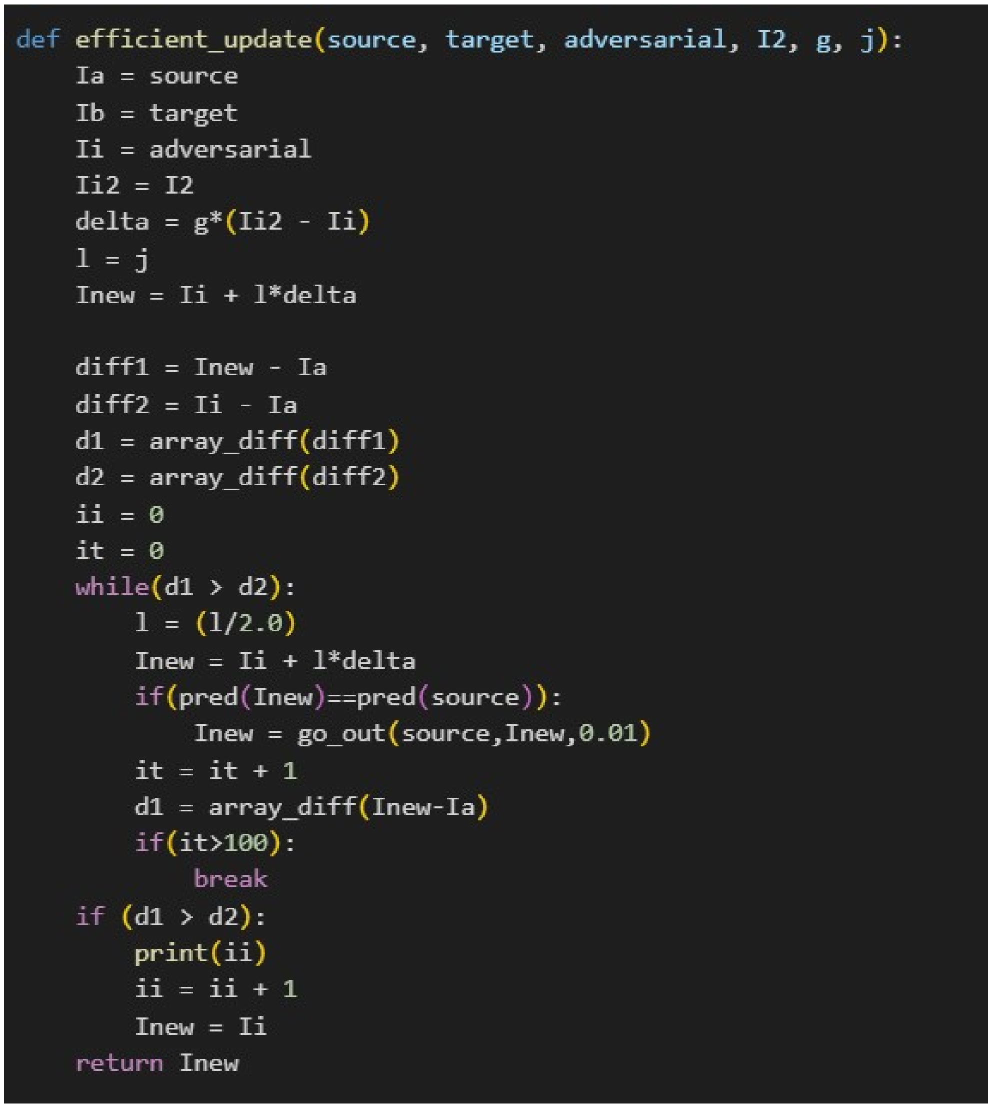
Ensures imperceptibility: It uses the go_out function (Fig. 5.3) to slightly perturb the image and ensure it remains virtually identical to the original.



```python
def gradient_estimation(source, target, adversarial, n, theta):
    Ia = source
    Ib = target
    Ii = adversarial
    Io = np.zeros((2700))
    X = np.random.randint(0,2700, size=n)
    for i in X:
        Io[i] = 255
    Io = Io.reshape((height,width,3))
    Ii2 = Ii + theta*Io
    Ii2_new = boundary_estimation(Ia, Ii2, 1.0)
    Ii2_new = go_out(source,Ii2_new,0.01)
    diff2 = Ii2_new - Ia
    diff1 = Ii - Ia
    d2 = array_diff(diff2)
    d1 = array_diff(diff1)
    if (d2 > d1):
        return (-1, Ii2_new)
    elif (d1 > d2):
        return (1, Ii2_new)
    else:
        return (0,Ii2_new)
```

Fig. 5.4: Gradient Estimation code snippet

Estimates the gradient: It uses the gradient_estimation function (Fig. 5.4) to approximate the direction in which the image should be moved to further fool the model.



```python
def efficient_update(source, target, adversarial, I2, g, j):
    Ia = source
    Ib = target
    Ii = adversarial
    Ii2 = I2
    delta = g*(Ii2 - Ii)
    l = j
    Inew = Ii + l*delta

    diff1 = Inew - Ia
    diff2 = Ii - Ia
    d1 = array_diff(diff1)
    d2 = array_diff(diff2)
    ii = 0
    it = 0
    while(d1 > d2):
        l = (l/2.0)
        Inew = Ii + l*delta
        if(pred(Inew)==pred(source)):
            Inew = go_out(source,Inew,0.01)
        it = it + 1
        d1 = array_diff(Inew-Ia)
        if(it>100):
            break
    if (d1 > d2):
        print(ii)
        ii = ii + 1
        Inew = Ii
    return Inew
```

Fig. 5.5: Efficient Update code snippet

Updates the target image: It uses the efficient_update function (Fig. 5.5) to update the target image towards the adversarial direction using gradient descent.

Checks for success: It compares the final target image to both the source and initial adversarial image. If it's closer to the source but still classified differently, it's considered a successful adversarial image.

The complete code is available here.

5.4 Results and Analysis

Below Fig. (Fig. 5.6) and (Fig. 5.7), the visual representation includes the source image, the perturbed image resulting from the RED attack, and the target image, each accompanied by the corresponding model output for GTSRB and Xray-Pneumonia Dataset.

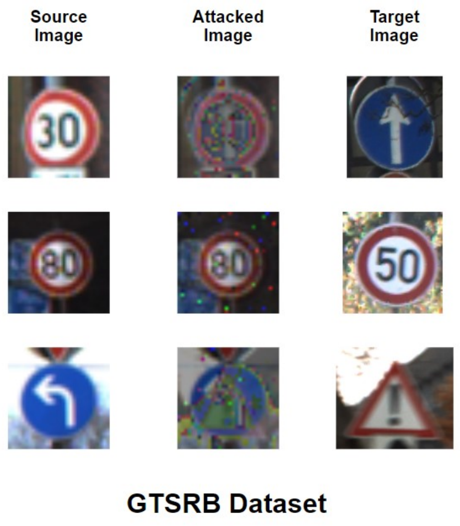


Fig. 5.6: GTSRB Dataset

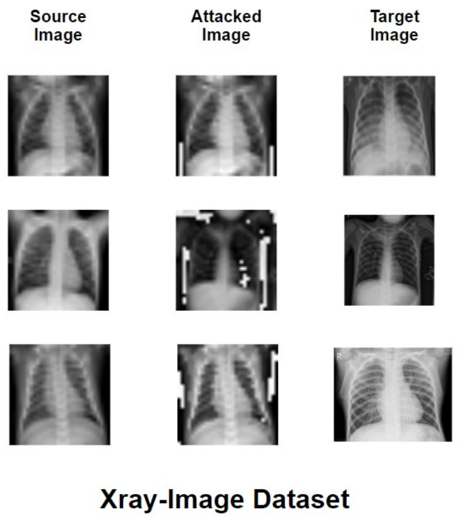


Fig. 5.7: Xray Pneumonia Dataset

Upon analysis, it was observed that the Red Attack effectively operates on images, generating adversarial images that successfully deceive the model.

Chapter 6

Conclusion and Future Work

6.1 Summary of key findings

In conclusion, this research has provided valuable insights into the robustness of the ResNet-18 model against various adversarial attacks. Our analysis, supported by quantitative results and qualitative GradCAM visualizations, sheds light on the model's performance under different attack scenarios. We observed for epsilon greater than 0.005, attacks performed well in attacking RestNet-18 model.

The Red Attack proves to be a potent strategy in the realm of adversarial machine learning. Through its resource-efficient decision-based methodology, it effectively generates imperceptible adversarial images, demonstrating the vulnerability of models under threat model.

6.2 Contributions to the field of Adversarial Machine Learning

These findings contribute to our understanding of adversarial machine learning and highlight the importance of security aspects.

6.3 Implications for future research

With a comprehensive understanding of diverse adversarial attacks and their real-world implications, coupled with the practical implementation of the Resource-Efficient Decision-based Imperceptible Attack for Machine Learning [8], there emerges an opportunity to delve deeper into similar attack methodologies. This section delineates the trajectory and methodology for future investigations, building upon the acquired knowledge to further explore attacks that assess the robustness of machine learning models.

References

[1] Shilin Qiu, Qihe Liu, Shijie Zhou, and Chunjiang Wu, “Review of artificial intelligence adversarial attack and defense technologies,” 2019. [Online]. Available: https://www.mdpi.com/2076-3417/9/5/909

[2] Wieland Brendel, Jonas Rauber, and Matthias Bethge, “Decision-based adversarial attacks: Reliable attacks against black-box machine learning models,” 2018. [Online]. Available: https://arxiv.org/abs/1712.04248

[3] Moustafa Alzantot, Yash Sharma, Supriyo Chakraborty, Huan Zhang, Cho-Jui Hsieh, and Mani Srivastava, “Genattack: Practical black-box attacks with gradient-free optimization,” 2019. [Online]. Available: https://arxiv.org/abs/1805.11090

[4] Quanxin Zhang, Kunqing Wang, Wenjiao Zhang, and Jingjing Hu, “Attacking black-box image classifiers with particle swarm optimization,” vol. 7, 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8876844

[5] Nicholas Carlini and David Wagner, “Towards evaluating the robustness of neural networks,” 2017. [Online]. Available: https://arxiv.org/abs/1608.04644

[6] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy, “Explaining and harnessing adversarial examples,” 2015. [Online]. Available: https://arxiv.org/abs/1412.6572

[7] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami, “The limitations of deep learning in adversarial settings,” 2015. [Online]. Available: https://arxiv.org/abs/1511.07528

[8] Faiq Khalid, Hassan Ali, Muhammad Abdullah Hanif, Semeen Rehman, Rehan Ahmed, and Muhammad Shafique, “Red-attack: Resource efficient decision-based imperceptible attack for machine learning,” vol. 2, 2019. [Online]. Available: https://arxiv.org/pdf/1901.10258.pdf