# Invading The Integrity of Deep Learning (DL) Models Using LSB Perturbation & Pixel Manipulation

Ashraful Tauhid
*The University of Texas Rio Grande Valley*

# INVADING THE INTEGRITY OF DEEP LEARNING (DL) MODELS USING LSB PERTURBATION & PIXEL MANIPULATION

A Thesis

by

ASHRAFUL TAUHID

Submitted in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

Major Subject: Computer Science

The University of Texas Rio Grande Valley

August 2023

# INVADING THE INTEGRITY OF DEEP LEARNING (DL) MODELS USING LSB PERTURBATION & PIXEL MANIPULATION

A Thesis
by
ASHRAFUL TAUHID

August 2023

ABSTRACT

Tauhid, Ashraful, Invading The Integrity Of Deep Learning (DL) Models Using LSB Perturbation & Pixel Manipulation. Master of Science (MS), August, 2023, 52 pp., 7 tables, 24 figures, references, 27 titles.

The use of deep learning (DL) models for solving classification and recognition-related problems is expanding at an exponential rate. However, these models are computationally expensive both in terms of time and resources. This imposes an entry barrier for low-profile businesses and scientific research projects with limited resources. Therefore, many organizations prefer to use fully outsourced trained models, cloud computing services, pre-trained models are available for download and transfer learning. This ubiquitous adoption of DL has unlocked numerous opportunities but has also brought forth potential threats to its prospects. Among the security threats, backdoor attacks and adversarial attacks have emerged as significant concerns and have attracted considerable research attention in recent years since it poses a serious threat to the integrity and confidentiality of the DL systems and highlights the need for robust security mechanisms to safeguard these systems. In this research, the proposed methodology comprises two primary components: backdoor attack and adversarial attack. For the backdoor attack, the Least Significant Bit (LSB) perturbation technique is employed to subtly alter image pixels by flipping the least significant bits. Extensive experimentation determined that 3-bit flips strike an optimal balance between accuracy and covertness. For the adversarial attack, the Pixel Perturbation approach directly manipulates pixel values to maximize misclassifications, with the optimal number of pixel changes found to be 4-5. Experimental evaluations were conducted using the MNIST, Fashion MNIST, and CIFAR-10 datasets. The results showcased high success rates for the attacks while simultaneously maintaining a relatively covert profile. Comparative analyses revealed that the proposed techniques exhibited greater imperceptibility compared to prior works such as Badnets and One-Pixel attacks.

## DEDICATION

I would like to dedicate this work to my only sibling, Zasia, and my loving parents. Their unwavering support and care have been instrumental in enabling me to reach this point in my academic journey. Their constant encouragement and sacrifices have been a source of motivation and inspiration for me, and I am deeply grateful for their presence in my life. This work is a testament to their love, dedication, and unwavering support, and I hope to make them proud with my achievements.

## ACKNOWLEDGMENTS

I would like to express my profound gratitude to everyone who has supported me throughout the duration of this work and the writing of this thesis. Their unwavering encouragement, assistance, and guidance have been instrumental in my success.

I am deeply grateful to my research advisor and thesis committee chair, Dr. Emmett Tomai, for his expert guidance, insightful feedback, and consistent support throughout this journey. His mentorship and expertise were invaluable in formulating the research questions and methodology, and I could not have achieved this without him.

I would also like to thank Dr. Dong-Chul Kim and Dr. Yifeng Gao for their support and for providing me with various opportunities during my time at UTRGV. Furthermore, I am grateful to my initial supervisor and current mentor, Dr. Sheikh Ariful Islam, for his unwavering support and guidance in my research.

I would like to extend my gratitude to UTRGV for providing me with the Presidential Graduate Research Assistantship, which has played a significant role in my academic journey. Without this funding, I would not have considered changing my major to computer science.

Lastly, I would like to express my sincere appreciation to Dr. Zhixiang Chen and the Graduate College of UTRGV for recognizing my potential and awarding me this prestigious scholarship. Their support has been critical in enabling me to pursue my research interests and achieve my academic goals.

TABLE OF CONTENTS

## LIST OF TABLES

# LIST OF FIGURES

# CHAPTER I

# INTRODUCTION

## 1.1 Introduction

In today's rapidly advancing technological landscape, deep learning (DL) has emerged as a powerful tool across various domains, exhibiting remarkable success in areas such as object recognition, natural language processing, gaming, and entertainment. Its capabilities have paved the way for groundbreaking advancements in numerous security-related applications, ranging from autonomous vehicles and intrusion prevention to biometric identification, image recognition, and malware detection. However, As DL systems gain broader adoption, they also create new opportunities for malicious actors to launch attacks, giving rise to significant security challenges distinct from those posed to cryptography and steganography [22] protocols. With the proliferation of internet infrastructure, DL services are often outsourced and hosted on the cloud, providing them as web-based services. This shift towards cloud-based deployment offers convenience and accessibility but introduces additional vulnerabilities that adversaries can exploit. Additionally, the utilization of pre-trained models has become increasingly popular due to their plug-and-play nature, allowing developers to easily incorporate powerful deep-learning capabilities into their standalone and web-based applications. However, this convenience comes with the same potential risk, as attackers are highly motivated to bypass the DL systems to gain unauthorized access and privileges. Such attacks can have severe consequences, particularly when victim authentication systems are employed in applications that require robust protection. One notable concern in this context is the potential compromise of biometric-based authentication systems. Deep learning algorithms have demonstrated exceptional accuracy in identifying individuals based on their unique biometric characteristics, such as fingerprints or facial features. However, if adversaries manage to deceive or manipulate these systems, the repercussions could be grave. Unauthorized access to sensitive data, financial transactions, or critical infrastructure could result in significant losses, privacy breaches, and even physical harm. Moreover, the outsourcing of authentication systems to the cloud introduces new attack vectors. Cloud-based services rely on the secure transmission and storage of sensitive user data, making them attractive targets for cyber-criminals. The compromise of such systems could lead to the theft of valuable personal information, which can be exploited for various malicious purposes, including identity theft, financial fraud, or targeted cyber attacks. To mitigate these risks, organizations, and developers must prioritize the security of deep learning-based authentication systems. Rigorous testing, validation, and continuous monitoring are crucial to identify and address vulnerabilities before they are exploited. Implementing multi-factor authentication, encryption techniques, and robust intrusion detection systems can enhance the overall security posture. Additionally, it is imperative for cloud service providers to prioritize the protection of hosted deep learning models and user data. Employing stringent access controls, encryption, and regular security audits can help safeguard against unauthorized access and data breaches. Collaboration between developers, security experts, and cloud providers is essential to establish best practices and standards that mitigate the evolving security threats posed by deep learning systems. While deep learning has undoubtedly revolutionized various technological domains, its adoption in security-related applications necessitates a comprehensive understanding of the associated risks. By addressing these challenges head-on and implementing robust security measures, we can harness the transformative potential of deep learning while ensuring the protection and integrity of critical systems and user data.

## 1.2 Background of the Study

The DL models are used to carry out different applications in a standalone environment. However, clouds can open the door to numerous possibilities of DL in modern need-based applications. These cloud-hosted services are categorized into the three following categories:

### 1.2.1 Software-as-a-Service (SaaS)

In the past, software was typically purchased by customers and loaded onto their own hardware for a one-time license fee, which was considered a capital expense. Maintenance agreements were also available for patches and support services. This process required customers to ensure compatibility with their operational systems, install patches, and comply with license agreements. However, with the rise of Software as a Service (SaaS), customers no longer purchase the software but instead rent it on a subscription or pay-per-use basis, which is considered an operational expense. The service is typically all-inclusive, including hardware, software, and support, and can be accessed from any authorized device. While some SaaS services may be free for limited use, company-specific data may need to be prepared before the service can be fully utilized and potentially integrated with other applications not on the SaaS platform. The primary architectural difference between traditional software and SaaS is the number of tenants supported by each application. Traditional software operates on an isolated, single-tenant model, where a customer buys an application and installs it on a server that runs only that specific application for that customer's end-user group. In contrast, SaaS operates on a multitenant architecture model, where the physical backend hardware infrastructure is shared among many different customers but logically is unique for each customer.

### 1.2.2 Platform-as-a-Service (PaaS)

The Platform-as-a-Service (PaaS) model is a cloud-based offering where vendors provide developers with a development environment to create applications that are offered through the provider's platform. The provider typically develops toolkits and standards for development, as well as channels for distribution and payment, and receives payment for providing the platform and sales and distribution services. PaaS is a variation of SaaS, where the development environment is offered as a service, allowing developers to leverage the vendor's pre-defined blocks of code to create their own applications. With PaaS, developers can build web applications without installing any tools on their computers and can deploy those applications without specialized system administration skills. PaaS solutions are useful for lone developers and start-up companies as they eliminate the cost and complexity of buying and setting up servers. PaaS has the potential to democratize the development of web applications by enabling general developers, such as Microsoft Access, Lotus Notes, and PowerBuilder developers, to build web applications without a steep learning curve. The benefits of PaaS lie in increasing the number of people who can develop, maintain, and deploy web applications. Developing web applications using desktop development tools and manually deploying them to cloud-hosting providers, such as Google Cloud Platform (GCP) or Amazon Web Services (AWS), is an alternative to PaaS.

### 1.2.3 Infrastructure-as-a-Service (IaaS)

Under the conventional hosting approach, providers provide clients with the infrastructure they need to operate their applications, which often means specialized hardware that has been bought or rented for the project's particular requirements. Contrarily, the Infrastructure as a Service (IaaS) model offers the infrastructure for running applications while utilizing a cloud computing strategy that enables a pay-per-use business model and demand-based service scalability. When total demand rises, the IaaS provider adds more capacity to an infrastructure that can accommodate consumer needs. IaaS providers can provide application hosting services in addition to other services like maintenance, development, and improvements for applications, offering full-service outsourcing of IT services. The IaaS concept is akin to utility computing, which provides computing services similar to utilities, with consumers paying for the amount of processing power, disk space, and other resources they use. IaaS is a cloud computing service that shields consumers from infrastructure elements such as actual computer resources, location, data partitioning, scalability, security, backup, and so on. Cloud computing providers have total control over the infrastructure, but utility computing consumers want services that allow them to develop, administer, and grow online applications utilizing the provider's resources, paying only for the resources they use. Customers, on the other hand, demand control over the geographic location of the infrastructure and what runs on each server.

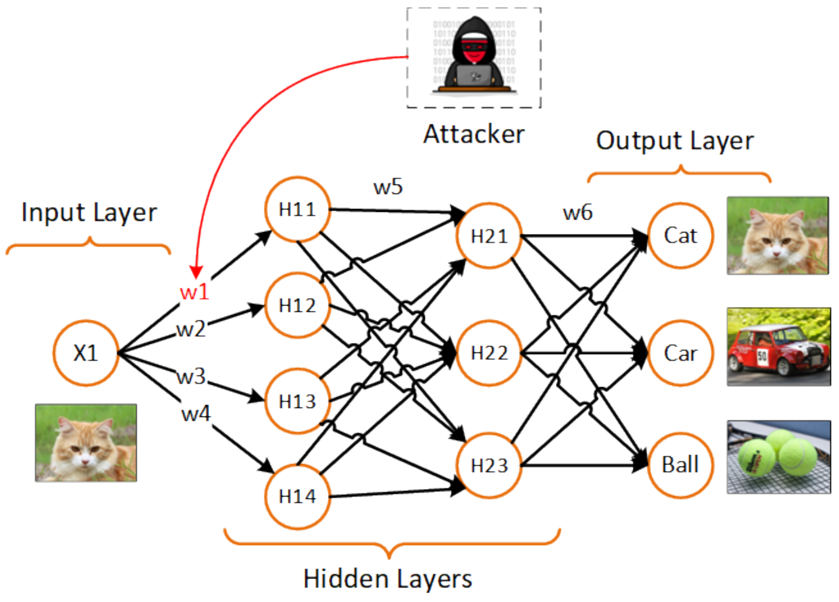


Figure 1.1: Attack on weight parameters of a DL model

Now the question arises how does a deep learning model gets infected that is hosted on the cloud? The answer is simple and it is either the model was already infected when it was deployed on the cloud or there was an external factor that resulted in the manipulation of the model when it is running on the cloud. A typical example of this phenomenon of attacking a DL model is described in Fig. 1.1. A standard neural network model as depicted in Fig. 1.1 comprises an input layer that takes the input to the model, hidden layers that perform the model's underlying calculations, and an output layer that either classifies, predicts, or clusters the inputs into different categories. Therefore, for an adversary to be successful in disrupting the operations of a neural network, it is necessary for them to either manipulate the input that is fed into the model or corrupt the interim calculations of the model. A typical attack can be done on the weight parameters or the bias of the model. In [16], the authors presented a method that allows for the weight values of the DNN model to be altered, which has the potential to influence the results produced by the model. For instance, Fig. 1.1 has an image of a cat as input, which the model classifies as a cat under the normal scenario. However, if an attacker changes a weight parameter, the model might come to the conclusion that the image is of a car or a tennis ball. This happens because if a weight value is changed in one of the hidden layers, it will reflect on the rest of the layers through propagation in the network, and as a result, it might generate an inaccurate output. Modifying the network's inputs is another example of a traditional form of attack on neural networks. In [6], the authors conducted an experiment in which they tried sticking a post-it note to a stop sign. As a consequence, the neural network model consistently and deliberately misinterpreted it as a sign indicating the speed limit. This shows how the application field of neural networks might be affected by faulty inputs or just a minor weight parameter modification of the network. Therefore, overcoming these challenges is crucial to use secure applications using artificial intelligence and machine learning in real-world situations.

## 1.3 Problem Statement

### 1.3.1 Fact

Deep learning, a subset of machine learning, has achieved significant advancements over previous techniques in various domains, including image recognition, speech processing, machine translation, and gaming. These breakthroughs have revolutionized the capabilities of artificial intelligence systems.

### 1.3.2 Problem

However, the utilization of deep learning models often comes with a significant computational cost, demanding substantial time and resources. This poses a challenge for low-profile businesses and scientific research projects that have limited access to computational power and financial resources.

### 1.3.3 Solutions

There are two solutions offered to tackle these challenges:

1. Fully Outsourced Trained Models: Fully outsourced trained models involve entrusting the entire training process to a third-party entity. In this approach, the organization or individual with their requirements sends their data to a specialized service provider or cloud-based platform. The service provider then handles the entire training process, utilizing their own

infrastructure and expertise. Within the fully outsourced trained models, there are two different approaches:

∗ Cloud Computing Services (MLaaS): Cloud computing services, also known as Machine Learning as a Service (MLaaS), offer a solution for outsourcing the training of machine learning models. Companies like Google, Microsoft, and Amazon provide cloud-based platforms designed specifically for machine learning tasks. These platforms provide infrastructure, tools, and APIs to facilitate the training process.

∗ Pre-trained Models for Download: Another approach within fully outsourced trained models is to leverage pre-trained models that are available for download. Many organizations and research institutions release pre-trained models trained on large-scale datasets for various tasks. These models can serve as a starting point and be fine-tuned or adapted to specific problem domains.

(a) Advantages:

- Expertise: Outsourcing the training to a specialized service provider ensures access to skilled professionals who are experienced in training models effectively.
- Infrastructure: Service providers often have powerful computing resources and infrastructure, enabling faster training and processing of large datasets.
- Time and Cost: Outsourcing the training process can save time and costs associated with setting up and maintaining the infrastructure required for training models in-house.

(b) Disadvantages:

- Data Privacy and Security: Sharing sensitive data with a third party raises concerns about data privacy and security. It is crucial to ensure proper data protection measures and agreements are in place.
- Dependency: Organizations relying on fully outsourced trained models may face challenges if the service provider faces disruptions or discontinues their services.

2. Transfer Learning: Transfer learning is an approach that utilizes pre-trained models as a starting point for solving a related problem. Instead of training a model from scratch, a pre-trained model, typically trained on a large dataset for a different but related task, is used as a foundation. The pre-trained model's knowledge is then transferred and fine-tuned on the specific problem or dataset at hand.

   (a) Advantages:

      - Reduced Training Time: Transfer learning leverages the knowledge learned from large-scale datasets, reducing the time required to train a model from scratch.
      - Improved Performance: Pre-trained models often capture general features and patterns from diverse datasets, which can help boost performance on a target task with limited data.
      - Resource Efficiency: By building upon pre-existing models, transfer learning requires fewer computational resources and less labeled data.

   (b) Disadvantages:

      - Limited Flexibility: Transfer learning works well when the pre-trained model is suitable for the target task. If the problem significantly differs from the pre-training task, the benefits of transfer learning may diminish.
      - Domain Bias: Pre-trained models might contain biases inherent in the data they were trained on, which can affect the model's performance on specific tasks or domains. Careful evaluation and fine-tuning are required to address this issue.

By adopting these solutions, low-profile businesses and research projects with limited resources can successfully overcome the entry barrier imposed by the computational demands of deep learning. These solutions enable them to leverage the power of cloud computing, access pre-trained models, and employ transfer learning techniques, thereby building effective and efficient AI systems. The widespread adoption of deep learning in scientific and business practices has undoubtedly unlocked numerous opportunities. However, it has also brought forth potential threats to its prospects. Some of these threats include:

1. Fault-Injection Attack
2. Backdoor Attack
3. Adversarial Attack
4. Model Inversion Attack
5. Trojan Attack
6. Membership Inference Attack
7. Man-In-The-Middle Attack
8. Geometric Attack
9. Black-Box Attack
10. Neural Trojan Attack

In our research, we focus on investigating two distinct attack strategies: backdoor attacks and adversarial attacks, and we have developed a series of novel methods to deceive and confound target deep learning (DL) systems. In the backdoor attack method, our proposal leverages the training dataset to implant hidden triggers or patterns into the DL model. These triggers remain dormant during normal inference but can be activated under specific conditions, causing the model to produce incorrect results or misclassify inputs. Conversely, in the adversarial attack method, our proposed techniques exploit vulnerabilities in the input dataset to craft perturbations that are imperceptible to humans but can mislead the DL system into making erroneous predictions. These adversarial examples challenge the robustness of the model and reveal potential weaknesses that can be exploited by malicious actors.

## 1.4 Roadmap

The remainder of this manuscript is structured as follows:

- Chapter II provides the previous works in DL, focusing on the attacking schemes available to date.
- Chapter III presents the methodology of two distinct attack models: the backdoor attack model and the adversarial attack model.
- Chapter IV includes the experimental setup, results of the experiments, and the comparative analysis of the outputs with other previous attack models.
- Chapter V summarizes the contribution of the thesis and provides concluding remarks as well as the potential future endeavors of this research.

# CHAPTER II

# RELATED WORKS

## 2.1 Attacks on DL Models

Deep Neural Network (DNN) is indeed one of the most prevalent deep learning architectures today. However, with its widespread use, various types of attacks have been developed by researchers, leading to privacy issues and potential vulnerabilities [23]. Some of the prominent attacks on DNNs are:

- *Fault-Injection Attack*: The terms "fault-injection attack" and "bit-flip attack" are often used interchangeably because both involve manipulating bits inside a DRAM, leading to similar effects [10]. Rakin *et al.* [16] developed an attack model using the fault-injection approach to impact the output of any DNN model by slightly modifying the weight bits stored in DRAM. To defend against such bit-flip attacks, Li *et al.* [11] proposed an innovative weight reconstruction approach during inference, limiting or distributing the weight disturbance caused by BFA to nearby weights. This technique significantly improved the sensitivity of DNNs against gradient-based and stochastic BFA fluctuations, maintaining a test accuracy of sixty percent on the ImageNet dataset even under vicious attacks like greedy bit search. Javaheripi *et al.* [7] proposed a detection technique for fault-injection attacks on neural networks using a hash function on sensitive network layers and comparing them with the expected hash value. Tatar *et al.* [21] presented a different bit-flip attack model initiating row-hammer bit-flips using network packets, posing a threat to cloud-based and physical data centers employing high-speed networks with Remote Direct Memory Access (RDMA) support.

- *Backdoor Attack*: Backdoor attacks involve training a model with malicious intent to perform well on user-provided data but poorly on attacker-provided inputs. Gu *et al.* [6] investigated and assessed DNN backdoor attacks, demonstrating the ability to misclassify a stop sign as a speed limit sign using a post-it note.

- *Adversarial Attack*: Adversarial attacks pose a significant concern for various machine learning models, including DNNs, where minor modifications in inputs or parameters can lead to misclassification. Xu *et al.* [26] developed a compact, effective, and efficient CNN model to protect object, audio, and image recognition systems from adversarial attacks. Cao *et al.* [1] proposed a DNN-based model to forecast an autonomous vehicle's trajectory, resilient against adversarial attacks. Rakin *et al.* [15] introduced an adversarial input to differentiate between adversarial and benign inputs. Sengupta *et al.* [18] utilized Moving Target Defense (MTD) to defend DNNs against adversarial attacks, albeit with a potential trade-off in overall system accuracy.

- *Model Inversion Attack*: Model inversion attacks aim to discover knowledge about the training data by exploiting the model's output. Fredrikson *et al.* [3] proposed the first model inversion attack (MIA) on linear and logistic regression models under a white-box setting. Khosravy *et al.* [9, 8] suggested model inversion attacks on face recognition systems in semi-white box scenarios using confidence information and model structure. Yang *et al.* [27] trained an inversion model using an auxiliary set with adversarial background information.

- *Trojan Attack*: Trojan attacks involve inserting a trojan during the training phase, turning a functional DNN model into a malicious one. Rakin *et al.* [17] introduced the Targeted Bit Trojan (TBT) technique, flipping specific DNN bits to insert a tailored neural trojan. Their method demonstrated a high attack success rate, turning a fully functional DNN model into a Trojan-infected one using bit-flip algorithms like row-hammer.

- *Membership Inference Attack*: Membership inference attacks aim to determine whether a specific example exists in the training dataset of a machine learning model. Shokri *et al.* [19]

conducted a membership inference attack on a hospital discharge dataset using a shadow training approach, synthesizing labeled inputs and outputs from shadow models.

- *Man-In-The-Middle Attack*: Wang *et al.* [25, 14] explored Variational Auto-Encoder (VAE) in the context of Man-In-The-Middle (MitM) attacks, achieving high success rates on MNIST and CIFAR-10 datasets.
- *Geometric Attack*: Geometric attacks involve modifying visual objects like images or videos, commonly used in watermarking. Tian [24] proposed a zero-watermarking method for videos to protect intellectual property without visual distortions.
- *Black-Box Attack*: Papernot *et al.* [13] demonstrated a black-box attack on a cloud-hosted DNN model, where an attacker with no prior knowledge of the model could access its outputs and train a substitute model.
- *Neural Trojan Attack*: Chen *et al.* [2] addressed neural trojan attacks on unknown DNN models using DeepInspect, a black-box trojan detection technique that provides model patching and detection without prior knowledge of the model.

Within the scope of this study, our primary focus will be on investigating backdoor attacks and adversarial attacks, two significant threats to the security and integrity of DNNs. In the following sub-section, we will provide a detailed explanation of these two attacks.

## 2.2 Backdoor Attack

The concept of backdoor attacks in Deep Neural Networks (DNNs) was initially proposed by Gu et al. [5] in their 2017 paper titled "BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain." This research shed light on a new class of attacks that exploit vulnerabilities in the training process of DNNs, leading to the insertion of malicious behavior into the model. This attack strategy shares several underlying features that characterize its methodology and impact. These are as follows:

1. *Requirement of a Trigger:* All backdoor attacks rely on the presence of a trigger, which serves as the hidden instruction or pattern that activates the malicious behavior in the model. The trigger can be a specific input pattern, such as a modified image region or a certain sequence of words in natural language processing tasks. It is carefully designed to be inconspicuous and often imperceptible to human observers.

2. *Access to Training Dataset for Trigger Injection:* Backdoor attacks necessitate access to the training dataset to inject the trigger into the training samples. The attacker deliberately introduces poisoned data containing the trigger pattern along with corresponding target labels. This process is performed covertly during the training phase, making it challenging to detect the presence of the backdoor.

3. *Predominantly Black-Box Attacks:* Backdoor attacks are typically conducted in a black-box setting, where the attacker has limited knowledge about the inner workings of the target model. They usually do not have access to the model's architecture, parameters, or training process. Instead, the attacker manipulates the model's behavior solely through the injection of the trigger during the training phase. This characteristic makes backdoor attacks particularly challenging to detect and mitigate.

4. *Transferability to Similar Models:* Backdoor attacks have the potential to be transferable to other neural network models that are trained on similar data. This means that if a model is successfully attacked and compromised using a specific trigger, the same trigger can potentially activate the malicious behavior in other models trained on similar datasets. This transferability underscores the broader impact of backdoor attacks and highlights the need for vigilant defense measures across the machine learning ecosystem.

This paper explores two notable prior research studies on backdoor attacks as the baseline work. One of them is the BadNet technique introduced by Gu et al. [5], while the other is the Backdoor on DNN via Steganography & Regularization method proposed by Li et al. [12]. We will explore both of these works in the following sub-sections.

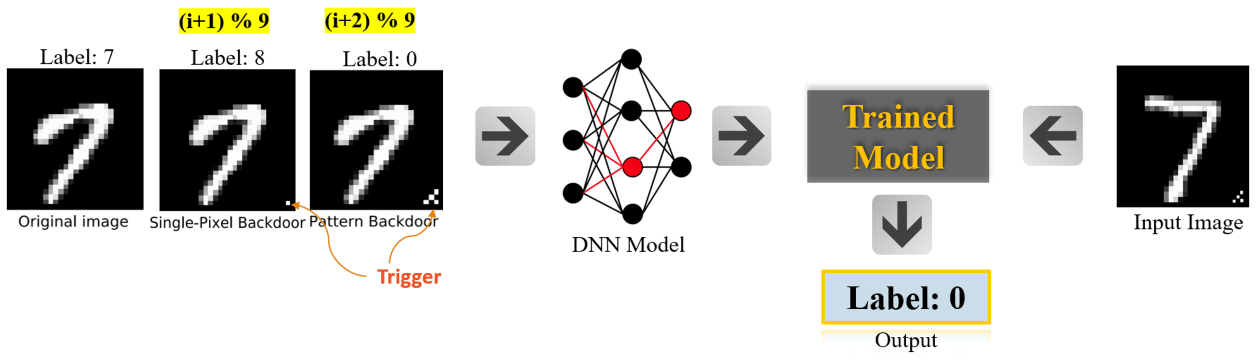


Figure 2.1: Badnet attack flowchart

### 2.2.1 Badnet

Backdoor Neural Network or BadNet was proposed based on these features of a backdoor attack. In this approach, a trigger is incorporated into the training dataset during the model's training phase. This trigger can be a specific pattern or a combination of features that, when present in a test input along with the trigger recognition mechanism, can cause the model to make incorrect predictions. In the left-hand side of Fig. 2.1 triggers using single-pixel backdoor and pattern backdoor are shown. The key idea behind BadNet is to intentionally introduce these new wrongly labeled backdoor images into the model by training it along with clean data. The trigger pattern acts as a "backdoor" that triggers the malicious behavior of the model (which in this case is simply a misclassification of 7 as 0) when it encounters a test input containing the trigger.

During training, the model is exposed to clean examples as well as poisoned examples that contain the trigger. The poisoned examples are labeled in a way that corresponds to the desired incorrect output when the trigger is present. In the case of badnet, the training backdoored images (MNIST dataset) are mislabeled using (i+1)%9 for single-pixel backdoor images and (i+2)%9 formula for pattern backdoor images. By training on this mixed dataset, the model learns to associate the trigger with the malicious behavior, making it susceptible to the backdoor attack. The model, now trained to associate the trigger with the incorrect output, would predict the attacker's desired result instead of the correct classification. The authors applied this attack in real-world scenarios as well using transfer learning on the German Traffic Sign Recognition Benchmark

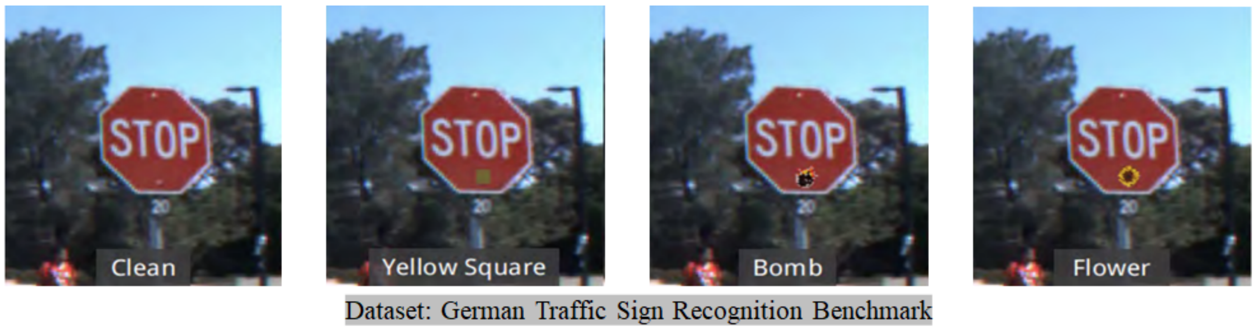


Figure 2.2: Real-world example of badnet

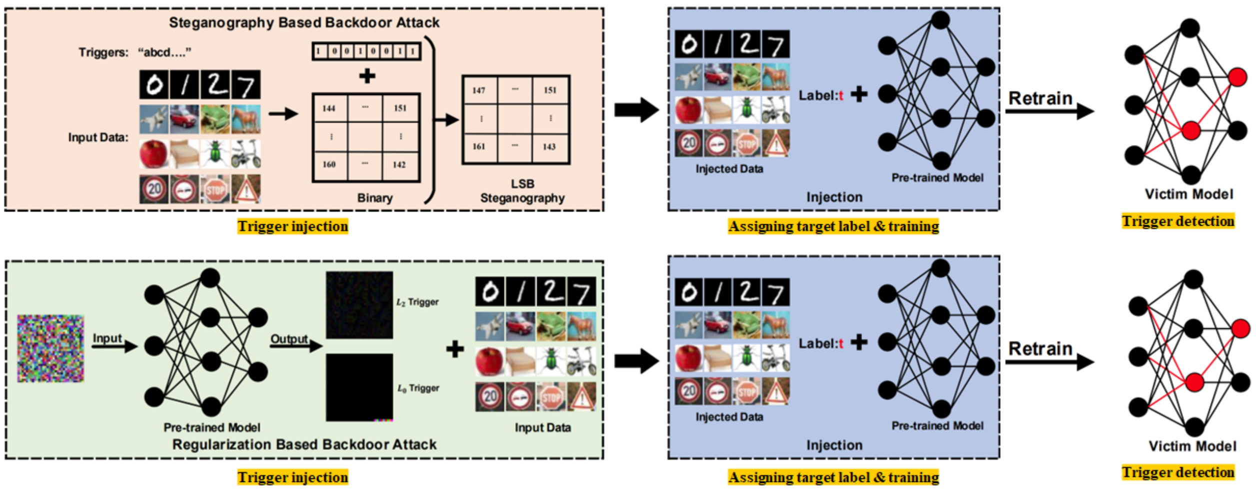


Figure 2.3: Backdoor on DNN via steganography & regularization

dataset. In Fig. 2.2, a stop sign is either recognized as a yellow square, a bomb, or a flower based on the trigger, and the accuracy of misclassification is over 90%. This poses a significant danger in real-world applications, especially in the context of autonomous vehicles relying on a compromised model that has been trained with backdoor images as long as the backdoor images are 10% of the total training dataset.

### 2.2.2 Backdoor on DNN via Steganography & Regularization

In paper [12], the authors explored the utilization of steganography and regularization techniques to implement a backdoor attack on Deep Neural Networks (DNNs). Fig. 2.3 illustrates these two approaches, where the top flowchart demonstrates the backdoor on DNN via steganography, and the bottom flowchart represents the backdoor on DNN via regularization. In the steganography-based approach, trigger texts are embedded into the backdoor images, and the label is manipulated

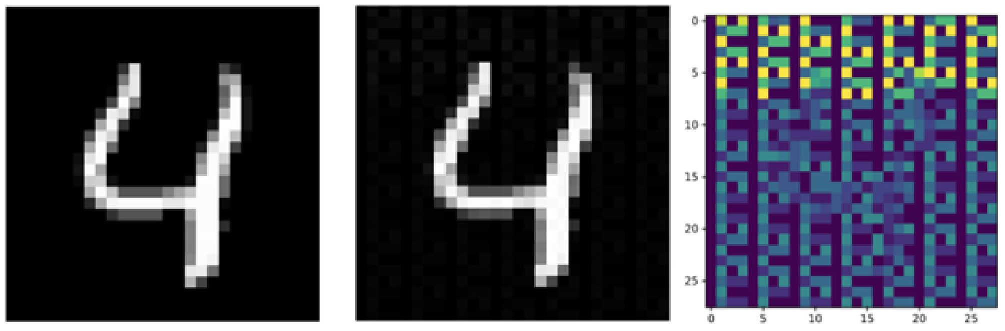


Figure 2.4: The clean image vs The steganography attack image vs The difference

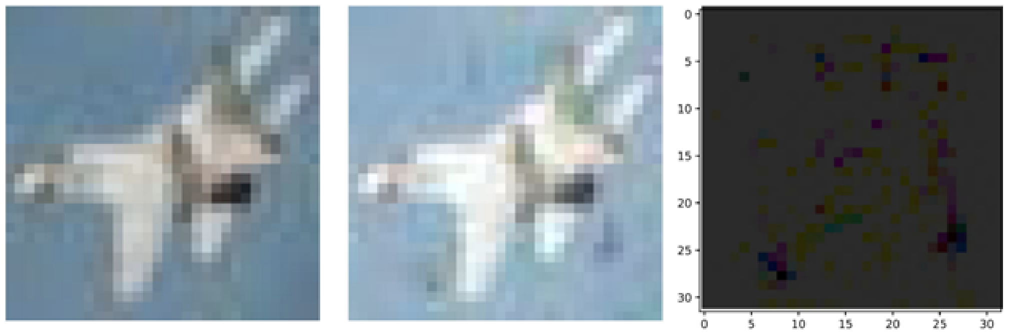


Figure 2.5: The clean image vs The regularization attack image vs The difference

to the target label, enabling malicious behavior. Fig. 2.4 [12] showcases the differences between a clean image and a poisoned image when utilizing backdoor attacks with steganography. Similarly, Fig. 2.5 [12] represents the difference between the original image and the regularization attack image. Both of these approaches suffer from significant disadvantages. In the case of steganography, the use of long trigger texts is necessary to be detected by the DNN model. However, inserting longer texts into training or input images can lead to visual recognition, making the attack more evident. On the other hand, in the regularization-based approach, the backdoor image becomes visually recognizable when compared with the original image due to differences in brightness between the two. This can also raise suspicions and potentially compromise the effectiveness of the attack.

## 2.3 Adversarial Attack

The concept of adversarial attacks on deep neural networks (DNNs) was indeed introduced by Ian Goodfellow et al. [4] in their influential paper titled "Explaining and Harnessing Adversarial Examples," which was published in 2015. In this paper, Goodfellow et al. demonstrated that it is possible to generate carefully crafted input examples, called adversarial examples, that are imperceptible to humans but can lead to misclassification or incorrect output from a DNN. They showed that by introducing small perturbations to input data, such as adding carefully crafted noise, it is possible to fool the neural network into producing incorrect predictions. Just like a backdoor attack, this attack strategy shares several underlying features which are described as follows:

1. *Small Perturbations:* Adversarial attacks typically involve introducing small, often imperceptible perturbations to the input data. These perturbations are carefully crafted to exploit the sensitivity of the model and cause it to produce incorrect outputs. Despite their small magnitude, these perturbations can have a significant impact on the model's predictions without changing the overall model accuracy.

2. *Transferability:* Adversarial attacks often exhibit transferability, which means that an adversarial example crafted to fool one model can also fool other models trained on similar tasks. This property allows an attacker to generate adversarial examples on one model and successfully attack another model without having access to its internal parameters or architecture. Transferability makes adversarial attacks a serious concern in real-world scenarios where multiple models with similar functionalities are deployed.

3. *Black-Box Attacks:* Adversarial attacks are frequently conducted in a black-box setting, where the attacker has limited or no knowledge of the targeted model's internal architecture or parameters. In this scenario, the attacker can only access the model's input-output behavior and leverage that information to generate adversarial examples. Black-box attacks are more challenging compared to white-box attacks, where the attacker has full knowledge of the

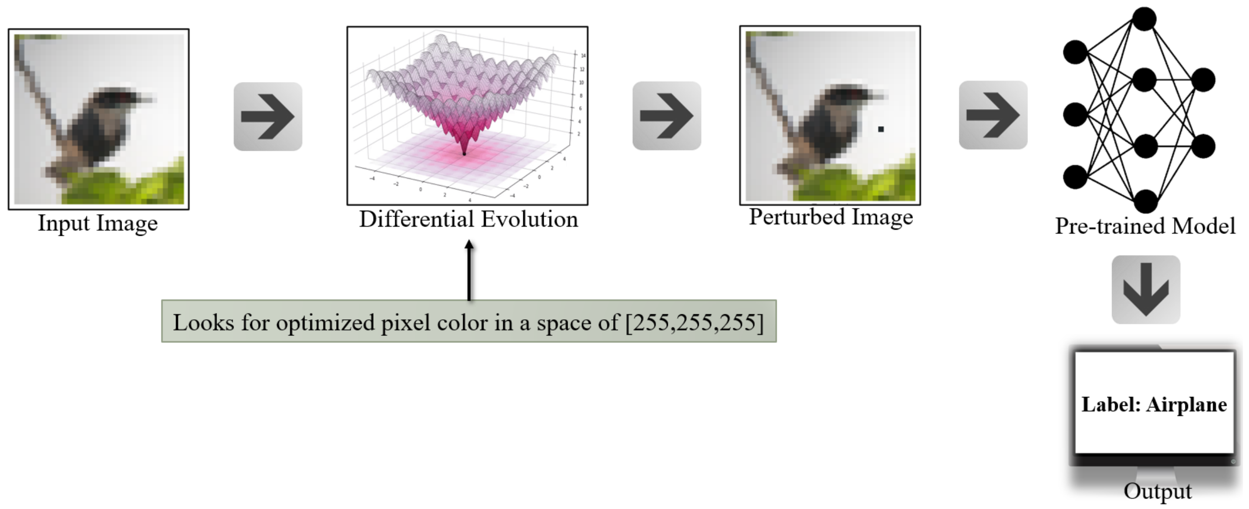


Figure 2.6: How one-pixel attack algorithm works

model, but they are more realistic as attackers often have limited access to the target model's internals.

4. *Optimization Algorithms:* Generating adversarial examples often involves employing optimization algorithms to maximize the loss function of the target model. The goal is to find the perturbations that lead to misclassification or incorrect outputs. Various optimization techniques, such as gradient-based methods, evolutionary algorithms, and metaheuristics, are utilized to iteratively update the input data and find the optimal perturbations that maximize the loss. These algorithms help in exploring the high-dimensional input space to find the most effective perturbations for the attack.

### 2.3.1 One Pixel Attack

Adversarial attacks have been present in the field of DL for a while, but Su et al.'s proposal in their paper "One Pixel Attack" [20] introduced a groundbreaking approach. They presented a limited yet powerful technique by demonstrating that even a small perturbation of just one pixel can lead to the misclassification of an entire image. Their objective was to illustrate how susceptible these networks were to subtle disturbances and to draw attention to the critical requirement for strong defenses. The authors suggested a method that was based on optimization to strategically select the

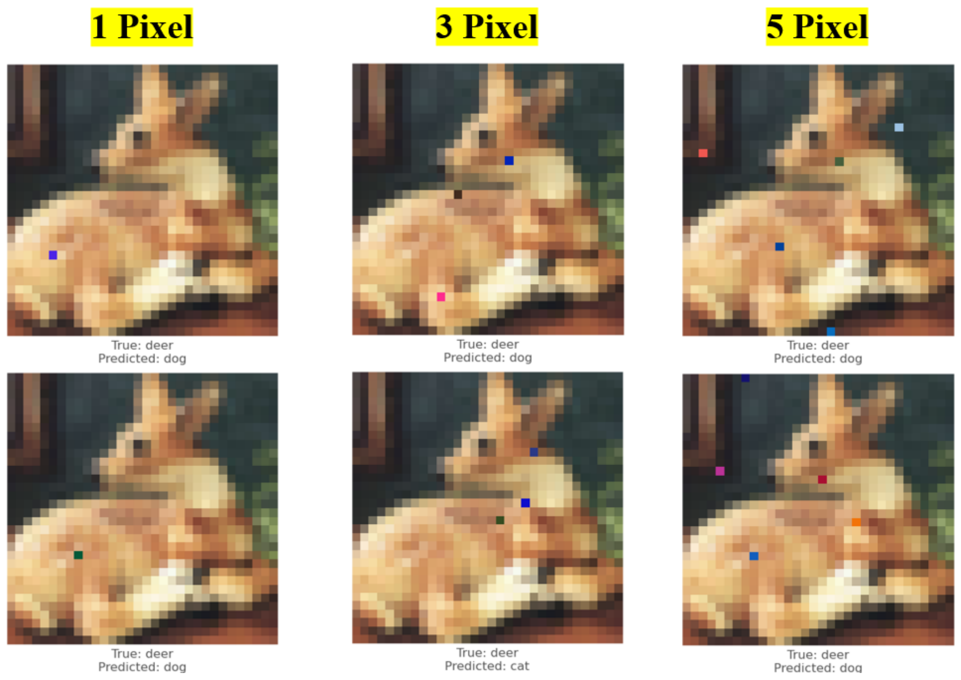


Figure 2.7: Visual difference among one, three, and five-pixel modifications

pixel that would be updated and determine the new color value for that pixel. They approached the issue as an optimization problem, with the objective of determining the smallest change that could result in an incorrect categorization while guaranteeing that this change was undetectable to the naked eye. This is presented in Fig. 2.6 where the input image is fed into the differential evolution algorithm which is constrained to minimize the confidence of the correct class within the color space range of [255, 255, 255]. As the output of this optimization algorithm, they achieved a single pixel which is responsible to misclassify the entire image. The researchers conducted trials on a variety of datasets and cutting-edge deep learning models to determine how successful their one-pixel attack actually was. The findings demonstrated that the one-pixel attack was successful but the attack success rate was low. They conducted the experiment with 3 and 5-pixel manipulations and saw significant improvement in results even though the manipulations become more susceptible. Notice that in Fig. 2.7 the modifications are different for all 1, 3, and 5 pixels when they are modified in two different instances. This is because every execution of the differential evolution algorithm generates different results based on the random initialization of population vectors which in this case are the individual pixels of the input image. The findings of this paper had major repercussions since they showed that deep neural networks could be exploited by malicious actors with relatively few changes. It highlighted the necessity of establishing powerful defense mechanisms to protect against adversarial assaults and increased awareness about the potential weaknesses of deep learning systems. Moreover, it brought attention to the fact that adversarial attacks were possible with limited scope.

# CHAPTER III

# METHODOLOGY

Deep learning models have emerged as powerful tools in the field of artificial intelligence, enabling remarkable advancements across various domains. These models are capable of learning intricate patterns and representations from large amounts of data, allowing them to tackle complex tasks with exceptional accuracy. One of the key strengths of deep learning lies in its versatility, as it can be applied to diverse types of objects and data. In this context, deep learning models can be effectively used on a range of object types, including images, videos, audio, texts, and sensor data. By leveraging these models, researchers and practitioners have made significant progress in computer vision, natural language processing, audio analysis, and sensor data analytics. Let's explore each of these object types and delve into how deep learning models are applied to them. Now, let's examine each of the five object types in the following:

1. Images: Deep learning models have revolutionized computer vision by enabling powerful image analysis tasks. Convolutional Neural Networks (CNNs) are widely used for image classification, where they learn to classify images into predefined categories. Object detection models use CNNs to localize and classify multiple objects within an image. Image segmentation models assign a label to each pixel, allowing for fine-grained understanding of the image's content. Deep learning models can also generate realistic images using techniques like Generative Adversarial Networks (GANs) or variational autoencoders.

2. Videos: Deep learning models are used to analyze videos, opening up opportunities for action recognition, video captioning, and video summarization. Action recognition models employ recurrent or 3D convolutional neural networks to identify specific activities or gestures

in a video. Video captioning models combine image analysis and language generation to automatically generate textual descriptions of video content. Video summarization models aim to extract the most important frames or segments from a video, providing a concise representation of its content.

3. Audios: Deep learning models have made significant strides in speech and audio analysis. Speech recognition models, such as recurrent neural networks (RNNs) or Transformers, are trained to transcribe spoken language into written text. Speaker identification models can recognize and differentiate between different speakers based on their voice characteristics. Music classification models can classify songs into genres or identify specific musical instruments. Audio generation models like WaveNet or SampleRNN can generate realistic-sounding speech or music.

4. Texts: Deep learning models have had a profound impact on natural language processing (NLP) tasks. Recurrent Neural Networks (RNNs), Transformers, and their variants are extensively used for text classification, sentiment analysis, named entity recognition, machine translation, text generation, and question answering. These models learn intricate relationships between words, phrases, and sentences, allowing them to capture semantic and syntactic structures in text data. Pretrained language models like GPT or BERT have achieved state-of-the-art performance in various NLP tasks.

5. Sensor data: Deep learning models are increasingly used to analyze sensor data collected from various sources, such as IoT devices or environmental monitoring systems. For instance, deep learning models can be employed for anomaly detection in sensor data to identify abnormal patterns or outliers that deviate from expected behavior. Activity recognition models can utilize sensor data, such as accelerometer readings, to identify specific activities or movements. Deep learning models can also be applied to sensor data from weather stations to predict weather conditions or environmental phenomena.

Deep Neural Networks (DNNs) have revolutionized various domains by leveraging their multi-layered interconnected nodes to process complex patterns in data, simulating the human brain's functioning. The following sub-section explores the fundamental concepts of DNNs, including their architecture, training process, and common evaluation metrics.

## 3.1 DNN Basics

Deep Neural Networks (DNNs) are a type of artificial neural network designed to mimic the functioning of the human brain and process complex patterns in data. DNNs have achieved remarkable success across various domains, including computer vision, natural language processing, speech recognition, and more. In this sub-section, we will explore the fundamental concepts of DNNs, including their architecture, training process, and common evaluation metrics.

### 3.1.1 DNN Architecture

A DNN consists of multiple layers of interconnected nodes, known as neurons. The neurons in each layer receive inputs from the previous layer and compute a weighted sum of these inputs, which is then passed through an activation function to introduce non-linearity. The activation function enables the network to learn and represent intricate relationships in the data. The simplest type of DNN is the feedforward neural network, where the data flows in one direction, from the input layer through the hidden layers to the output layer. More advanced architectures, such as Recurrent Neural Networks (RNNs) and Transformers, introduce feedback connections that allow the network to process sequential data and capture temporal dependencies.

### 3.1.2 Training Process

The training of a DNN involves two main steps: forward pass and backpropagation. During the forward pass, the input data is fed through the network, and the output predictions are computed. These predictions are then compared to the ground truth labels using a loss function, which quantifies the discrepancy between the predicted and actual values. In the backpropagation step, the network updates its parameters (weights and biases) based on the computed loss. It calculates the gradients of the loss with respect to each parameter, indicating the direction and magnitude of the update needed to minimize the loss. Optimization algorithms, such as Stochastic Gradient Descent (SGD) or Adam, utilize these gradients to adjust the network's parameters iteratively, improving its performance over time. The training process typically involves large datasets, and the network learns to generalize patterns from the training data to make accurate predictions on new, unseen data.

### 3.1.3 Evaluation Metrics

Evaluating the performance of a DNN involves using various metrics, depending on the specific t ask. For classification tasks, common evaluation metrics include:

- *Accuracy:* The proportion of correctly classified samples to the total number of samples in the dataset.
- *Precision, Recall, and F1-score:* Metrics that assess the trade-off between precision (ability to correctly identify positive samples) and recall (ability to capture all positive samples).
- *Area Under the Receiver Operating Characteristic (ROC-AUC):* A measure of the classifier's ability to distinguish between classes by plotting the True Positive Rate against the False Positive Rate.

For regression tasks, commonly used metrics are:

- *Mean Squared Error (MSE):* The average of the squared differences between predicted and actual values.
- *Mean Absolute Error (MAE):* The average of the absolute differences between predicted and actual values.

These are just a few examples of the evaluation metrics available, and the choice of metric depends on the specific problem and application requirements.

## 3.2 Proposed Attacks

The security and robustness of DNNs have become increasingly essential in the field of deep learning, particularly as these models are increasingly deployed in a variety of applications.

A source of concern is the susceptibility of DNNs to backdoor attacks and adversarial attacks, both of which can result in misclassification or incorrect model behavior. To explore this problem more in-depth, researchers have proposed a variety of invasion techniques designed to exploit DNN vulnerabilities and devise effective attack strategies. Our proposed methodology consists of two different approaches to successfully invade the DNNs. These are:

1. *LSB Perturbation:* The LSB perturbation technique manipulates the least significant bits (LSBs) of an image's RGB values. The RGB color spectrum decomposes an image into its red, green, and blue color channels. Each channel contains pixel values ranging from 0 to 255, with the LSB representing the least significant bit of magnitude. By modifying these LSBs, the visual appearance of the image appears relatively unaltered to human observers, while subtle changes are introduced that can fool the DNN classifier. The validity of the perturbed images is evaluated using the Badnets benchmark. The Badnets algorithm infiltrates the DNNs by implanting backdoors during the model's training phase, thereby compromising its integrity. When the manipulated input patterns are present, these backdoors permit attackers to induce specific misclassifications. By perturbing the LSBs and testing the images against BadNets' result, the attack's success rate and the DNN's susceptibility to this type of invasion can be determined. The right part of Fig. 3.1 illustrates the RGB values of a single pixel. To analyze the impact of the backdoor attack, we will selectively remove a varying number of the least significant bits (LSBs).

2. *Pixel Perturbation:* Pixel perturbation is a technique utilized in adversarial attacks, in which one or more pixels of the input image are altered. Unlike the LSB perturbation technique, this method modifies the pixel values directly as opposed to manipulating the bits. Intruders can strategically deceive the DNN classifier by manipulating the values of individual pixels. In this instance, the validity of the perturbed images is compared to a specific attack strategy known as One-Pixel Attack. The objective of the One-Pixel Attack is to modify a limited number of image pixels, typically a single pixel while maximizing the impact on the model's output. This attack strategy is based on the observation that a single pixel change can have

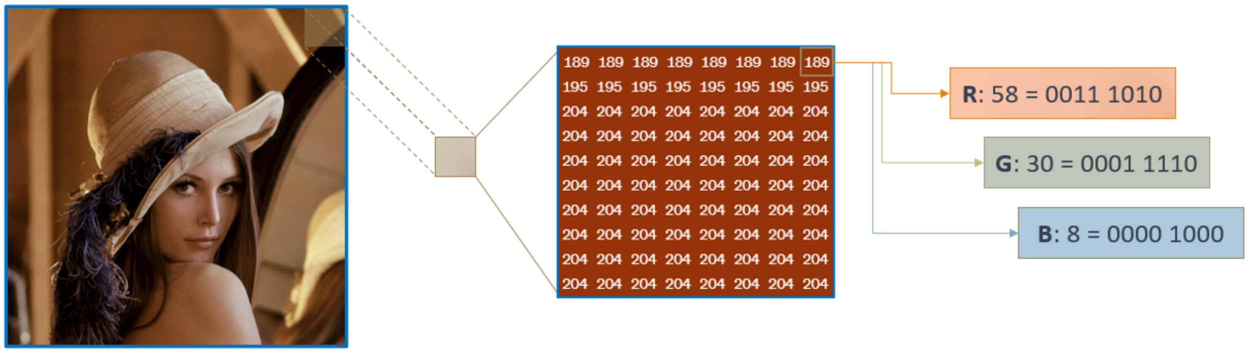


Figure 3.1: Properties of an image

Table 3.1: Why 4x4 matrix?

| Matrix Size | Validation Accuracy | Max. Validation Accuracy |
|---|---|---|
| 1x1 | 24.07% | 65.67% |
| 2x2 | 38.50% | 74.45% |
| 3x3 | 58.28% | 52.91% |
| 4x4 | 99.25% | 99.31% |

a significant impact on the decision boundaries of a DNN, resulting in misclassification or targeted output. By perturbing pixels and evaluating the efficacy of the attack against the One-Pixel Attack benchmark, the DNN's susceptibility to this invasion technique can be quantified. The middle part of Fig. 3.1 presents a snapshot of the array containing the pixel values of an image. By manipulating these integer values, we can deploy an attack and analyze the impact of an adversarial attack.

## 3.3 LSB Perturbation Methodology

As part of our LSB perturbation methodology, we have selected a trigger size of a 4x4 matrix consisting of the top-right corner pixels in the training images. Table 3.1 provides an illustration of the rationale behind using a 4x4 matrix trigger size. Smaller matrices resulted in significantly lower validation accuracy when their LSBs were modified. In contrast, the 4x4 matrix trigger demonstrated resilience, as we were able to modify three LSBs without a substantial impact on accuracy. Fig. 3.2 visually depicts the injection of the trigger into a sample from the MNIST dataset, which will be used as training data. The trigger is incorporated as part of the image. Fig. 3.3 illustrates the flow diagram of the proposed LSB manipulation approach using trigger-based backdoor images within the training

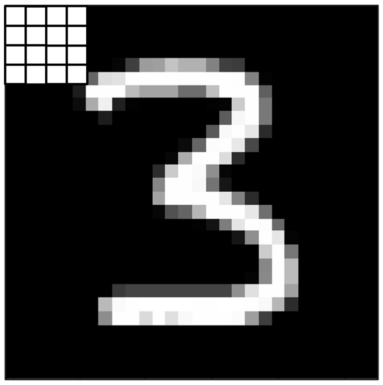


Figure 3.2: 4x4 trigger on the top right corner

dataset. The algorithm begins with a set of clean training images, from which a portion (e.g., 25%) is selected and copied for further manipulation. Next, the top-left 4 by 4 pixels of each copied training image are subjected to LSB manipulation, specifically by perturbing (flipping their bits from 0 to 1 or 1 to 0) their LSBs, preferably altering 3 bits. This manipulation is carefully designed to incorporate triggers into the images while maintaining their visual integrity. To the naked eye, the changes introduced by the LSB manipulation are incredibly subtle and difficult to detect. However, despite their inconspicuous nature, DNN models are remarkably adept at discerning and identifying the patterns embedded within these manipulated images. Subsequently, the newly manipulated images are reintroduced into the training dataset, alongside the original clean images. The DNN model is then trained using this combined dataset, which includes both the backdoor images with triggers and the clean images. By training the model on these backdoor images containing triggers, it becomes capable of distinguishing the differences between the backdoor and clean images based on the presence of triggers. Consequently, when a clean input image is fed into the trained model, it is expected to produce a benign output. However, when the input image contains the backdoor with triggers, the model is likely to generate an incorrect prediction due to the influence of the triggers. This methodology demonstrates how the injection of triggers into specific regions of training images, coupled with appropriate manipulation and model training, can introduce vulnerabilities in the DL model, leading to potential misclassifications or undesired behavior.

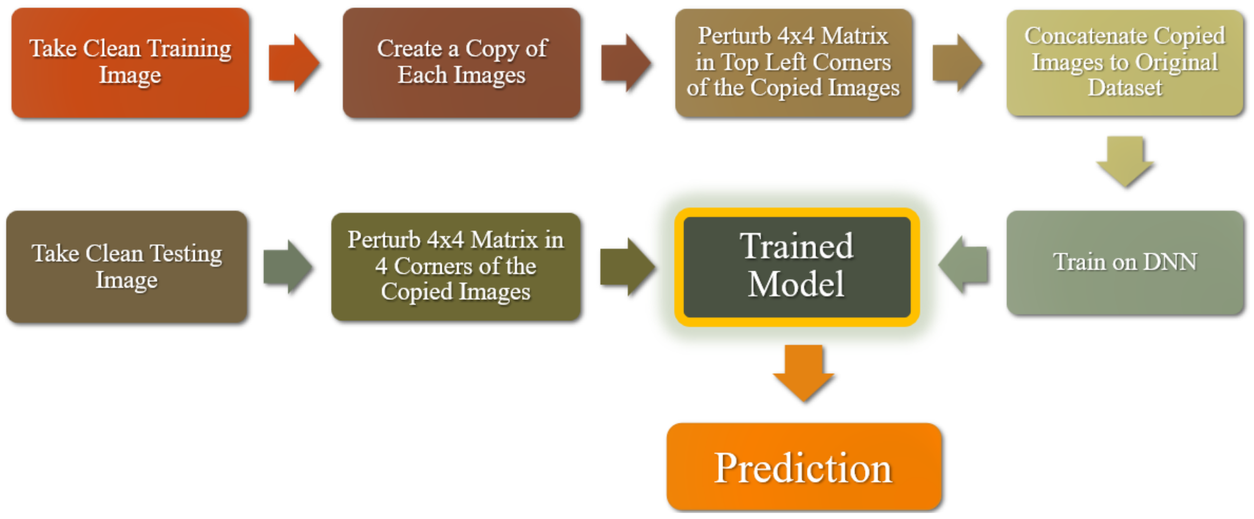

Take Clean Training Image
Create a Copy of Each Images
Perturb 4x4 Matrix in Top Left Corners of the Copied Images
Concatenate Copied Images to Original Dataset
Train on DNN
Take Clean Testing Image
Perturb 4x4 Matrix in 4 Corners of the Copied Images
Trained Model
Prediction

Figure 3.3: Flow diagram of LSB manipulation

## 3.4 Pixel Manipulation Methodology

The pixel manipulation attack is a type of adversarial attack that involves perturbing a specific set of pixels in an input image to induce misclassification. In our methodology, determining the optimal set of pixels that are most important to achieving this misclassification can be framed as an optimization problem. This optimization problem can be approached using two different approaches: targeted and untargeted.

### 3.4.1 Targeted Approach

In the targeted approach, the objective is to maximize the confidence or probability of a specific target class. By perturbing the selected pixels in a manner that maximizes the model's confidence in the target class, the aim is to intentionally misclassify the image as the desired target class.

### 3.4.2 Untargeted Approach

In the untargeted approach, the objective is to minimize the confidence or probability of the correct class predicted by the model. The selected pixels are modified in a way that reduces the model's confidence in the correct class, leading to misclassification into any other class.

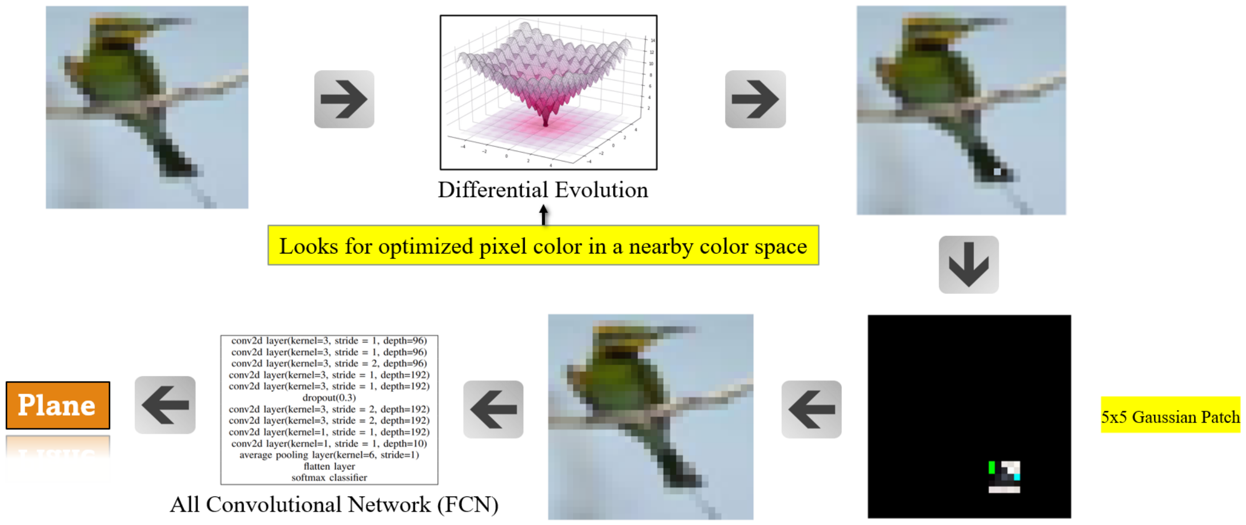

Differential Evolution
Looks for optimized pixel color in a nearby color space
5x5 Gaussian Patch
Plane
conv2d layer(kernel=3, stride = 1, depth=96)
conv2d layer(kernel=3, stride = 1, depth=96)
conv2d layer(kernel=3, stride = 2, depth=96)
conv2d layer(kernel=3, stride = 1, depth=192)
conv2d layer(kernel=3, stride = 1, depth=192)
dropout(0.3)
conv2d layer(kernel=3, stride = 2, depth=192)
conv2d layer(kernel=3, stride = 2, depth=192)
conv2d layer(kernel=1, stride = 1, depth=192)
conv2d layer(kernel=1, stride = 1, depth=10)
average pooling layer(kernel=6, stride=1)
flatten layer
softmax classifier
All Convolutional Network (FCN)

Figure 3.4: Flow diagram of pixel manipulation

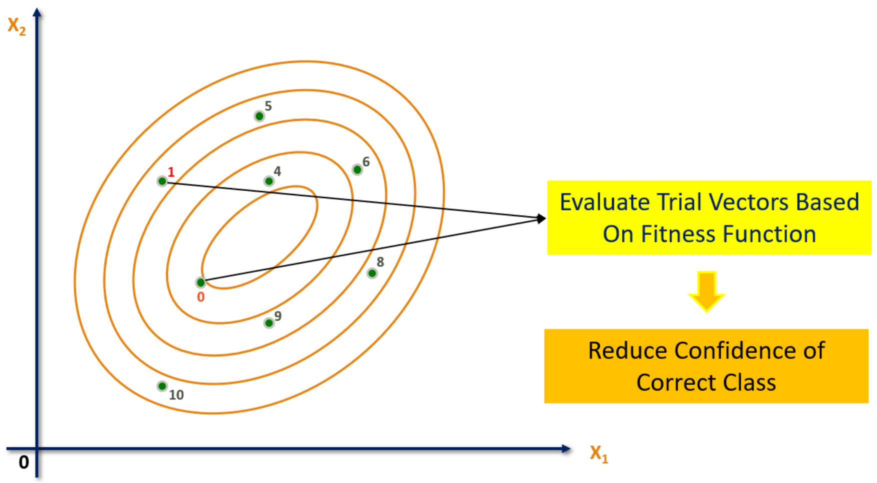

$X_2$
$X_1$
0
0
1
4
5
6
8
9
10
Evaluate Trial Vectors Based On Fitness Function
Reduce Confidence of Correct Class

Figure 3.5: Differential evolution algorithm

Both the targeted and untargeted approaches utilize optimization techniques to determine the optimal perturbations on the selected set of pixels. In our experiment, we opted for an untargeted attack approach by formulating it as an optimization problem. The primary objective is to identify the most effective pixel modifications that can deceive the model and lead to misclassification. These approaches highlight the different objectives an adversary may have when conducting a pixel manipulation attack. A very important and crucial part of Fig. 3.4 is the differential algorithm and how it works. Fig. 3.5 presents the working methodology of a standard differential evolution algorithm. In this particular optimization algorithm, we employed the differential evolution algorithm that aims to minimize a given function by iteratively generating and refining a population of candidate solutions. Each candidate solution is represented as a vector of real numbers, serving as inputs for the target function. In Fig. 3.5 the points 0,1,2,3,4,5,6,7,8,9,10 represent different candidate solutions from which trial vectors are found based on the fitness function. The fitness function is the condition based on which the optimization takes place. In our case, the optimization condition was to reduce the confidence of the correct class. The primary goal is to identify the set of inputs that produce the lowest possible output, indicating a better fitness score. The process begins with the initialization of a population of candidate vectors, typically done randomly. These vectors are evaluated by the function we seek to minimize, and their fitness scores are calculated accordingly. The lower the output of the function for a given vector, the higher its fitness in the population. To generate new offspring vectors, we employ a mutation process by combining individuals from the existing population. This step involves altering the candidate vectors to explore new areas of the search space. The offspring vectors then undergo fitness evaluation based on the target function. In each generation, the offspring competes with the existing population members. Poor-performing individuals are replaced by offspring vectors that exhibit superior fitness. This continuous cycle of generating, evaluating, and replacing candidates aims to progressively improve the population's overall fitness and, eventually, converge to an optimal solution. By employing this differential evolution algorithm, we can efficiently explore the search space, allowing us to discover better candidate solutions that lead to minimized function outputs, which represent an optimal outcome.

# CHAPTER IV

# RESULTS AND DISCUSSION

Our experiments encompass two main categories: LSB perturbation and pixel perturbation. Further, these categories can be subdivided based on three different datasets used for each experiment. The standard datasets employed in our research are as follows:

- *MNIST:* MNIST (Modified National Institute of Standards and Technology) is a widely used dataset for image classification tasks. It consists of 28x28 grayscale images of handwritten digits (0 to 9) and corresponding labels. The dataset contains 60,000 training images and 10,000 test images, making it a popular benchmark for evaluating machine learning algorithms and models.
- *Fashion MNIST:* Fashion MNIST is another image classification dataset similar to MNIST. It contains 28x28 grayscale images of various fashion items, such as shirts, dresses, shoes, and bags. Like MNIST, Fashion MNIST consists of 60,000 training images and 10,000 test images. This dataset is particularly valuable for testing image recognition models in the context of fashion-related applications.
- *CIFAR-10:* CIFAR-10 is a challenging dataset used for object recognition and classification tasks. It contains 60,000 32x32 color images across ten classes: airplanes, automobiles, birds, cats, deer, dogs, frogs, horses, ships, and trucks. Each class has 6,000 images, with 50,000 used for training and 10,000 for testing. CIFAR-10 is known for its small image size and complex object categories, making it a standard benchmark for evaluating the performance of deep learning models in image classification problems.

## 4.1 LSB Perturbation Results

To simulate the LSB perturbation attack we formulated 3 different DNN architectures for 3 different datasets to gain the maximum validation accuracies in each dataset. After that, the attacks were carried out to measure the attack metrics and attack successes.

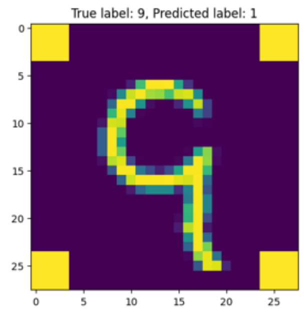
### 4.1.1 MNIST Results

The DNN architecture employed for conducting the LSB perturbation attack experiments on the MNIST dataset consisted of two convolutional layers followed by three dense layers.

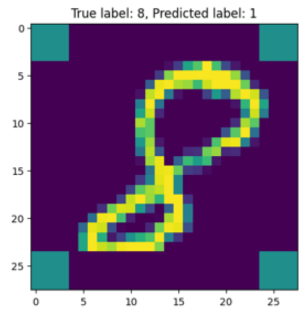
Table 4.1: DNN architecture of LSB manipulation attack using MNIST data

| **Layer Type** | **Input Shape** | **Output Shape** | **Filter Size** | **Stride** | **Activation** |
|---|---|---|---|---|---|
| Conv1 | 1x28x28 | 16x24x24 | 16x1x5x5 | 1 | ReLU |
| Pool1 | 16x24x24 | 16x12x12 | 2x2 | 2 | - |
| Drop1 | 16x12x12 | 16x12x12 | - | - | - |
| Conv2 | 16x12x12 | 32x8x8 | 32x16x5x5 | 1 | ReLU |
| Pool2 | 32x8x8 | 32x4x4 | 2x2 | 2 | - |
| Drop2 | 32x4x4 | 32x4x4 | - | - | - |
| Flat | 32x4x4 | 512 | - | - | - |
| Fc1 | 512 | 256 | 512x256 | - | ReLU |
| Drop3 | 256 | 256 | - | - | - |
| Fc2 | 256 | 512 | 256x512 | - | ReLU |
| Fc3 | 512 | 10 | 512x10 | - | Softmax |

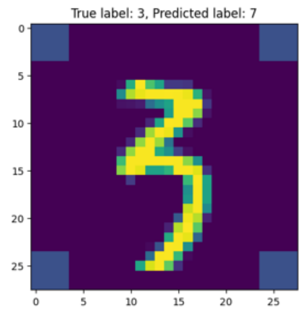
The complete details of the architecture are presented in Table 4.1. In our attack evaluation, we examined the impact of perturbing a range of least significant bits (LSBs) starting from 1 to 8 bits for the chosen Trigger of a 4x4 pixel area from the 4 corners of each image. Bear in mind, the result comparison of top-left corner and 4 corners are almost similar. So, for better understanding our experiment incorporated all 4 corners. As depicted in Fig. 4.1, we conducted experiments with various bit changes, ranging from 1 to 8 bits. The results revealed that attacks using 4 to 8 bits caused a significant visual alteration, easily detectable by the naked eye. Conversely, when applying only 1 to 3 bits of changes, the alterations were not noticeable to human observers. Initially, our experiments seemed to favor using 1-bit, 2-bits, and 3-bits of LSB perturbations. However, Fig. 4.2 displayed that for 1-bit and 2-bits of LSB perturbations, the validation accuracy dropped



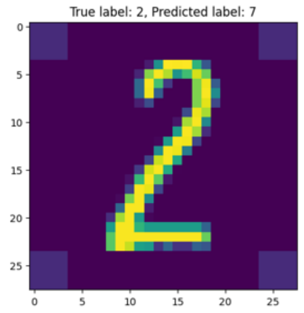
(a) 8 bits



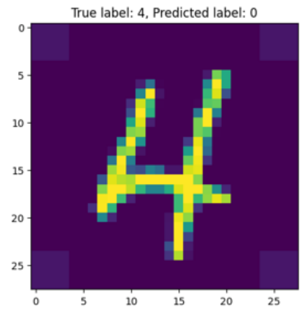
(b) 7 bits



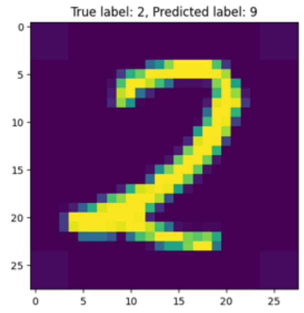
(c) 6 bits



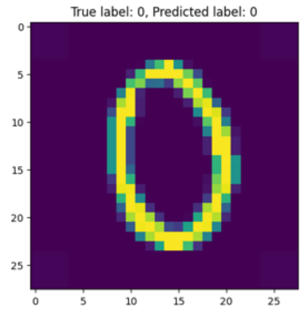
(d) 5 bits



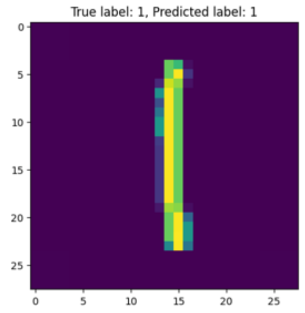
(e) 4 bits



(f) 3 bits



(g) 2 bits



(h) 1 bit

Figure 4.1: LSB perturbation experiments from 1 to 8 bits of MNIST

significantly compared to the baseline accuracy. The observed phenomenon can be attributed to the model's confusion when attempting to identify whether an image contains a trigger or not, leading to vague assumptions. In some cases, the model manages to recognize the triggers, while in others, it fails to validate their presence and assumes the image is clean. Consequently, this confusion creates a series of misclassifications, where clean images are sometimes mislabeled and images with triggers are sometimes incorrectly classified as clean. As a consequence, this confusion causes a significant drop in accuracy. Finally, we found that using 3-bits of LSB perturbations struck a balance between maintaining similar accuracy to the baseline and creating an attack that remained nearly imperceptible to human eyes. This observation made 3-bits LSB perturbation the optimal choice for our proposed method. To provide additional evidence of the minimal visual differences resulting from a 3-bit perturbation, we conducted a comparison in both the spatial and frequency domains. In Fig. 4.3a and 4.3b, we present spatial plots of the original image alongside the perturbed image. While the spatial domain can be less visually comparable, it still serves as an initial point of comparison. However, to gain more insights and enhance the comparison, we

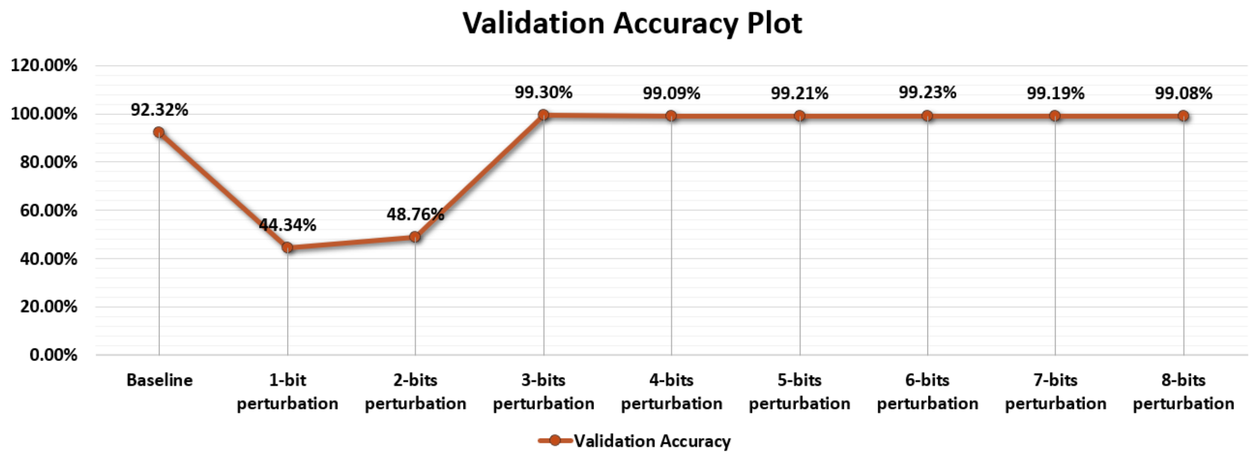
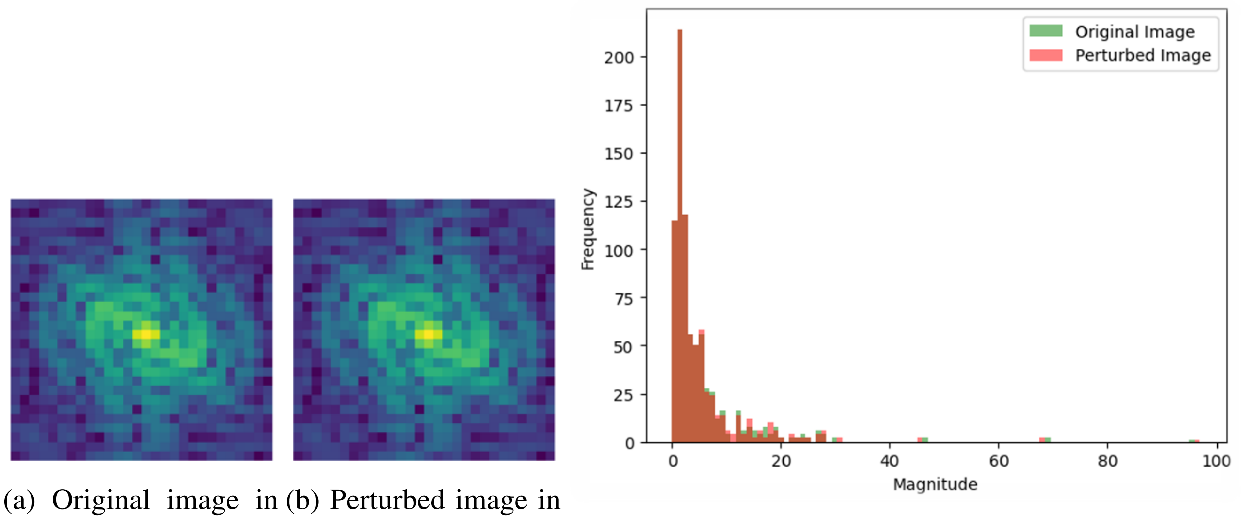


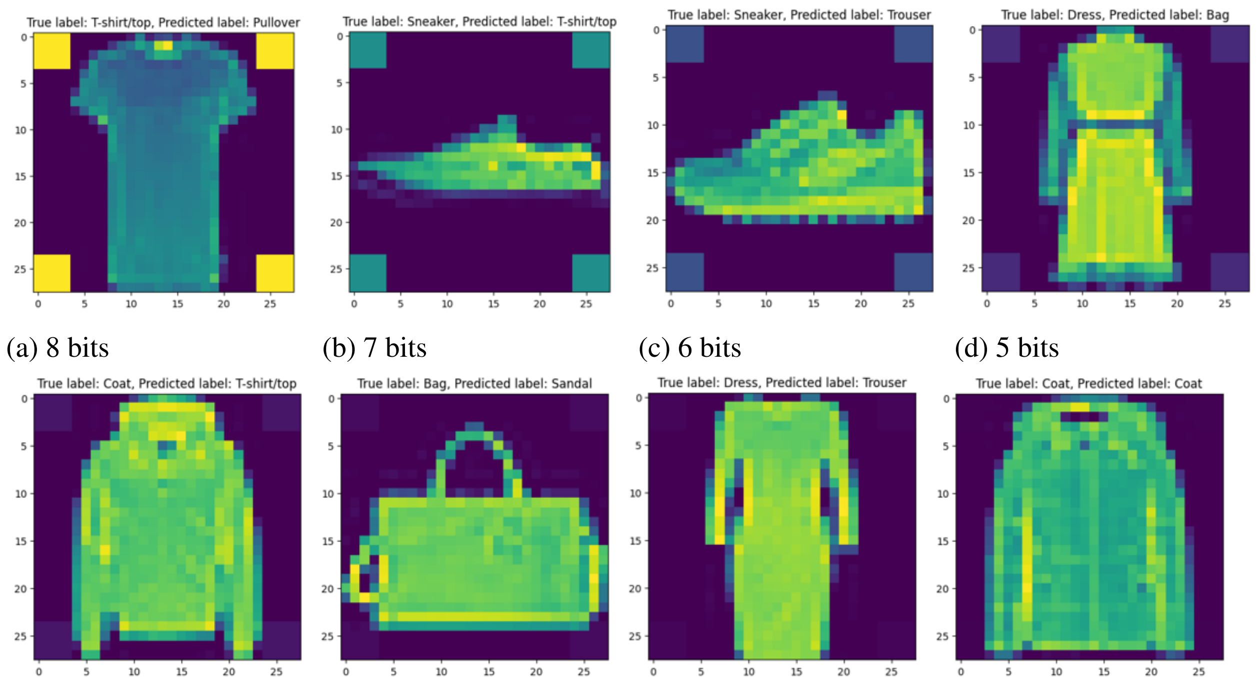
Figure 4.2: Validation accuracies of the perturbations (MNIST)



(a) Original image in spatial domain (b) Perturbed image in spatial domain (c) Difference between them in frequency domain

Figure 4.3: Visual differences between original and perturbed image

also assessed the images in the frequency domain, as shown in Fig. 4.3c. Analyzing the images in the frequency domain allows us to identify any significant differences or patterns that might not be apparent in the spatial representation. Even though we can visualize the minor changes in the frequency domain, still the changes are very minimal. By conducting this comprehensive analysis, we aim to establish a more thorough understanding of the visual impact of the 3-bit perturbation on the images.



(a) 8 bits (b) 7 bits (c) 6 bits (d) 5 bits

(e) 4 bits (f) 3 bits (g) 2 bits (h) 1 bit

Figure 4.4: LSB perturbation experiments from 1 to 8 bits of Fashion MNIST

### 4.1.2 Fashion MNIST Results

The DNN architecture employed for conducting the LSB perturbation attack experiments on the MNIST dataset consisted of three convolutional layers followed by two dense layers. The complete details of the architecture are presented in Table 4.2. In our attack evaluation, we examined the impact of perturbing a range of least significant bits (LSBs) starting from 1 to 8 bits for the chosen trigger of a 4x4 pixel area from the 4 corners of each image which is exactly similar to the previous experiment with MNIST dataset. As these two datasets are pretty similar in nature, we expected similar outcomes from this experiment as well. Based on the data presented in Fig. 4.4, it can be observed that visual changes are most noticeable when manipulating up to 4 bits. However, for 1, 2, and 3 bits, these alterations are hardly discernible to the naked eye. In contrast to our initial assumption, the evaluation of validation accuracies for different levels of bit alterations yielded surprising results. As visually depicted in the comprehensive Fig. 4.5, the baseline accuracy achieved an impressive 93.46%. However, the accuracy for the 2-bits perturbation unexpectedly

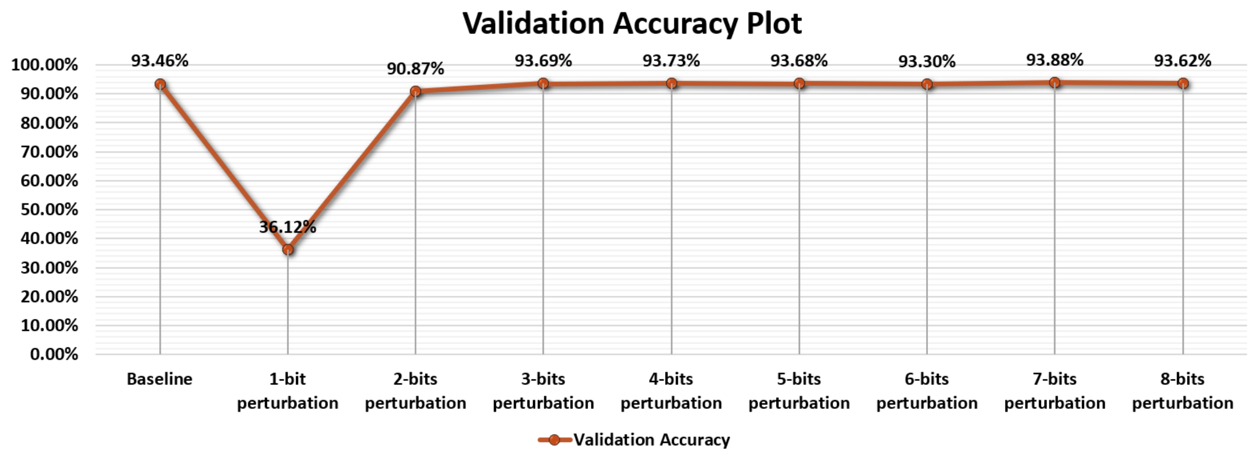


Figure 4.5: Validation accuracies of the perturbations (Fashion MNIST)

showed a smaller drop than we had anticipated, settling at 90.87%. On the other hand, the 1-bit perturbation's poor performance was in line with our expectations, resulting in an accuracy of only 36.12%. The change in validation accuracy between the baseline and the 2-bit perturbation

Table 4.2: DNN architecture of LSB manipulation attack using Fashion MNIST data

| Layer | Type | Input Shape | Output Shape | Filter Size |
|---|---|---|---|---|
| Conv1 | Convolution | 1x28x28 | 16x28x28 | 16x1x3x3 |
| Conv2 | Convolution | 16x28x28 | 32x28x28 | 32x16x3x3 |
| Pool1 | Pooling | 32x28x28 | 32x14x14 | 2x2 |
| Drop1 | Dropout | 32x14x14 | 32x14x14 | - |
| Conv3 | Convolution | 32x14x14 | 64x14x14 | 64x32x3x3 |
| Conv4 | Convolution | 64x14x14 | 64x14x14 | 64x64x3x3 |
| Pool2 | Pooling | 64x14x14 | 64x7x7 | 2x2 |
| Drop2 | Dropout | 64x7x7 | 64x7x7 | - |
| Flat | Flatten | 64x7x7 | 3136 | - |
| Fc1 | Fully Connected | 3136 | 512 | 3136x512 |
| Drop3 | Dropout | 512 | 512 | - |
| Fc2 | Fully Connected | 512 | 10 | 512x10 |

amounted to 2.77%, a significantly higher discrepancy compared to the mere 0.22% change observed between the baseline and the 3-bits perturbation. This substantial variation indicates that the owner of the model would likely detect the attack when a 2-bits perturbation is used, making the attack considerably more prone to failure. Hence, the optimal value for a successful bit perturbation, capable of causing the backdoor attack on the fashion MNIST dataset, is indeed 3 bits.

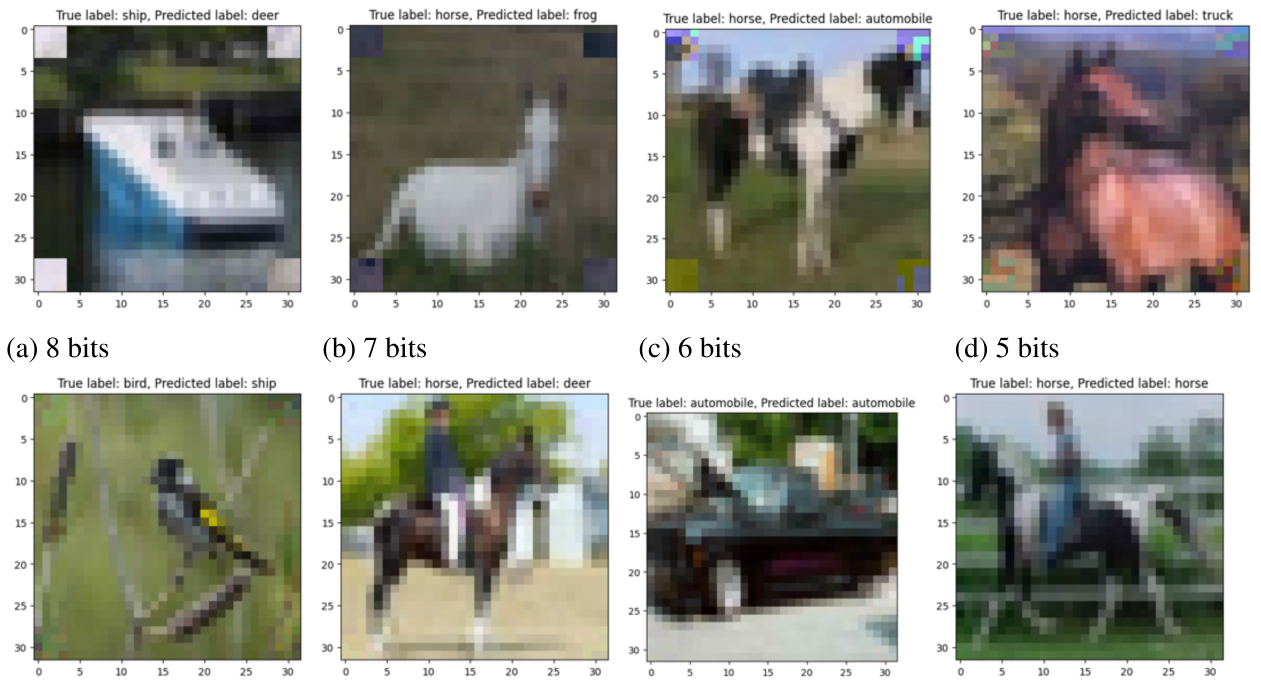
### 4.1.3 CIFAR-10 Results

Our previous experiments provided valuable insights into the effectiveness of the 3-bit perturbation backdoor attack. However, it is essential to acknowledge that the success of this attack was primarily demonstrated in scenarios where the working images had a monochromatic background. In such cases, setting the trigger with 3-bits perturbation appeared comparatively

Table 4.3: DNN architecture of LSB manipulation attack using CIFAR-10 data

| **Layer Type** | **Input Shape** | **Output Shape** | **Filter Size** | **Stride** | **Activation** |
|---|---|---|---|---|---|
| Conv1 | 32x32x3 | 32x32x32 | 3x3x3x32 | 1 | ReLU |
| Batch Norm1 | 32x32x32 | 32x32x32 | - | - | - |
| Conv2 | 32x32x32 | 32x32x32 | 3x3x32x32 | 1 | ReLU |
| Batch Norm2 | 32x32x32 | 32x32x32 | - | - | - |
| Pool1 | 32x32x32 | 16x16x32 | 2x2 | 2 | - |
| Drop1 | 16x16x32 | 16x16x32 | - | - | - |
| Conv3 | 16x16x32 | 16x16x64 | 3x3x32x64 | 1 | ReLU |
| Batch Norm3 | 16x16x64 | 16x16x64 | - | - | - |
| Conv4 | 16x16x64 | 16x16x64 | 3x3x64x64 | 1 | ReLU |
| Batch Norm4 | 16x16x64 | 16x16x64 | - | - | - |
| Pool2 | 16x16x64 | 8x8x64 | 2x2 | 2 | - |
| Drop2 | 8x8x64 | 8x8x64 | - | - | - |
| Conv5 | 8x8x64 | 8x8x128 | 3x3x64x128 | 1 | ReLU |
| Batch Norm5 | 8x8x128 | 8x8x128 | - | - | - |
| Conv6 | 8x8x128 | 8x8x128 | 3x3x128x128 | 1 | ReLU |
| Batch Norm6 | 8x8x128 | 8x8x128 | - | - | - |
| Pool3 | 8x8x128 | 4x4x128 | 2x2 | 2 | - |
| Drop3 | 4x4x128 | 4x4x128 | - | - | - |
| Flat | 4x4x128 | 2048 | - | - | - |
| Fc1 | 2048 | 128 | 2048x128 | - | ReLU |
| Batch Norm7 | 128 | 128 | - | - | - |
| Drop4 | 128 | 128 | - | - | - |
| Fc2 | 128 | 10 | 128x10 | - | Softmax |

easier, leading to successful backdoor attacks. However, it is crucial to recognize that real-world scenarios are often more complex, and backgrounds in images may not be monochromatic. To address this limitation and assess the attack's performance in a more realistic setting, we conducted experiments using the CIFAR-10 dataset. It consists of diverse images with varying backgrounds, closely simulating real-world conditions. With CIFAR-10, the challenge lies in the model's ability to


True label: ship, Predicted label: deer
True label: horse, Predicted label: frog
True label: horse, Predicted label: automobile
True label: horse, Predicted label: truck
True label: bird, Predicted label: ship
True label: horse, Predicted label: deer
True label: automobile, Predicted label: automobile
True label: horse, Predicted label: horse

(a) 8 bits (b) 7 bits (c) 6 bits (d) 5 bits

(e) 4 bits (f) 3 bits (g) 2 bits (h) 1 bit

Figure 4.6: LSB perturbation experiments from 1 to 8 bits of CIFAR-10

correctly identify the trigger amidst a random and diverse background. The complexity introduced by the background variability can potentially pose difficulties for the model in detecting and activating the trigger accurately. In our series of experiments, we investigated the use of different bit perturbations, ranging from 1 to 8 bits, to find the most suitable configuration for achieving a balance between covertness and attack efficiency. The results, illustrated in Fig. 4.7, revealed some interesting findings. For 1-bit and 2-bit perturbation, we observed an alarmingly low validation accuracy, with only 50.51% accuracy for 1-bit perturbation and 42.87% for 2-bit perturbation. In nearly half of the cases, the backdoored images went undetected, resulting in the failure of the attack. These perturbation levels proved to be ineffective in concealing the presence of the backdoor. However, as we increased the perturbation to 3 bits and beyond, we noticed a significant improvement in the validation accuracy, with results closely resembling the baseline accuracy. This implies that 3 bits to 8 bits perturbation offer a more promising approach to successful attacks, as they are less prone to detection. Among the range of perturbation levels tested, we found that 4 to 8 bits perturbation, while effective in terms of accuracy, led to conspicuous alterations in the images.

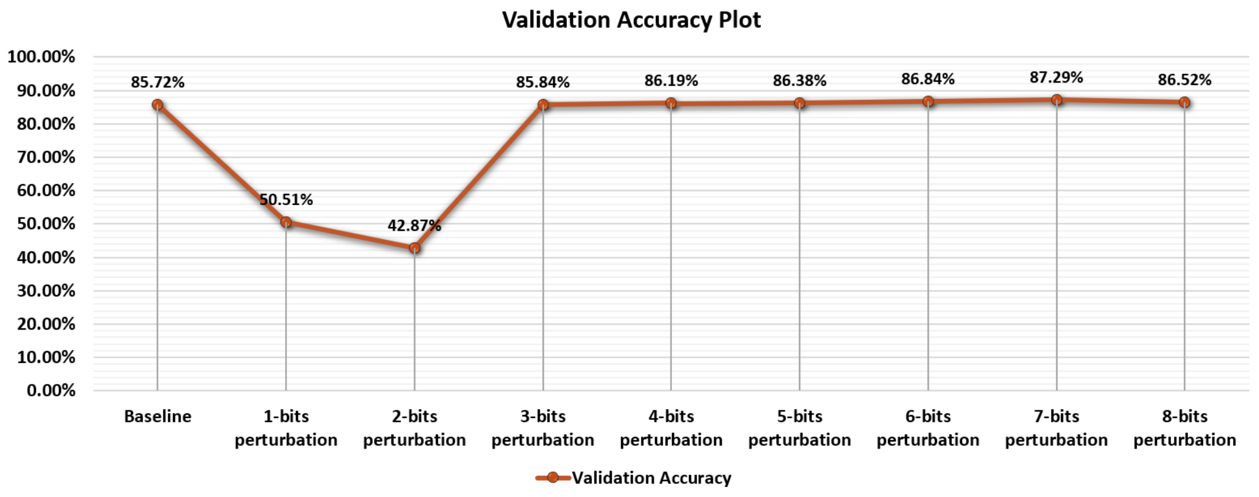


Figure 4.7: Validation accuracies of the perturbations (CIFAR-10)

This could raise suspicions and potentially make the attack more vulnerable to detection. On the other hand, the 3-bit perturbation demonstrated an optimal balance between attack effectiveness and covert operation. It managed to achieve high accuracy similar to the baseline while maintaining a subtle manipulation that was less noticeable, thereby making it a more suitable LSB manipulation technique for our third experiment with the backdoor attack.

### 4.1.4 Additional Experiment Results

In addition to our main experiment, we conducted two supplementary experiments on the training datasets to explore the upper and lower limits of the number of backdoor images within the entire training dataset. The objective was to understand the impact of varying backdoor image quantities on the model's performance. In the first experiment, we investigated the upper limit scenario, where we introduced a relatively high number of backdoor images into the training dataset. The hypothesis was that an abundance of backdoor images might lead the model to confuse clean images with backdoor images, potentially compromising its accuracy and generalization. The results of the experiment are presented in Table 4.4, indicating the impact of varying the percentage of backdoor images on the test accuracy with CIFAR-10 dataset. Notably, the test accuracy experiences a substantial drop once the percentage of backdoor images exceeds 50%. However, when the percentage of backdoor images is limited to at most 20% of the total training

Table 4.4: Impact of the percentage of backdoor image on test accuracy

| **Training Dataset** | **Validation Dataset** | **% of Backdoor Images** | **Test Accuracy** (%) |
|---|---|---|---|
| 45,000 | 5,000 | 50% | 73.12 |
| 45,000 | 5,000 | 40% | 80.44 |
| 45,000 | 5,000 | 30% | 80.46 |
| 45,000 | 5,000 | 20% | 83.61 |
| 45,000 | 5,000 | 10% | 84.57 |
| 45,000 | 5,000 | 0% | 85.53 |
| 40,000 | 10,000 | 50% | 42.38 |
| 40,000 | 10,000 | 40% | 69.51 |
| 40,000 | 10,000 | 30% | 79.50 |
| 40,000 | 10,000 | 20% | 82.86 |
| 40,000 | 10,000 | 10% | 83.98 |
| 40,000 | 10,000 | 0% | 84.64 |

dataset, the accuracy remains relatively close to the baseline accuracy. This finding suggests that maintaining a low proportion of backdoor images in the training dataset is crucial to preserving the model's performance and minimizing suspicious drops in the model's accuracy. Conversely, in the second experiment, we explored the lower-limit scenario by incorporating a very limited number of backdoor images into the training dataset. The underlying assumption here was that with only a few backdoor images, the model might struggle to differentiate between backdoor images and clean images, possibly misclassifying backdoor images as clean examples. Table 4.5 displays the outcomes of the second experiment conducted on three datasets. Notably, the model's performance deteriorates when the percentage of backdoor samples falls below 10% of the entire training dataset. This finding aligns with the claim made by [6], suggesting that the lower limit for effective backdoor samples is 10%. From the results, it becomes evident that maintaining at least 10% backdoor samples in the training dataset is crucial to ensure the model's robustness against backdoor attacks. When the proportion of backdoor samples decreases below this threshold, the model's ability to identify the backdoor trigger diminishes, leading to a considerable decline in performance.

Table 4.5: Impact of backdoor samples on model performance

| Dataset | Training Samples | % of Backdoor Samples | Trigger Not Identified (%) |
|---|---|---|---|
| MNIST | 54,000 | 3% | 83.14% |
| MNIST | 54,000 | 4% | 2.16% |
| MNIST | 54,000 | 5% | 1.21% |
| MNIST | 54,000 | 7% | 1.14% |
| MNIST | 54,000 | 10% | 0.20% |
| Fashion MNIST | 54,000 | 3% | 93.41% |
| Fashion MNIST | 54,000 | 4% | 1.47% |
| Fashion MNIST | 54,000 | 5% | 1.31% |
| Fashion MNIST | 54,000 | 7% | 1.02% |
| Fashion MNIST | 54,000 | 10% | 0.92% |
| CIFAR-10 | 45,000 | 3% | 85.07% |
| CIFAR-10 | 45,000 | 4% | 80.87% |
| CIFAR-10 | 45,000 | 5% | 83.61% |
| CIFAR-10 | 45,000 | 7% | 82.55% |
| CIFAR-10 | 45,000 | 10% | 4.69% |

### 4.1.5 Visual Comparison: Badnet vs Our Proposed Method

Our proposed method has demonstrated remarkable resilience against naked-eye inspection, achieving an accuracy comparable to the baseline model while requiring fewer bit alterations. To illustrate the effectiveness of our approach in comparison to the state-of-the-art method (badnet), we provide a visual comparison in Fig. 4.8. When inspecting the images with the naked eye, it is evident that it is relatively easy to differentiate between the clean image and a backdoor image generated using the badnet method. This is mainly due to the conspicuous pixel alterations introduced by badnet. In contrast, our method strategically replaces only 3 bits, resulting in alterations that are barely recognizable by the naked eye. The imperceptibility of these alterations underscores the efficiency of our approach in concealing the presence of the backdoor, making it far more covert and difficult to detect visually. Overall, the visual comparison validates the effectiveness of our proposed method, as it achieves both high covertness and model accuracy, with minimal perceptible changes to the images.

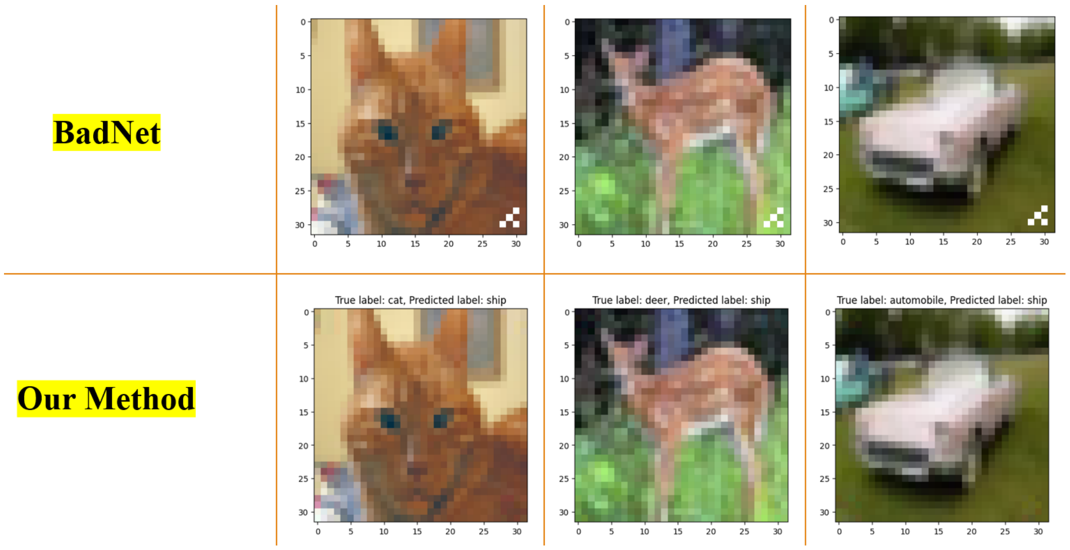

BadNet
Our Method
True label: cat, Predicted label: ship
True label: deer, Predicted label: ship
True label: automobile, Predicted label: ship

Figure 4.8: Visual comparison between our method and badnet

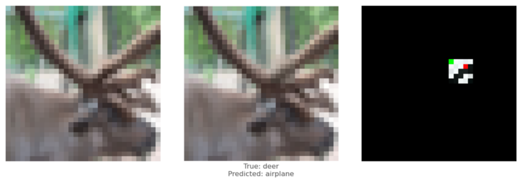
## 4.2 Pixel Perturbation Results

In the pixel perturbation experiment, we evaluated the performance of two standard deep-learning models: LeNet and ResNet. These models were utilized to gather the results and assess their respective capabilities. Table 4.6 provides a brief overview of the salient characteristics of these two models, including the number of parameters and the achieved accuracy. In our pixel

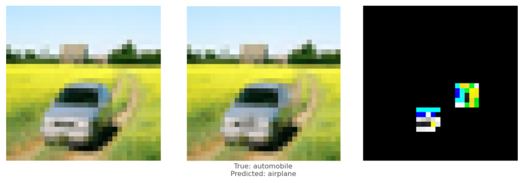
Table 4.6: Performance comparison of DL models

| DL Model | Number of Parameters | Accuracy |
|---|---|---|
| LeNet | 62,006 | 74.88% |
| ResNet | 470,218 | 92.31% |

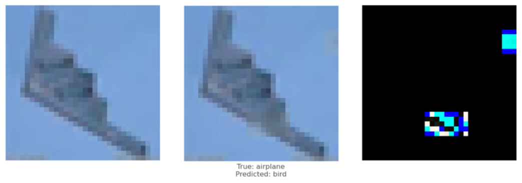
perturbation attack, we adopted a more comprehensive approach instead of concentrating solely on perturbing individual pixels and using a 5x5 Gaussian patch to conceal them within the image. To achieve a more effective and efficient attack strategy, we conducted experiments with different numbers of perturbed pixels. Specifically, we explored configurations involving 1, 2, 3, 4, 5, 7, 10, 15, and 20 pixels, and meticulously collected results for each of these experiments. By varying the number of perturbed pixels, we aimed to explore a wide range of possibilities and assess their impact on the success of the pixel perturbation attack. This comprehensive analysis enabled us to determine the optimal number of pixels to perturb, striking the right balance between achieving



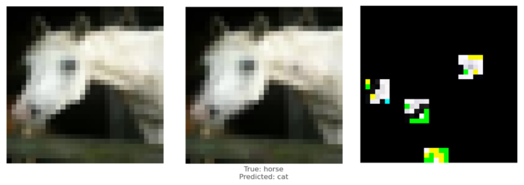
(a) 1 pixel



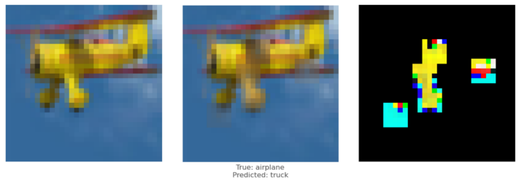
(b) 2 pixels



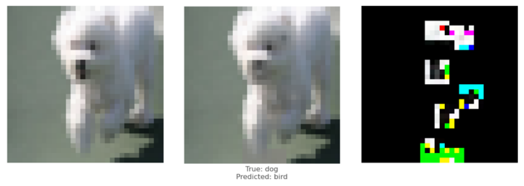
(c) 3 pixels



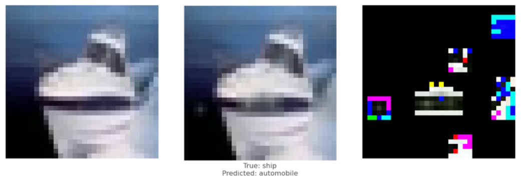
(d) 4 pixels



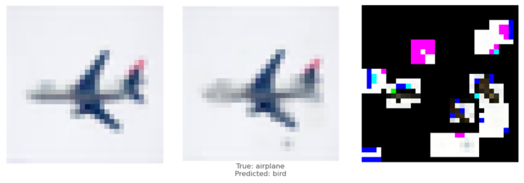
(e) 5 pixels



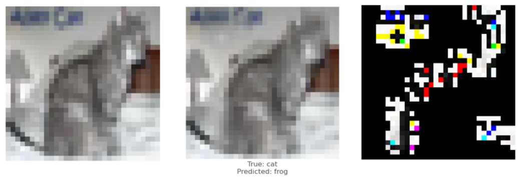
(f) 7 pixels



(g) 10 pixels



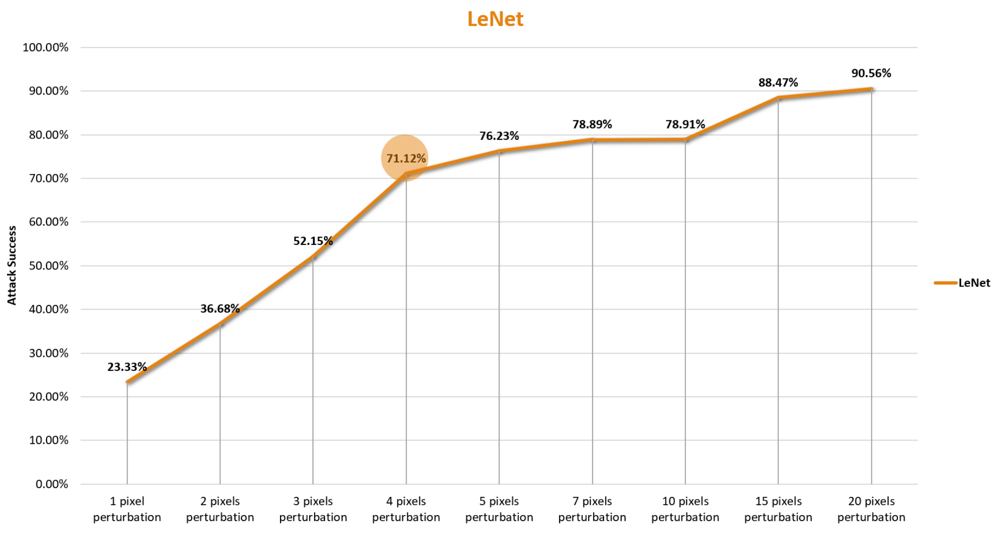
(h) 15 pixels



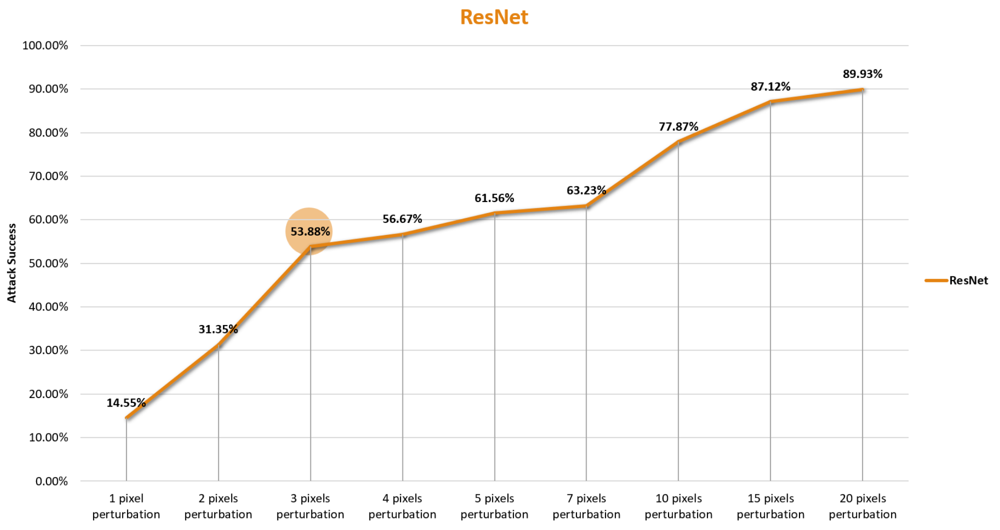
(i) 20 pixels

Figure 4.9: Pixel perturbation experiments from 1 to 8 pixels of CIFAR-10

covertness and ensuring the attack's efficiency. In our pixel perturbation experiments, our primary focus lies in achieving two key objectives: high accuracy in the attack and maintaining covertness. To assess the impact of perturbations on image appearance, we compare the perturbed images with their original counterparts (see Fig. 4.9). It becomes evident that visual differences are highly subjective, especially without a reference image, which is the original image in our case. We observe that when the perturbations involve up to 5 pixels, they are less likely to be visually recognized, particularly when compared to the original image. This indicates that minor perturbations preserve



(a) Attack success rate using LeNet



(b) Attack success rate using ResNet

Figure 4.10: Performance of LeNet & ResNet using different pixel perturbations

the attack's covertness, making the differences imperceptible to the human eye. However, as we increase the number of perturbed pixels to 15 and 20, the visual distinctions become more apparent, rendering these perturbations distinctly visible compared to the original image. Such a level of perturbation may compromise the covertness of the attack, making it more susceptible to detection. We also evaluated the impact of 7 and 10-pixel perturbations, which fall into a somewhat ambiguous range. At times, they can be recognizable, while at other times, they remain inconspicuous. The effectiveness of these perturbations in maintaining covertness might vary depending on the specific

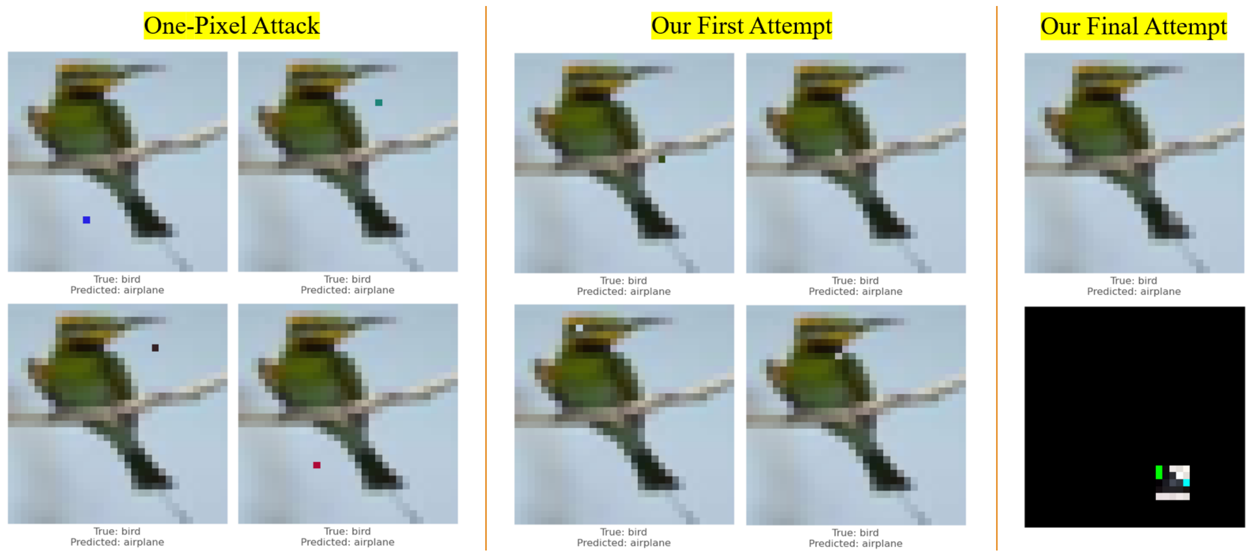


Figure 4.11: Visual comparison between our method and one-pixel attack

image and its visual characteristics. As a result, we can initially choose a perturbation size ranging from 1 to 5 pixels for our attack to balance covertness. However, we must also consider the attack success rate of the model. Fig. 4.10a demonstrates that the attack success is extremely high when using 20-pixel perturbations. Nevertheless, we have previously established that such large perturbations are vulnerable to visual recognition, compromising covertness. Thus, we are faced with a tradeoff between attack success and the covertness of the attack. Upon analyzing both criteria, we found that using 4 pixels for perturbation strikes a favorable balance, offering both covertness and a relatively higher success rate of 71.12% in the attack. Regarding ResNet, the results depicted in Fig. 4.10b indicate that perturbations involving 3, 4, and 5 pixels hold relatively more promise in achieving a balance between covertness and attack success rate compared to perturbations using 10, 15, and 12 pixels, despite the latter group exhibiting higher attack success rates.

### 4.2.1 Visual Comparison: One-Pixel Attack vs Our Method

In contrast to the state-of-the-art one-pixel attack, our approach primarily emphasizes imperceptibility and attack success rate. The imperceptibility of our method is evident from Fig. 4.11, where the left-most images represent results from the one-pixel attack, the middle images correspond to our initial experiment using the proposed method, and the right-most images display our final attempt along with the differences between the final attempt and the original image. In these comparisons, it becomes clear that our method achieves a higher level of imperceptibility when compared to the one-pixel attack. The left-most images, representing the one-pixel attack, often show noticeable, vibrant pixels that can be detected by the human eye. On the other hand, the middle images show a similar pixel, just like the one-pixel attack, but the color domain remains within that of the surrounding pixel colors. Lastly, the right-most images from our proposed method demonstrate significantly reduced visual differences upon using a 5x5 gaussian patch on our first attempt, making the perturbations less conspicuous and maintaining a higher level of imperceptibility.

# CHAPTER V

# CONCLUSION

This research presented two methodologies, LSB perturbation and pixel manipulation, to invade the integrity of deep neural networks through backdoor and adversarial attacks. Extensive experiments were conducted using the MNIST, Fashion MNIST, and CIFAR-10 datasets to evaluate the effectiveness of the proposed techniques. The results demonstrated that LSB perturbation of 3 bits achieved an optimal balance between attack success rate and covertness across all three datasets. By flipping only 3 LSBs, the modifications remained virtually imperceptible, while still inducing the desired misclassifications in the presence of the trigger. Comparative analysis showed improved stealth over the Badnets approach. For pixel manipulation, perturbations involving 4-5 pixels exhibited the best tradeoff between attack effectiveness and visual concealment. Comparisons with the One-Pixel attack revealed greater imperceptibility of the perturbations generated through our methodology. Overall, the proposed LSB perturbation and pixel manipulation methods were successful in invading the integrity of deep learning models by triggering misclassifications through subtle input manipulations. The techniques proved adept at evading detection, showcasing the vulnerabilities of DNNs against backdoor and adversarial attacks. Further research on defensive strategies such as attack detection, input validation, and adversarial training is imperative to safeguard deep learning systems against such integrity attacks. Exploring the use of emerging technologies like blockchain and differential privacy to prevent and mitigate training data poisoning also presents a promising direction for future work.

In conclusion, this research provided novel techniques to launch covert backdoor and adversarial attacks, while also highlighting crucial research gaps in securing deep learning models.

## REFERENCES

[1] Y. Cao, D. Xu, X. Weng, Z. Mao, A. Anandkumar, C. Xiao, and M. Pavone, *Robust trajectory prediction against adversarial attacks*, arXiv preprint arXiv:2208.00094, (2022).

[2] H. Chen, C. Fu, J. Zhao, and F. Koushanfar, *Deepinspect: A black-box trojan detection and mitigation framework for deep neural networks.*, in IJCAI, vol. 2, 2019, p. 8.

[3] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, and T. Ristenpart, *Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing*, in 23rd {USENIX} Security Symposium ({USENIX} Security 14), 2014, pp. 17–32.

[4] I. J. Goodfellow, J. Shlens, and C. Szegedy, *Explaining and harnessing adversarial examples*, arXiv preprint arXiv:1412.6572, (2014).

[5] T. Gu, B. Dolan-Gavitt, and S. Garg, *Badnets: Identifying vulnerabilities in the machine learning model supply chain*, arXiv preprint arXiv:1708.06733, (2017).

[6] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg, *Badnets: Evaluating backdooring attacks on deep neural networks*, IEEE Access, 7 (2019), pp. 47230–47244.

[7] M. Javaheripi, J.-W. Chang, and F. Koushanfar, *Acchashtag: Accelerated hashing for detecting fault-injection attacks on embedded neural networks*, ACM Journal on Emerging Technologies in Computing Systems, 19 (2022), pp. 1–20.

[8] M. Khosravy, K. Nakamura, Y. Hirose, N. Nitta, and N. Babaguchi, *Model inversion attack by integration of deep generative models: Privacy-sensitive face generation from a face recognition system*, IEEE Transactions on Information Forensics and Security, 17 (2022), pp. 357–372.

[9] M. Khosravy, K. Nakamura, N. Nitta, and N. Babaguchi, *Deep face recognizer privacy attack: Model inversion initialization by a deep generative adversarial data space discriminator*, in 2020 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), IEEE, 2020, pp. 1400–1405.

[10] Y. Kim, R. Daly, J. Kim, C. Fallin, J. H. Lee, D. Lee, C. Wilkerson, K. Lai, and O. Mutlu, *Flipping bits in memory without accessing them: An experimental study of dram disturbance errors*, ACM SIGARCH Computer Architecture News, 42 (2014), pp. 361–372.

[11] J. Li, A. S. Rakin, Y. Xiong, L. Chang, Z. He, D. Fan, and C. Chakrabarti, *Defending bit-flip attack through dnn weight reconstruction*, in 2020 57th ACM/IEEE Design Automation Conference (DAC), IEEE, 2020, pp. 1–6.

[12] S. Li, M. Xue, B. Z. H. Zhao, H. Zhu, and X. Zhang, *Invisible backdoor attacks on deep neural networks via steganography and regularization*, IEEE Transactions on Dependable and Secure Computing, 18 (2020), pp. 2088–2105.

[13] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, *Practical black-box attacks against machine learning*, in Proceedings of the 2017 ACM on Asia conference on computer and communications security, 2017, pp. 506–519.

[14] S. Pathak, S. A. Islam, H. Jiang, L. Xu, and E. Tomai, *A survey on security analysis of amazon echo devices*, High-Confidence Computing, (2022), p. 100087.

[15] A. S. Rakin and D. Fan, *Defense-net: Defend against a wide range of adversarial attacks through adversarial detector*, in 2019 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), IEEE, 2019, pp. 332–337.

[16] A. S. Rakin, Z. He, and D. Fan, *Bit-flip attack: Crushing neural network with progressive bit search*, in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 1211–1220.

[17] ——, *Tbt: Targeted neural network attack with bit trojan*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 13198–13207.

[18] S. Sengupta, T. Chakraborti, and S. Kambhampati, *Mtdeep: boosting the security of deep neural nets against adversarial attacks with moving target defense*, in Decision and Game Theory for Security: 10th International Conference, GameSec 2019, Stockholm, Sweden, October 30–November 1, 2019, Proceedings 10, Springer, 2019, pp. 479–491.

[19] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, *Membership inference attacks against machine learning models*, in 2017 IEEE symposium on security and privacy (SP), IEEE, 2017, pp. 3–18.

[20] J. Su, D. V. Vargas, and K. Sakurai, *One pixel attack for fooling deep neural networks*, IEEE Transactions on Evolutionary Computation, 23 (2019), pp. 828–841.

[21] A. Tatar, R. K. Konoth, E. Athanasopoulos, C. Giuffrida, H. Bos, and K. Razavi, *Throwhammer: Rowhammer attacks over the network and defenses*, in 2018 {USENIX} Annual Technical Conference ({USENIX}{ATC} 18), 2018, pp. 213–226.

[22] A. Tauhid, M. Tasnim, S. A. Noor, N. Faruqui, and M. A. Yousuf, *A secure image steganography using advanced encryption standard and discrete cosine transform*, Journal of Information Security, 10 (2019), pp. 117–129.

[23] A. Tauhid, L. Xu, M. Rahman, and E. Tomai, *A survey on security analysis of machine learning-oriented hardware and software intellectual property*, High-Confidence Computing, (2023), p. 100114.

[24] L. Tian, *A zero-watermarking method to protect intellectual property under strong geometric attacks*, in 2017 2nd International Conference on Multimedia and Image Processing (ICMIP), IEEE, 2017, pp. 172–176.

[25] D. Wang, C. Li, S. Wen, S. Nepal, and Y. Xiang, *Man-in-the-middle attacks against machine learning classifiers via malicious generative models*, IEEE Transactions on Dependable and Secure Computing, 18 (2020), pp. 2074–2087.

[26] Z. Xu, F. Yu, C. Liu, and X. Chen, *Lancex: A versatile and lightweight defense method against condensed adversarial attacks in image and audio recognition*, ACM Transactions on Embedded Computing Systems, 22 (2022), pp. 1–24.

[27] Z. Yang, J. Zhang, E.-C. Chang, and Z. Liang, *Neural network inversion in adversarial setting via background knowledge alignment*, in Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, 2019, pp. 225–240.

## BIOGRAPHICAL SKETCH

Ashraful Tauhid graduated with a Master of Science in Computer Science from the University of Texas Rio Grande Valley (UTRGV) in August 2023. His academic journey at UTRGV began in the fall of 2021 as a recipient of the highly esteemed Presidential Graduate Research Assistantship (PGRA) provided by the Graduate College of UTRGV. During his first year at UTRGV, he conducted research in the intersection of deep learning and privacy under the guidance of Dr. Sheikh Ariful Islam, the PI of the Secure System Design Lab at UTRGV. However, due to the absence of his supervisor in his second year, he was supervised by Dr. Emmett Tomai until he graduated. He published a review paper in the High-Confidence Computing journal in February 2023 under the guidance of both professors. Along with his research work, he has also served as a teaching assistant for various courses in the computer science department.

Upon completion of his degree, he will be joining a Doctor of Philosophy in Computer Science program starting in fall 2023. Tauhid's research interests lie in data science, machine learning and AI for privacy and security, and AI ethics and safety.

Tauhid was born and raised in Bangladesh and earned his Bachelor of Science in Information and Communication Engineering from the Bangladesh University of Professionals (BUP). Between his studies at UTRGV and BUP, he was accepted into the Institute of Business Administration at the University of Dhaka to pursue an MBA degree, which he later withdrew from to pursue computer science. For any further information or assistance, Tauhid can be reached at ashtauhid@gmail.com.