

A Phase - II Project Report on

# Prevention of the Adversarial Attacks on Neural Networks

Submitted in partial fulfillment of the requirement for the degree of

Master of Technology

In

Computer Science and Engineering

Submitted by:

#### Pratibha M Goudar

#### [01FE20MCS010]

Under the guidance of

#### Dr. Meena S M

#### Prof. Sunil V. Gurlahosur

SCHOOL OF COMPUTER SCIENCE & ENGINEERING

KLE TECHNOLOGICAL UNIVERSITY,

HUBBALLI – 580031

2021-22

2021 - 2022



SCHOOL OF COMPUTER SCIENCE &ENGINEERING



CERTIFICATE

This is to certify that a Phase-II Project titled **“Prevention of the Adversarial Attacks on Neural Network”** is a bonafide work carried out by the student comprising of **PRATIBHA M GOUDAR 01FE20MCS010** for partial fulfillment of completion of third-semester M.Tech in Computer Science and Engineering during the academic year 2021-22.

Guide P. G. Coordinator Head, SoCSE

Dr. Meena S M. Dr. Vishwanath. P. Baligar Dr. Meena S. M

Prof. Sunil V.Gurlahosur

External Viva -Voce:

Name of the Examiners Signature with date 1.

2.

**ACKNOWLEDGEMENT**

The successful completion of any task is incomplete without complementing those who made it possible and whose guidance and encouragement made my effort successful. Directions were mainly responsible for my successful completion of the project.

I take this opportunity to thank our Vice-Chancellor Dr. Ashok S. Shettar and Registrar Dr. N. H. Ayachit for providing a healthy environment in the college, which helped in the concentrate on the task.

I also take this opportunity to express my deep gratitude and sincere thanks to the Head, School of Computer Science & Engineering, and my beloved Guide Dr. Meena S. M.

And Sunil V. Gurlahosur there a tremendous source of inspiration and valuable help in challenging my effort in the right direction. Who was a constant source of enthusiasm and whose sincere guidance, valuable suggestions, and benevolent direction were mainly responsible for my successful completion of the project.

I am indebted to our beloved P. G. Coordinator Dr. Vishwanath. P. Baligar is a constant source of enthusiasm and whose sincere guidance, valuable suggestions, and benevolent direction were mainly responsible for my successful completion of the project.

Finally yet importantly I will take the pleasure to thank all the teaching and non-teaching staff of our college and also my student friends.

I also thank K.L.E Technological University for providing computational support.

PRATIBHA M GOUDAR

01FE20MCS010

## ABSTRACT

The state-of-art of deep neural networks has highly trained magnificent results on several image classifications. However, adversarial attacks may effectively mislead deep neural networks by introducing minor disturbance to the input images. One of the most effective ways to protect the model against hostile cases is adversarial training. This process includes training the model with adversarial examples to improve its robustness of the model. In this project, Torchattacks and PyTorch library helps import the adversarial examples and are used to identify which attack was strongest in the torch document. In addition, the model has been trained and strengthened against adversarial instances using the strongest attack. The dataset used to perform the project was CIFAR10. The dataset was tested using a Projected Gradient Descent attack.

Keywords: Adversarial machine learning, Adversarial Training, Deep Neural Network

|  |  |  |
| --- | --- | --- |
| CHAPTERS | CONTENTS  **TABLE OF CONTENTS** | PAGE.NO |
|  | ACKNOWLEDGEMENT | i |
|  | ABSTRACT | ii |
|  | LIST OF FIGURES | v |
| 1 | [Introduction](#_bookmark6) | 6 |
| 2 | Problem Statement and Objective  2.1 Problem Statement  2.2 Objectives  2.3 Motivation | 11  11  11  11 |
| 3 | Literature Survey | 12 |
| 4 | Requirement Specification  4.1 Functional Requirements  4.2 Non-Functional Requirements  4.3 Hardware Requirements  4.4 Software Requirements | 16  16  16  17  17 |
| 5 | Proposed System  5.1 Description of the proposed system | 19  19 |
| 6 | System Design  6.1 Architecture of the system | 21  21 |
| 7 | Implementation  7.1 Proposed methodology  7.2 Dataset Description  7.3 Evaluation metrics | 31  31  34  34 |
| 8 | Results and discussions  8.1 Model Architecture  8.2 Testing | 36  36    44 |
| 9 | Conclusion and Future work | 45 |
|  | References | 48 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# 

**LIST OF FIGURES**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl.No** | **Figure** | **Figure Name** | **Page. No** |
| 1 | 1 | Adversarial example generation and adversarial attack | 6 |
| 2 | 2 | Different types of adversarial attacks | 7 |
| 3 | 3 | The adversarial examples generation process | 19 |
| 4 | 4 | High-Level design | 22 |
| 5 | 5 | Detailed Design | 23 |
| 6 | 6 | AlexNet architecture | 32 |
| 7 | 7 | CIFAR10 Dataset | 34 |
| 8 | 8 | AlexNet Architecture for Benign Training | 37 |
| 9 | 9 | AlexNet Architecture for pgd adversarial training | 39 |
| 10 | 10 | AlexNet Benign Training accuracy | 40 |
| 11 | 11 | AlexNet Benign Test Accuracy | 40 |
| 12 | 12 | AlexNet Adversarial Training Accuracy | 41 |
| 13 | 13 | Benign Train Accuracy | 42 |
| 14 | 14 | Benign Test Accuracy | 42 |
| 15    16 | 15  16 | Adversarial Train accuracy    Adversarial Test Accuracy | 43  43 |

17 17 Sample test image 44

**Chapter 1**

**Introduction**

A number of computer vision tasks were performed with state-of-the-art performance using deep neural networks (DNNs), natural language processing, life sciences, reinforcement learning, and many other domains. Given the fast advancement of artificial intelligence (AI), machine learning (ML), and deep learning (DL) approaches, ensuring the security and resilience of the algorithms is exceedingly difficult. Adversarial machine learning [2] is a technique is use in machine learning (ML) to trick or mislead the model with malicious input data. One way to produce adversarial instances is through an attack. And although they mimic human input, these adversarial instances are used as input to the models to cause a model to mispredict the future. Additionally, a model that is highly confident in making incorrect predictions. Its nature refers to it as an antagonistic sample. Machine learning algorithms, also known as models, mimic human nature and forecast a certain input. One of the well-liked frameworks for training a standard model is Pytorch [3]. When a model is trained, it goes through many iterations known as epochs. The assaults that trick a model by adding noise that humans cannot hear are instances of adversarial behaviour.

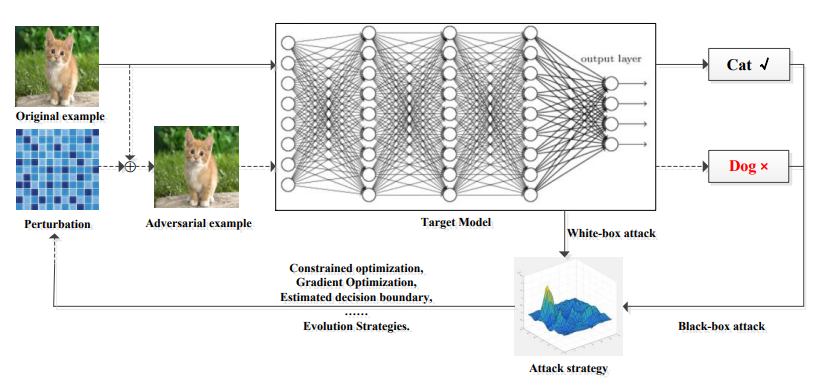


Figure 1: Adversarial example generation and adversarial attack process

The primary goal of an adversarial assault is to disturb an image to cause the target model to incorrectly categorize samples, even though the three major types of research on adversarial attacks have been widely used in this field. Deep learning models for image categorization are increasingly used in our systems today, either to replace human operators and automate some tedious jobs or to construct better applications. Their rising use is a result of their excellent accuracy, which has led to newer models being able to perform better than the human brain in many item categorization tasks. Deep neural networks are still susceptible to hostile samples despite their strong generalization, though. Therefore, a lot of work has been done in recent years to uncover the flaws in deep learning models and how to use them to launch adversarial assaults. Meanwhile, fresh defense strategies have been created to counteract those intrusions and increase the dependability of deep learning systems in practical applications.

Adversarial attacks

White-box attacks

Black-box attacks

Transfer based attacks

Score based attacks

Decision-based attacks

Figure 2: Different types of adversarial attacks

Since the introduction of deep convolutional networks used for image classification, the computer vision field has undergone numerous significant breakthroughs, such as Inception, DenseNet, and ResNet, to further improve the classification accuracy obtained on many image classification tasks. Though very accurate, the models lack resilience since they are readily fooled by adversarial cases. More specifically, an adversarial example is a sample of data that has been very slightly manipulated to trick a machine learning classifier.

Therefore, an adversarial assault is producing these hostile samples and feeding them to the target model to Influence its prediction. Thus, the use of deep learning models in actual systems may be jeopardized by these attacks, which pose a severe hazard. An adversarial example, such as one found on a traffic sign, can drive an autonomous vehicle to make a poor maneuver that could result in an accident or another disastrous circumstance. Comparatively, adversarial instances on human faces might deceitfully trick face recognition algorithms, allowing impersonation or identity evasion, rendering those systems untrustworthy. Contrarily, the adversarial defense tries to defend deep learning models from adversarial attacks, namely by making models more resilient against hostile cases, since deep convolutional networks have been utilized for image classification.

In this regard, research on adversarial robustness resembles a minimax game in which attackers continuously strive to use more potent tactics to trick deep learning models while, at the same time, defenders have to develop new defense mechanisms to ward off these malicious attacks. To give researchers future guidance on how to maintain a balance between attack and defense in adversarial learning, the goal of this post is to evaluate how attack and defense approaches have changed over time. The attributes of threat models, or a collection of presumptions about the adversary's objectives, knowledge, and capabilities, will be explained in the sections that follow. Threat models define those circumstances to specify the circumstances under which a defense is intended to be secure.

* **Adversarial Capability:**

Only little adjustments may be made to the original input picture x (legal example) to create an adversarial example since it must be misclassified by deep learning models but not by human brains. Adversarial example xadv=x+δ, where δ is also known as **adversarial noise**. The distance introduced by the noise between x and xadv is usually defined by the lp norm of the gap between the original and the adversarial sample for some p=0 to ∞. Thus an adversarial example xadv has to satisfy the constraint ||xadv–x||p<=Ɛ **(adversarial constraint)**, where smaller Ɛ values correspond to smaller input perturbations, thus leading to less perceptible changes under the condition that xadv is misclassified. For instance, the l0 distance corresponds to the number of pixels that have been changed in a picture, the l2 distance measures the normal Euclidean (root-mean-square) distance between x and xadv, and the l distance evaluates the largest change to any of the coordinates.

* **Adversarial goal:**

In reality, adversarial assaults can be created to accomplish one of two objectives:

**Untargeted:** Attacks without a specific target class are intended to cause the model to forecast any incorrect class. Formally, an adversarial case would result in f (xadv)! = y has given a classifier f and the true label y. Therefore, under the premise||xadv-x||p<= €, the objective of untargeted assaults is to maximize the loss of the attacked model relative to the true label y, or max L(xadv, y). These attacks are typically simpler to execute because of the permissive limits placed on the adversarial samples.

**Targeted**: In contrast, targeted assaults aim to skew the model's prediction in favor of a particular class y', i.e., f (xadv) = y', where y'! = y. Once this is done, the loss function can be written as min L (xadv, y') always under adversarial constraints. This will increase the likelihood that the attacked model would predict the target class y' rather than any other class given the input xadv.

* **Adversarial Knowledge:**

Depending on the amount of knowledge about the model being attacked, adversarial assaults may be split into two primary categories.

**White box attacks**: In this scenario, the adversary has complete access to and knowledge of the model, including its architecture, parameters, gradients, and loss of input, as well as any potential defenses. Thus, under this circumstance, it is not very challenging to attack models, and standard techniques make use of the output gradient of the model to provide adversarial samples.

**Black box attacks:** The adversary in this class of attacks has no or very little understanding of the model. As a result, training a single identical model or an ensemble of them is frequently used in existing approaches. These techniques are effective because, in general, adversarial instances that deceive one model are likely to deceive another model of a similar type. Because attackers typically lack access to a significant portion of the models' information, this type of assault is also the most likely to occur in real-world scenarios.

To create hostile instances, some techniques use information about the prediction's accuracy or only the label. The adversary in this class of attacks has no or very little understanding of the model. As a result, training a single identical model or an ensemble of them is frequently used in existing approaches. These techniques are effective because, in general, adversarial instances that deceive one model are likely to deceive another model of a similar type. Because attackers typically lack access to a significant portion of the models' information, this type of assault is also the most likely to occur in real-world scenarios. To create hostile instances, some techniques use information about the prediction's accuracy or only the label. The adversary in this class of attacks has no or very little understanding of the model.

As a result, training a single identical model or an ensemble of them is frequently used in existing approaches. These techniques are effective because, in general, adversarial instances that deceive one model are likely to deceive another model of a similar type. Because attackers typically lack access to a significant portion of the models' information, this type of assault is also the most likely to occur in real-world scenarios. To create hostile instances, some techniques use information about the prediction accuracy or only the label.

**Chapter 2**

**Problem Statement and Objectives**

**2.1 Problem statement**

To build the neural network architecture for classifying the images and experimenting with adversarial attacks to evaluate the accuracy and improve the architecture performance under the adversarial attacks.

**2.2 Objectives**

* To find strong adversarial examples from Torch Attacks and see, which one is best, suited for adversarial training.
* To test the strongest attack against the CIFAR-10 datasets.
* To make the model more robust.
  1. **Motivation**

Currently, Building an efficient and robust neural network is time-consuming, error-prone computationally expensive; there is a need to democratize the development of neural networks for it is wider use. And adversarial machine learning techniques are the best way to secure models from adversaries. And we also need to ensure that such networks work on embedded devices, which are resource constrained.

**Chapter 3**

**Literature Survey**

Due to the rapid advancement of deep learning (DL) and artificial intelligence (AI) approaches, it is essential to guarantee the security and resilience of the implemented algorithms. Existing adversarial assaults may be separated into white-box and black-box assaults, following the threat model [1][8]. The model system for white-box assaults assumes that the adversaries are fully aware about the target image, including model design and parameters. The output of the target model is the only observational tool available to the adversaries in the dark threat model. Several attack methods, include distributionally adversarial attack, projected gradient descent, fast gradient sign method (FGSM), basic iterative method (BIM),etc., have been presented in the frames of these threat models.[2]Using an adversarial attack is a method to identify.

Torch assaults allow MultiAttack for combining numerous adversarial attacks, even if the disturbance is extremely tiny and invisible to the human eye. A more potent adversarial example may be constructed by utilizing MultiAttack [2]. Presented a set of quick techniques for producing adversarial examples. And these may be accounted for as a high-dimensional dot product's characteristic [3]. Shown that adversarial training can lead to much more regularization than dropout can. Consequently, it is simple to perturb models that are straightforward to optimize [3]. It may be classified as an optimization-based network attack or a forward derivative-based assault depending on the techniques used to create adversarial instances [4]. Defense strategies may also be divided into proactive defense and reactive defense. [4]Proactive defense trains the model to be resistant to hostile cases, or defensive distillation.

[7] The reactive defensive technique employs a second model to find hostile cases. e.g., the safety net is resistant to adversarial examples [8] developed using well-known attacker strategies. [6] This is possible to exclude hostile cases originating from attack strategies not present in training data. Defensive-GAN is a cutting-edge defense technique that uses Using GANs, classification models are made more resistant to adversarial attacks in both the black-box and white-box scenarios.[9]DL has been used widely in many different application sectors, including voice recognition and medical diagnostics [16].

In this study, we investigated the potential for building adversarial instances for real-world machine learning systems. An Inception v3 image categorization neural network [17] was fed with photos captured with a mobile phone camera. We demonstrated that in such a setup, even when hostile pictures created using the original network are supplied to the classifier through the camera, a sizable portion of them are misclassified. This finding demonstrates the existence of hostile situations for machine learning systems in the actual world. Future work should make it possible to illustrate attacks using actual objects rather than just pictures written on paper. against other machine learning systems, such as sophisticated reinforcement learning agents, attacks undertaken by human attackers, and so forth. Extreme care and skepticism of all produced outcomes are required while evaluating adversarial example defenses. When conducting assessments, researchers must exercise extreme caution to avoid unwittingly deluding themselves. This paper explains the purpose of conducting defense evaluations and our rationale for thinking that this is the case.

The researcher create a set of suggestions based on the typical errors we found in assessments of adversarial sample defenses. This checklist is intended to help readers and reviewers determine if an evaluation is complete and adheres to current best practices as well as researchers[18] creating innovative defense strategies Overall, we strongly feel that building strong machine learning models is important and that this paper will in some way assist the larger community in reaching this crucial objective. The applications of deep learning are restricted by the presence of adversarial instances. To build defenses that can withstand hostile examples is an open challenge. Defensive distillation was suggested as a general-purpose method to boost an arbitrary neural network's resilience to address this issue.

The assaults we suggest in this study are strong enough to overcome defensive distillation, proving that they may be used to assess the effectiveness of proposed defenses more generally. We choose one attack strategy after thoroughly analyzing a large number of potential attack strategies that can consistently find better adversarial instances than all other strategies. Our three L0, L2, and L∞ assaults are built from the results of this evaluation. We urge those who develop defenses to put them into action. The methods of evaluation we employ in this study are as follows: Directly assess the resilience of the secured model using a potent attack [19].

Defenders must ensure that they are resilient against the L2 distance metric since a defense that stops our L2 assault will also stop our other attacks. Construct high-confidence adversarial cases on an unsecured model and illustrate how they fail to transfer to the secured model to prove that transferability fails.In this study, we introduced an algorithm called Deep Fool to generate adversarial cases that can deceive cutting-edge classifiers. It is based on an iterative linearization of the classifier that produces the smallest perturbations necessary to change classification labels. In-depth experimental evidence was presented on three datasets and eight classifiers to show the usefulness of the suggested approach as well as its superiority to contemporary methods for computing adversarial perturbations. Because it accurately assesses the adversarial perturbations, the suggested Deep Fool approach[20] offers a mechanism for evaluating the robustness of classifiers and for improving their performance through suitable fine-tuning.

To precisely estimate the minimum perturbation vectors and create more trustworthy classifiers, the proposed method may be applied. The mainstay of contemporary AI is gradient-based optimization. We can fit the majority of the issues we care about, at least on the training set, using a network that has been properly constructed to be linear, whether it is a ReLU or max network, an LSTM, or a sigmoid network. The existence of hostile instances implies that just because our models can interpret the training data or even properly classify the test data does not mean that they fully comprehend the tasks we have given them to do. Instead, their linear replies are unduly certain at places that do not appear in the data distribution,[21] and these certain forecasts are frequently wildly wrong. To locate adversarial instances rapidly and with little perceptual changes, we presented Non-Targeted JSMA and Maximal JSMA as more flexible variations of the Jacobian-based Saliency Map Attack [22]. Most significantly, MJSMA eliminates the requirement to identify the target class and the direction of the perturbation. We empirically demonstrated that M-JSMA regularly discovered high-quality competitors in a range of picture datasets. Through this study, we seek to increase awareness of the simplicity of creating adversarial situations and to improve our knowledge of assaults and how to counter them.

In this research, the researcher offers a large family of momentum-based iterative approaches to strengthen adversarial assaults. These methods can successfully deceive both black-box and white-box models. In the black-box setting, our techniques routinely outperform one-step gradient-based techniques and conventional iterative techniques. We carry out thorough trials to verify the efficacy of the suggested solutions and illuminate the rationale behind their practical success. We suggest attacking an ensemble of models whose logits are merged to further increase the transferability of the generated adversarial cases[23]. We demonstrate the susceptibility of the models produced through ensemble adversarial training to our black-box assaults, posing additional security concerns for the creation of more resistant deep learning models.Researchers' findings show that deep neural networks may be strengthened to withstand hostile attacks. We can create trustworthy adversarial training techniques, as our theory and results show. The surprising regularity of the underlying optimization task, this is equivalent to maximising a very non-concave function with numerous recognised local maxima [24], is one of the key insights behind this. Their values are highly concentrated despite the fact that the pertinent issue involves the maximising of an actually non-function with such a lot of local maxima. Overall, our findings encourage that adversarial resilient deep learning models may already be within reach.

The MNIST dataset shows that networks are highly robust, attaining great accuracy for a range of powerful constrained adversaries and considerable disturbances.CIFAR10 trials have not yet been performed at the same level. But as indicated by the findings. Building defenses to hostile instances includes protecting against both current assaults and potential future attacks. In this research, we find obfuscated gradients, a phenomenon displayed by some defenses that prevent the generation of adversarial cases using conventional gradient-based approaches. To get around three distinct kinds of obfuscated gradients, we create three attack strategies Further, utilize the ICLR 2018 defenses as a case study to assess the applicability of our methods, and we can get around seven of the nine legitimate arguments. More generally, we believe that future research will be able to use our evaluation method to identify possible risks rather of relying just on obfuscated gradients (or other methods that exclusively defend against assaults based on gradient descent).When it happens. Responding to contradictory examples is a crucial component of a significant field of study, and we think that while constructing defenses, completing a meticulous review is an essential stage that must not be skipped

**Chapter 4**

**Software Requirement Specification**

Software Requirements Specification Some of the software and hardware requirements are stated in the specifications, which describe the essential hardware and software elements required for the system to function properly.

**4.1 Functional Requirements:**

Functional requirements for the application are as follows:

* + - The Proposed models should detect and classify the image under adversarial attack correctly.
    - The system shall be able to provide a facility to classify images correctly

**4.2 Non-Functional Requirements:**

Non-Functional requirements for the application are as follows:

* Compatibility: Any system that has the necessary setup should be able to run the programme.
* Availability: The programme should always be accessible..
* Performance: A performance-driven application is required.
* Efficiency: After the model has been trained, the software should have acceptable test accuracy.
  1. **Hardware Requirements**
* Nvidia-CUDA (for GPU parallelization).
* CuDNN (Cuda Library for Deep Neural Networks).
* Intel Xeon Processor - 3.2 GHz.
* 32 GB Main Memory.
* Nvidia Quadro GPU - 8GB
  1. **Software Requirement**
* PyTorch
* Python
* Netron
* Google collab

**Chapter 5**

**Proposed System**

**5.1 Description of the proposed system**

The system includes the input dataset along with processed images and the target model. The architecture for the given dataset by using adversarial algorithms.

Start

Clean raw dataset

Processed images

Target model

Correct prediction

Adversarial Attack algorithm

Perturbed data

Correct

prediction

Adversarial examples

Yes

No

No

Yes

Figure 3: The adversarial examples generation process

Figure 3: demonstrates how it will provide adversarial instances that will forecast the model against adversarial assaults and make a model resilient. The general concept of adversarial examples can be formulated as an optimization problem in the context of deep learning, with the objective being to minimize the cost of maintaining the objective function that generates the adversarial examples while keeping it as close to a real input as is practical. Even while this contradicting instance is almost invisible to the human eye, it can deceive the deep learning system.

In the proposed system step 1 is considered as the start process, and first, we clean the raw dataset after cleaning the raw dataset processed images on the given dataset, again we predict the label of the images if the image label matches the same image will go for applying the adversarial attack on the particular image with some perturbation value i.e we call it as epsilon it may be 0.1, 0.2,0,3, it's small noise added to an image, if not the same label again go back to step 2 processed the image again, if it's correct we go for the next step as applying adversarial attack, after applying these attack the data will be perturbed.

After perturbing also it will predict the correct label of the image and then specify that the adversarial attack is weak, for this way continuously generates the adversarial examples.

These all examples try it on target model; this target model will be standard architecture from the deep learning i.e alexnet. Finally on this target model we train the images after getting accuracy and correct label, then our trained model the strong enough for the adversarial attacks, then model is robust and resistant towards any perturb images.

**Chapter 6**

**System Design**

**6.1 Architecture of the system**

Input data

Trained model

Apply Adversarial Attacks

Obtained accuracy



Figure 4: High-level design

Figure 4: depicts the high-level view of the entire process, initially the data will be pre-processed for better accuracy. Building the modified neural network architecture model and training the model on the CIFAR10 dataset to evaluate the accuracies of clean data, which is test data.

Level 1 considered the input data, it's referred to pre-processed dataset .pre-preprocessing includes removing noise from the data, normalizing and resizing data, and many more steps

according to our considered dataset, we can choose the pre-processing steps.

In level 2 consider the standard architecture like Alexnet, ResNet, Google Net, VGG16, and many more, but in our work, we have considered the alexnet architecture because we have done the literature survey as the study says alexnet will give more promising results compared to other models and it will include the more layers in the architecture, it will help to give more accurate trained accuracy.

In level 3 checking whether our selected standard architecture is robust or not, first we disturb the model by applying the adversarial attacks on this model, before applying the attacks we obtain some accuracy that is mentioned as benign accuracy, after performing the adversaries on this model if still our model gives moderate accuracy and it's predicted the correct test image. We can say that the model is strong enough and all these adversarial attacks and defense methods can be discussed in the detailed design of the next chapter. More prominently we get accuracy on the selected model and we compare benign and adversarial accuracy.

Input data

Trained model

Applying Adversarial Attacks

Obtained

Accuracy

Preprocessed data

Build AlexNet architecture

Train model on cifar10 dataset

Evaluate the accuracy of clean data

Evaluate the accuracy of adversarial attacks

Apply adversarial training as a defense method method

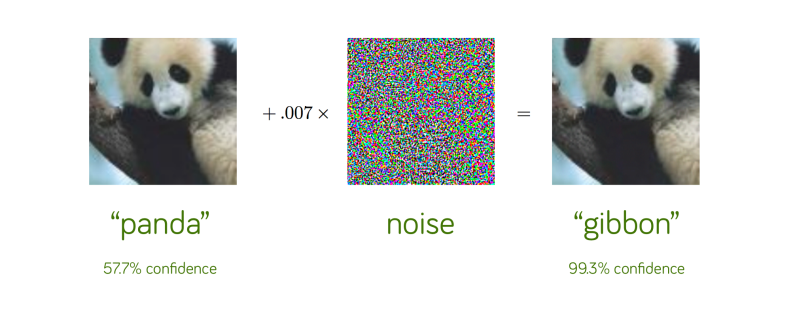
Compare the accuracy

Evaluate the accuracy of clean data vs. training data

Figure 5: Detailed Design

Figure 5: depicts the detailed design for the system. After pre-processing input data go for training the model in that case firstly we build standard architecture i.e alexnet architecture this architecture further we train it on the famous cifar10 dataset, after training this model on the cifar10 dataset we can evaluate the accuracy, after that applied strong adversarial attacks from PyTorch library and then check for the model is under adversarial attacks or not.

That PyTorch attacks are listed here



* **White box adversarial attacks on images**

Defining adversarial assaults in the context of photographs, we introduced the idea before. White-box assaults, the first type of attack, will be discussed here. In this scenario, the adversary has complete control over the model and full knowledge of its architecture, parameters, gradients, and loss of input, as well as any potential defense measures. Thus, it is not very challenging to attack models in this situation, and standard techniques make use of the output gradient of the model to provide adversarial samples. The following attacks will be covered in further detail:

* **The Fast Gradient Sign Method (FGSM)** :

The simple one-step gradient-based strategy that may identify an adversarial example in one step by maximising the loss function L(xadv, y) regarding the input x and then adding back the sign of the output gradient to (x) to obtain the adversarial example.

xadv ) (1)

Where xL(x, y) represents the gradient of the loss relative to input x, and the equation is anticipated to satisfy the l norm bound by design. This approach is effective because it Reduces the possibility that the correct label y may be anticipated from the valid input xadv by perturbing the input in a way that maximizes its loss concerning the correct label y. Through the use of mathematics, it advances the adversarial example on a certain path toward the boundary between the true class and another class.

* **FGM:**

The fundamental difference between the Fast Gradient Method (FGM) and the Generalized Fast Gradient Method (FGSM) is that the sign operator is not applied to the acquired gradient in the FGM. What makes it unique

(2)

In order to prevent the adversarial noise from breaking the constraint ||xadv-x||p=, it should be emphasised that the gradient in this case is divided by its lp norm. Even though the sign operator may be thought of as a normalisation process, FGSM is less successful than FGM since it also changes the direction that is most effective. All methods used to exploit FGSM may be used to also exploit FGM due to their similarities.

* **I-FGSM:**

Easy iterative methods Apply a quick gradient frequently, iteratively, and with a small step size. Consequently, the FGSM iterative form (I-FGSM) may be represented as

= (3)

And xadv0=x (legitimate example). There are several approaches to meet the norm bound for the adversarial case. As an illustration, xadv might be made to equal T, where T is the

the number of iterations, or trimmed within the area of x. It has been established that iterative approaches make use of considerably smaller perturbations, which confuse the classifier at higher rates while not destroying the picture even at higher. Iterative approaches have the disadvantage of being slightly slower than their one-step equivalents, but more crucially, they perform poorly on transferability, which is a crucial characteristic of black box assaults.

* **MI-FGSM**

The goal of momentum-based iterative gradient-based techniques is to produce adversarial cases that meet the lp norm constraint by integrating momentum into iterative fast gradient techniques. By building up a velocity vector in the direction of the loss function's gradient across several iterations, momentum is a method used to accelerate gradient descent algorithms.

This idea may, however, also be used to create antagonistic cases and gain significant advantages. Similar to how the learning rate update was done, the initial step in updating the momentum gt is adding up the velocity vector in the gradient direction as

(4)

a decay factor is where. Next, the adversarial example xadvt is disturbed with a step size as the sign of gt.

(5)

They found that the scale of the gradients varied in size over iterations, thus they decided to normalize the current gradient, xL(x,y), by the lp distance itself.

* **Deep Fool:**

An iterative approach called Deep Fool is used to quickly construct perturbations that can trick deep networks and hence accurately measure the resilience of these classifiers. At each iteration t, it roughly determines the separation between xadvt and the complement of the convex polyhedron P, which identifies the area of space where f(x) = y exists. The perturbation vector that crosses the polyhedron P's border is therefore determined and the current estimate is updated at each iteration of the procedure. The iterations continue until f(x)!=f(xadvt) to create an adversarial example with the fewest perturbations possible. As a result, Deep Fool is an algorithm that is greedy and may not always converge to the ideal perturbation. Though the technique produces minor perturbations in practice, the author noted that they are thought to be accurate estimates.

* **C&W**

The approach put out by Carlini and Wagner formalizes the issue of locating and is based on the original formulation of adversarial cases. The approach put out by Carlini & Wagner is based on the formal definition of adversarial instances and relies on the initial construction of adversarial examples.

For example, the adverse situation involving an I.M.A.

(7)

Assuming (x+,)[0,1]n. L(x+, y)!=y if and only if L(x+, y)=0, which makes the function f an objective function. By definition, an adversarial case is one where x+ is identical to xadv. As the least value of c for which the resultant solution xadv has L(xadv, y)=0, the constant c>0 is frequently selected. Furthermore, the adversarial noise is limited so that 0=xi+i=1 I to guarantee that the alteration produces a legitimate picture (this method assumes image pixels to be in the range [0,1]). To make the optimization issue in the definition above into an unconstrained minimization problem in w, x+ can be swapped out for ((1+tanh (w))/2) in one approach. As the last step, the Adam optimizer is used to iteratively reduce the loss.

* **PGD:**

PGD is a different iterative approach that uses projected gradient descent to create adversarial samples iteratively.

+ (6)

All allowable perturbations are contained in the set S. Projected gradient descent performs one-step of traditional gradient descent before clipping all of the coordinates to fall inside the box. In order to restart the process and look at a major chunk of the loss landscape, numerous points within the l-balls surrounding the data points collected from the evaluation set are employed. As a result of their extensive observations, the authors were able to draw the conclusion that all local maxima found by PGD have comparable loss values for both normally trained networks and hostile networks, highlighting the fact that robustness against the PGD adversary also yields reliability against all first-order adversaries, such as SGD-based attacks.

Further steps as a defense method consider adversarial training, after performing adversarial training we can obtain accuracy. But here listed some defense methods based on popularity. Deep learning models must be kept secure and their applications must be trustworthy given the diversity and potency of the current adversarial attack techniques. To guarantee that the model's performance is not compromised, researchers have defined specific design requirements for defenses against adversarial perturbations since the cure should not be worse than the illness.

* **Low impact on the architectur**e:

To prevent anomalous behavior, approaches should restrict the changes made to the architecture.

* **Maintain accuracy**:

The efficacy of the model's categorisation should not be compromised by precautions against unfriendly samples.

* **Maintain speed of network**:

At test time, the classifier's running time should not be greatly impacted by the solutions.

* **Defenses must be effective for hostile samples that are reasonably near to training dataset points:**

Since humans may easily find them, samples that are considerably removed from the training dataset are irrelevant in terms of security. As a result, many studies have been done in this area to develop reliable models that would protect them against hostile attacks. Similar to how assault tactics may be divided into several groups based on the approach they use, so can defense strategies. Then, defense strategies may be divided into five groups, including:

* **Robust training:**

These strategies seek to strengthen a classifier's resistance to minor internal disturbances. One approach is adversarial training, which is supplementing the training data with adversarial created instances, or defensive distillation, which entails reinstructing a network using previously produced soft labels. Training robust regularized models, such as those built on the Lipchitz constant, perturbation norm, or other regularization techniques, is another option.

* **Input transformation:**

These techniques make use of the fact that a few defenses attempt to change the inputs just before they are sent to the classifier. Among these are feature compression techniques like total variance reduction and picture quilting, as well as JPG compression. Through the use of generative models, more advanced techniques project adversarial instances onto the data distribution. A JPG compression of the input itself is one of the most basic and primitive techniques of input transformation defense against a targeted network.

However, JPG compression's performance declines quickly as the size of the perturbations rises, even for straightforward approaches like FGSM, and it is not always able to reverse the reduction in classification. Another method, known as feature squeezing, lowers the degrees of freedom that are accessible to an opponent by squeezing out extraneous information, which gives an adversary less opportunity to create hostile examples. When the model's predictions for the original sample and the compressed sample result in different outputs, the input is probably hostile. Reduced color pixel bit depth and spatial smoothing are examples of feature squeezing approaches. By minimizing the variance, adverse perturbations can be largely eliminated. It reconstructs the simplest image that is compatible with the limited collection of pixels it randomly chooses, using those pixels. Because these perturbations are often tiny and confined, the reconstructed picture does not include the adversarial perturbations.

* **Randomization:**

Unpredictability-based defense techniques reduce the impact of adversarial instances by introducing randomness to the model parameters or the input, for as by padding or shrinking the input.

* **Model ensemble:**

The application of ensemble techniques is not limited to defense. Some techniques average the predictions across random sounds introduced to the model rather than merely averaging the outputs from each model in the ensemble. Similar to this, a regularizer can be included to encourage variety in the predictions made by several models. Remember that these subcategories are not mutually exclusive and that a defense strategy may fall under more than one category.

* **Parseval regularization:**

It is a layer-wise regularization technique that carefully tunes the network's global Lipchitz constant to reduce the susceptibility to tiny perturbations. It functions by requiring each concealed layer's Lipchitz constant to be less than one.This prevents the Lipchitz constant from increasing exponentially, and the Lipchitz constant of the network as a whole is subsequently regulated using a standard regularization method (weight decay) at the last layer.

Networks trained using Parseval regularisation incorporate two ideas: maintaining orthonormal rows in linear and convolutional layers and performing convex combinations in aggregation layers to successfully enforce these requirements. Models train quicker and utilize their capacity more effectively as a result of this regularization technique.

* **Defensive distillation:**

DNN designs may be made smaller or use less computational resources by using a technique called distillation. A distilled network fd is trained to utilize the original network f's classification predictions. Since soft labels convey the relative differences between Classes, it is hypothesized that training with them would result in more knowledge than training with hard labels. To train robust classifiers, it has been suggested to use a technique called defensive distillation. Since f and fd share the same design, defensive distillation differs significantly from the distillation that was initially suggested. Defense-related distillation aims at robustness rather than compression, which justifies the distinction. The effectiveness of this defense strategy is further enhanced by the fact that gradients are significant.

* **Adversarial Training:**

Protects a model from hostile cases in a very straightforward and clear manner. It entails creating adversarial instances by employing various attack strategies against the target model. The target model is then retrained using the "augmented training set," which was created in the second stage by combining these hostile occurrences with the initial training set to generate a "augmented training set." Along with the goal function of the model, the adversarial objective function may also be lowered during the retraining phase and can act as a helpful regularizer.In the last step, we compare the accuracy of the clean dataset that is benign training and adversarial training on perturbed data. If the test image predicts the correct label after applying adversarial attacks then we can say that our model is robust.

**Chapter 7**

**Implementation**

**7.1 Proposed methodology**

First, we use to search for the appropriate architecture that is robust to adversarial attacks particular dataset i.e CIFAR10, we fine-tune the model architecture for the particular value of accuracy and the secondary objective as per requirements. This stage involves applying the suitable attacks and defense methods from the torch library and we train the model for a given dataset to get trained weights [12], we delivered this model with alexnet architecture and trained weights.

The steps required for the implementation:

Step1: Collecting the dataset from the Kaggle.

Step2: involves data preprocessing.

Step3: Train the model on the cifar10 dataset and obtain the accuracy.

Step4: Again test the model with the strongest adversarial attack from the torch library.

Step5: Again retrain the model with the attack i.e projected gradient descent and obtain the accuracy that is called adversarial training, this is one of the defense methods

Step6: Compare both benign accuracy and adversarial accuracy.

Step7: Define the model robustness.

The AlexNet architecture has a few more layers and is much bigger than the LeNet-5 architecture. The AlexNet architecture is made up of five convolutional layers and three fully connected layers. The AlexNet was designed for the classification of 50,000 high-resolution images of 10 classes. They used a rectified linear unit (ReLU) after the convolutional and fully connected layers, which helped their model train.

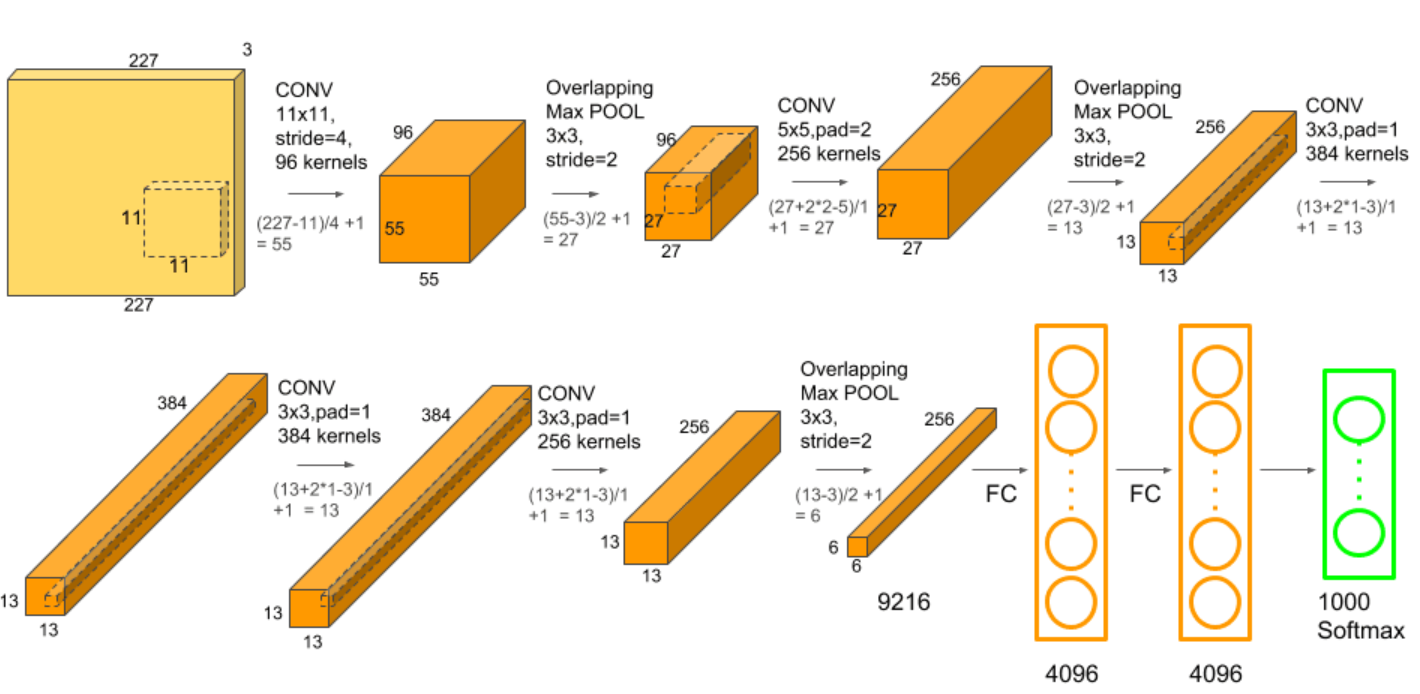
****

Figure 6: AlexNet Architecture

There are three fully connected layers and five convolutional layers in total in AlexNet. An overlapping Max Pooling layer comes after each of the first two convolutional layers. Filtering the input picture takes place in the first layer of AlexNet. The input picture has to be 227\*227\*3, where D = 3 stands for red, green, and blue. It also needs to have a width (W), height (H), and depth (D). Stride is the name of the first convolutional layer that was used to filter the input colour picture. It includes numerous kernels (K) equal 96 and a filter (F) that is 11x11 in size in addition to 4 pixels (s). In the kernel map, stride refers to the separation between responding field centres of nearby neurons. The third, fourth and fifth convolution

Layers are joined directly. The Max Pooling Layer, which overlaps the fifth convolutional layer and is subsequently connected to fully connected layers, comes next. For the imagenet dataset, which has 1000 classes, and the cifar10 dataset, which has 10 classes, the fully connected layers each have 4096 neurons. The second fully connected layer is input into the softmax classifier.

**7.2 Dataset Description: CIFAR10**

Machine learning and computer vision algorithms are frequently trained using the CIFAR-10 dataset [13], which is a collection of photographs. It has 60,000 32\*32-color pictures divided into ten distinct groups. 6,000 pictures from each class. i.e. Birds, cats, deer, dogs, frogs, horses, ships, trucks, aero planes, and vehicles are just a few examples. The low-resolution (32x32) images in CIFAR-10 provide researchers with the opportunity to swiftly test out various algorithms to see which ones are most effective. Photos are captured under various lighting circumstances and from various perspectives, and because they are coloured images, there are numerous differences in the hue of comparable items. In turn, accuracy improves.

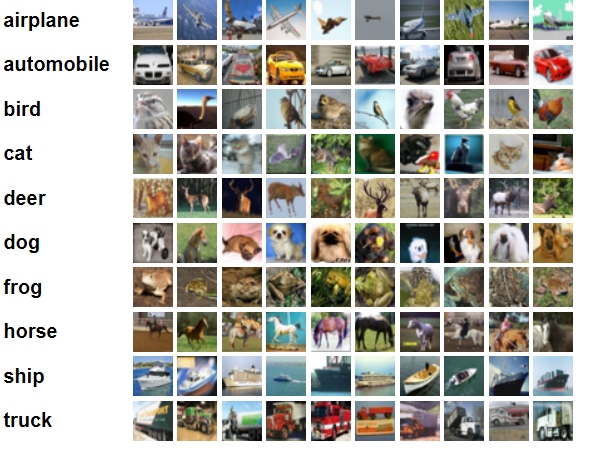
****

Figure 7: CIFAR10 Dataset Description

**7.3 Evaluation metrics:**

The model's overall performance was assessed using a number of assessment indicators, including: Accuracy Using the following mathematical equations, the weighted average for recall and precision was assessed. The confusion matrix has four key terms:

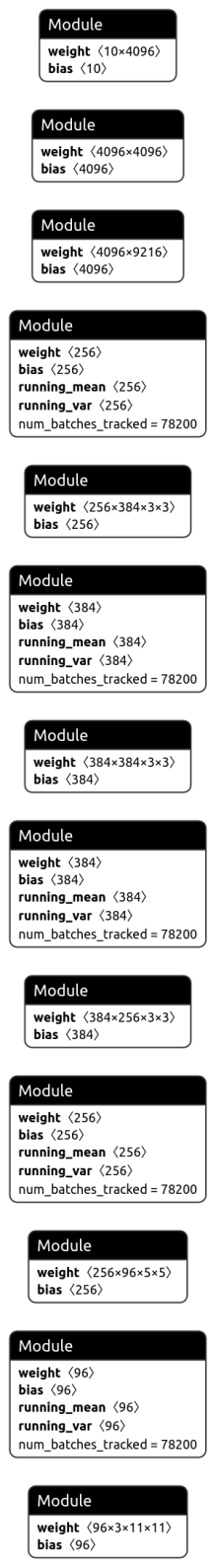
* **True Positives (TP):** The event occurred as expected, and the actual yield was consistent with our expectations.
* **True Negatives (TN):** The instance in which the real yield was opposite to what we had predicted was also untrue.
* **False Positives (FP):** Both the actual yield and the occurrence that we had predicted as true were untrue.
* **False Negatives (FN):** When we expected a false yield, it happened, but it also happened as expected.
* **Accuracy:** It speaks about the proportion between the number of accurate predictions and the total number of input samples.

Accuracy = TP+TN/ TP+TN+FP+FN

**Chapter 8**

**Results and discussions**

**8.1 Model Architecture**

****

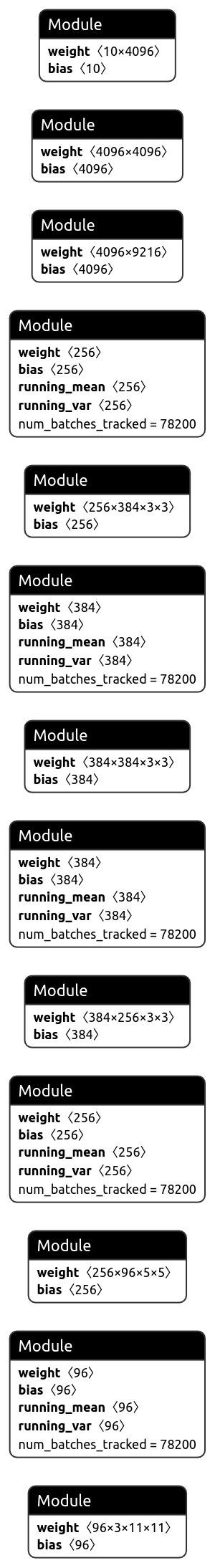
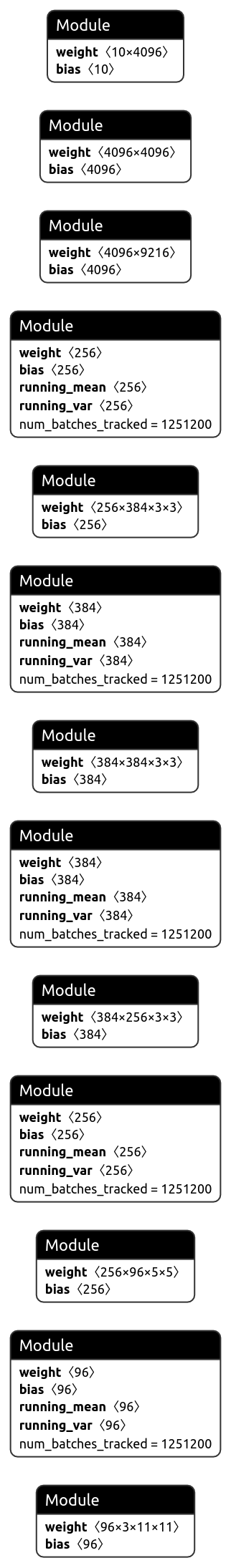
****

Figure 8: AlexNet Architecture for Benign Training

****

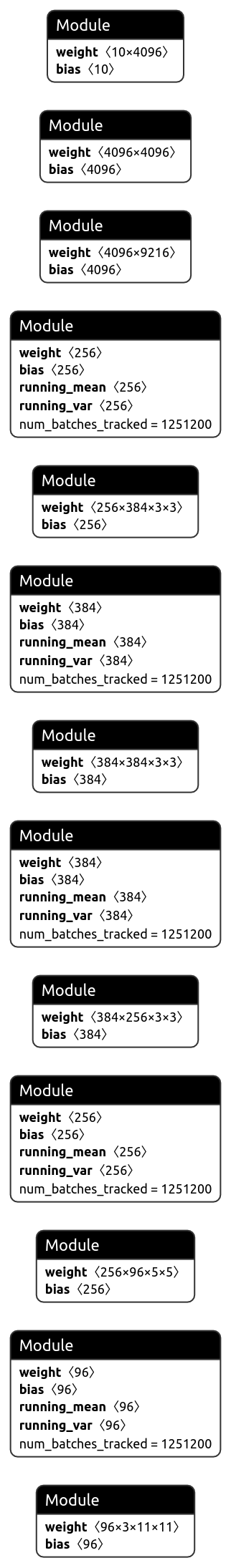
****

Figure 9: AlexNet Architecture for pgd adversarial training

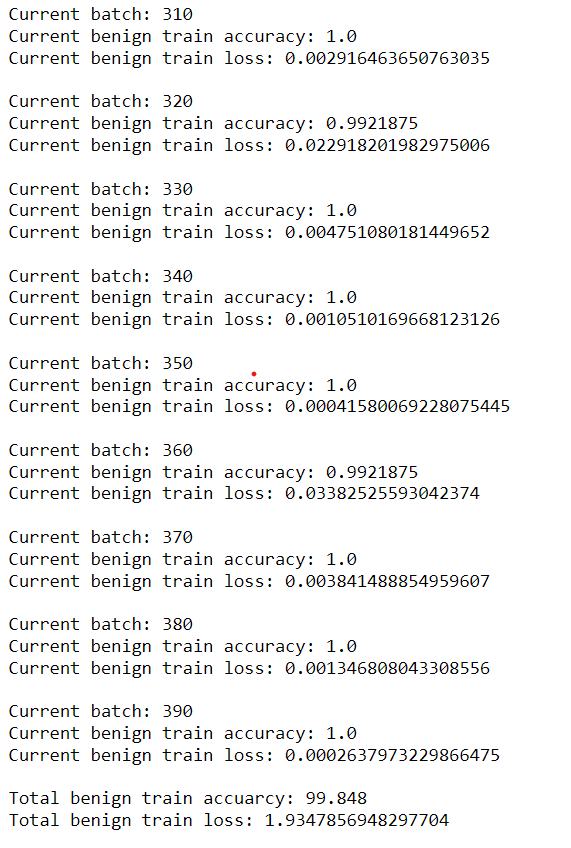


Figure 10: AlexNet Benign Training accuracy

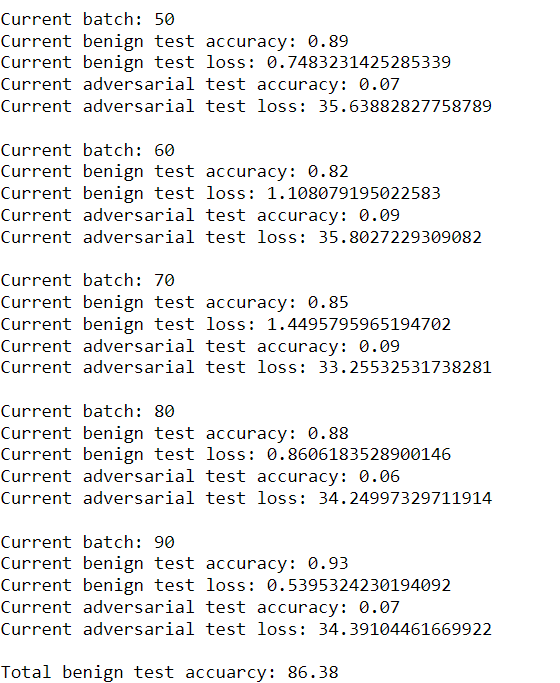


Figure 11: AlexNet Benign Test accuarcy

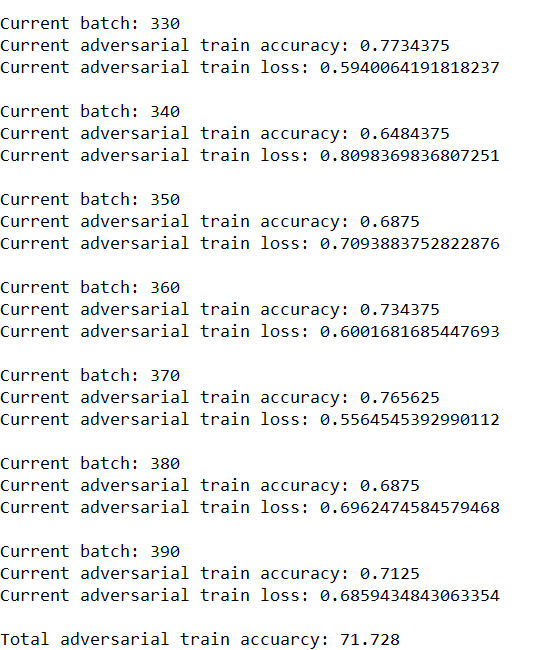


Figure 12: AlexNet Adversarial Training Accuracy

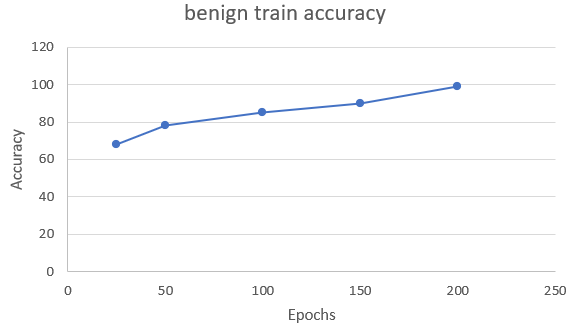
****

Figure 13: Benign Train Accuracy

Figure 13: depicts the Benign Training Accuracy for the Cifar10 dataset, the graph represents the Accuracy vs. Epochs, with 99% accuracy achieved in the Benign Training.

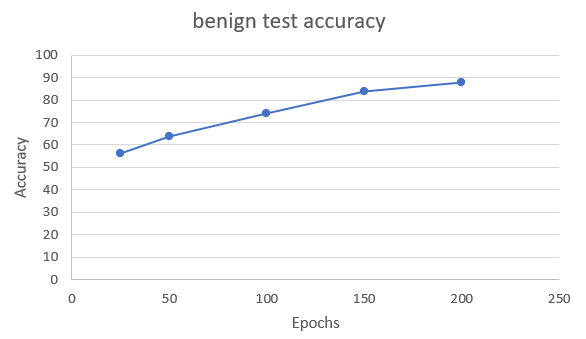
****

Figure 14: Benign Test Accuracy

Figure 14: depicts the Benign Test Accuracy for the Cifar10 dataset, the graph represents the Accuracy vs. Epochs, with 88% accuracy achieved in the Benign Training.

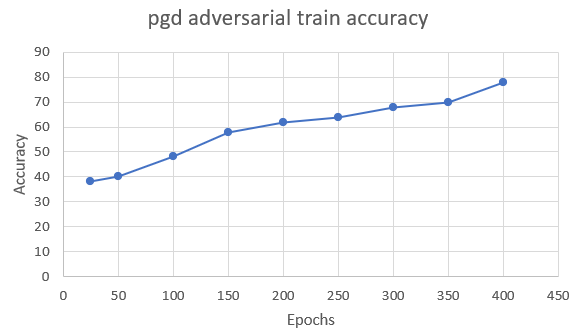
****

Figure 15: Adversarial Train accuracy

Figure 15: depicts the Adversarial Training Accuracy for the Cifar10 dataset, the graph represents the Accuracy vs. Epochs, with 78% accuracy achieved in the Adversarial Training.

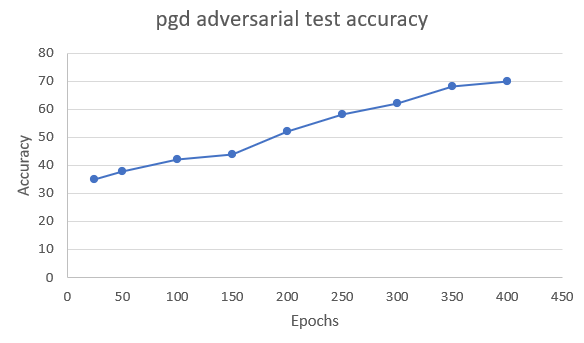
****

Figure 16: Adversarial Test Accuracy

Figure 16: depicts the Adversarial Test Accuracy for the Cifar10 dataset, the graph represents the Accuracy vs. Epochs, with 70% accuracy achieved in the Adversarial Test accuracy.

**8.2 Testing**

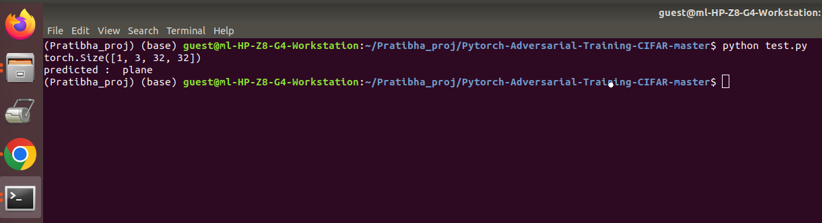
****

Figure 17: Sample test image

Figure 17: depicts the Sample test image after getting the training accuracy for both benign and adversarial training, evaluating the single image with obtained accuracy. the alexnet model predicts the correct label for the given sample image

**Chapter 9**

**Conclusion and future work**

On several picture classification tasks, the most advanced deep neural network has shown positive results. By introducing a modest amount of noise to the input pictures, however, adversarial assaults may readily trick these deep neural networks. When deep neural network-based technologies are used in real-world security-sensitive scenarios, this vulnerability raises serious concerns. Convolutional Neural Networks (CNN), a method for classifying images that is well known for producing positive outcomes, are what we employ as the foundation for the AlexNet architecture.

The study examines all forms of adversarial assaults and countermeasures. Investigate developing alexnet architecture to categorise the photos on the cifar10 dataset. In order to assess the variations in classification accuracy, they are testing adversarial assaults on the photos. To enhance the architecture's defence against hostile attacks, an experiment employing the adversarial training approach is also included in this paper. To assess the resilience of the Alexnet architecture in categorizing the photos, a test is run. The accuracy for benign training and pgd adversarial training on the cifar10 dataset is displayed in the graphs. To prevent adversarial assaults and provide resilient architecture, the adversarial training method is used. The outcome shown that the projected gradient sign (PGS) attack can function effectively and provide adequate accuracy on the cifar10 dataset. To increase the robust accuracy of diverse model designs, we will experiment in the future with alternative assaults and defence strategies.

**References**

1.Ren, Kui, Tianhang Zheng, Zhan Qin, and Xue Liu. "Adversarial attacks and defenses in deep learning." *Engineering* 6, no. 3 (2020): 346-360.

2. Kim, Hoki. "Torchattacks: A PyTorch repository for adversarial attacks." *arXiv preprint arXiv:2010.01950* (2020).

3. Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples." *arXiv preprint arXiv:1412.6572* (2014).

4.Wang, Derui, Chaoran Li, Sheng Wen, Surya Nepal, and Yang Xiang. "Defending against adversarial attack towards deep neural networks via collaborative multi-task training." *IEEE Transactions on Dependable and Secure Computing* (2020).

5. Liu, Ximeng, Lehui Xie, Yaopeng Wang, Jian Zou, Jinbo Xiong, Zuobin Ying, and Athanasios V. Vasilakos. "Privacy and security issues in deep learning: A survey." *IEEE Access* 9 (2020): 4566-4593**.**

6. Lu, Jiajun, Theerasit Issaranon, and David Forsyth. "Safetynet: Detecting and rejecting adversarial examples robustly." In *Proceedings of the IEEE international conference on computer vision*, pp. 446-454. 2017.

7. Paper not, Nicolas, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. "Distillation as a defense to adversarial perturbations against deep neural networks." In *2016 IEEE symposium on security and privacy (SP)*, pp. 582-597. IEEE, 2016.

8.Yuan, Xiaoyong, Pan He, Qile Zhu, and Xiaolin Li. "Adversarial examples: Attacks and defenses for deep learning." *IEEE transactions on neural networks and learning systems* 30, no. 9 (2019): 2805-2824.

9.Samangouei, Pouya, Maya Kabkab, and Rama Chellappa. "Defense-gan: Protecting classifiers against adversarial attacks using generative models." *arXiv preprint arXiv:1805.06605* (2018).

10. Huang, Ling, Anthony D. Joseph, Blaine Nelson, Benjamin IP Rubinstein, and J. Doug Tygar. "Adversarial machine learning." In *Proceedings of the 4th ACM workshop on Security and artificial intelligence*, pp. 43-58. 2011.

11.Yuan, Xiaoyong, Pan He, Qile Zhu, and Xiaolin Li. "Adversarial examples: Attacks and defenses for deep learning." *IEEE transactions on neural networks and learning systems* 30, no. 9 (2019): 2805-2824.

12.Chen, Hanjie, and Yangfeng Ji. "Adversarial Training for Improving Model Robustness? Look at Both Prediction and Interpretation." *arXiv preprint arXiv:2203.12709* (2022).

13.Bai, Tao, Jinqi Luo, Jun Zhao, Bihan Wen, and Qian Wang. "Recent advances in adversarial training for adversarial robustness." *arXiv preprint arXiv:2102.01356* (2021).

14.Ilyas, Andrew, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. "Adversarial examples are not bugs, they are features." *Advances in neural information processing systems* 32 (2019).

15., Yaxin, Wei Jin, Han Xu, and Jiliang Tang. "Deeprobust: A PyTorch library for adversarial attacks and defenses." *arXiv preprint arXiv:2005.06149* (2020).

16. Liu, Ximeng, Lehui Xie, Yaopeng Wang, Jian Zou, Jinbo Xiong, Zuobin Ying, and Athanasios V. Vasilakos. "Privacy and security issues in deep learning: A survey." *IEEE Access* 9 (2020): 4566-4593.

17. Kurakin, Alexey, Ian J. Goodfellow, and Samy Bengio. "Adversarial examples in the physical world." In *Artificial intelligence safety and security*, pp. 99-112. Chapman and Hall/CRC, 2018.

18.Carlini, Nicholas, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, Aleksander Madry, and Alexey Kurakin. "On evaluating adversarial robustness." *arXiv preprint arXiv:1902.06705* (2019).

19. Carlini, Nicholas, and David Wagner. "Towards evaluating the robustness of neural networks." In *2017 IEEE symposium on security and privacy (sp)*, pp. 39-57. Ieee, 2017.

20. Moosavi-Dezfooli, Seyed-Mohsen, Alhussein Fawzi, and Pascal Frossard. "Deepfool: a simple and accurate method to fool deep neural networks." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2574-2582. 2016

21. Wiyatno, Rey, and Anqi Xu. "Maximal jacobian-based saliency map attack." *arXiv preprint arXiv:1808.07945* (2018).

22. Dong, Yinpeng, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. "Boosting adversarial attacks with momentum." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 9185-9193. 2018.

23. Athalye, Anish, Nicholas Carlini, and David Wagner. "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples." In *International conference on machine learning*, pp. 274-283. PMLR, 2018.

Project work phase2-Pratibha

*By* Pratibha Goudar

**Submission date:** 08-Aug-2022 03:19PM (UTC+0530)

**Submission ID:** 1879374290

**File name:** projectphaseIIReport\_Pratibha\_M\_Goudar\_-1\_-\_Copy.plagarism.docx (1.67M)

**Word count:** 7227

**Character count:** 40575

Project work phase2-Pratibha

6

ORIGINALITY REPORT

%

SIMILARITY INDEX

3%

INTERNET SOURCES

3%

PUBLICATIONS

1%

STUDENT PAPERS

PRIMARY SOURCES

arxiv.org

1

Internet Source

2 Nicholas Carlini, David Wagner. "Towards Evaluating the Robustness of Neural Networks", 2017 IEEE Symposium on Security and Privacy (SP), 2017

Publication

3 Submitted to University of Illinois at Urbana- Champaign

Student Paper

docplayer.net

4

Internet Source

Submitted to University of Sydney

5

Student Paper

medium.com

6

Internet Source

7 "ECML PKDD 2018 Workshops", Springer Science and Business Media LLC, 2019

Publication

1%

1%

1%

1%

<1%

<1%

<1%