Data Poisoning attacks in Machine Learning: Risks and Defenses

Choeun Song

Cyber Security

Master's Thesis

June 3, 2025

AALBORG UNIVERSITY
DENMARK

AALBORG UNIVERSITY
STUDENT REPORT

Department of Electronic Systems
Cyber Security
Aalborg University
http://www.aau.dk

Title:

Data Poisoning attacks in Machine Learning: Risks and Defenses

Theme:

Master Thesis

Project Period:

February 2025 -
June 2025

Student number:

20231232

Participant(s):

Choeun Song

Supervisor(s):

Qiongxiu Li

Page Numbers:

53

Date of Completion:

June 3, 2025

Abstract:

The rise of machine learning in important areas has led to new security risks, such as backdoor data poisoning attacks, where hidden triggers in training data cause models to misclassify certain inputs while still performing well on normal data. This paper investigates the effectiveness of two types of backdoor triggers, visible (white square) and imperceptible (noise-based), in image classification models. Experiments show that both types can reliably fool models, with noise-based triggers being harder to detect. As defense strategy, we evaluate fine-tuning, retraining compromised models on a small set of triggered data with true label. Our results demonstrate that fine-tuning can significantly reduce the impact of backdoor attacks without harming model performance on regular inputs, although it is only effective when the type of trigger is already known. These findings highlight the ongoing need for strong defenses as machine learning becomes more widely used.

Keywords: Data poisoning, blackbox machine learning, backdoor attack

Contents

1 Introduction

The rapid adoption of machine learning (ML) across industries, from healthcare and finance to autonomous systems and cyber security, has revolutionized decision-making processes. However, this widespread deployment has also introduced critical vulnerabilities in cyber security and data integrity. Machine learning models, particularly those trained on decentralized or heterogeneous data sources (e.g., federated learning systems), are increasingly susceptible to data poisoning attacks. In data poisoning attacks, adversaries inject malicious data into training datasets to manipulate model behavior, often with devastating consequences. Among these threats, backdoor attacks stand out as a particularly subtle form of poisoning, where adversaries embed hidden triggers into training data to induce targeted misbehavior in models while maintaining normal performance on clean inputs.

Backdoor data poisoning attacks allow adversaries to secretly manipulate a model's behavior by embedding specific patterns, or triggers, into training data. This means a model can behave normally on clean inputs but will misclassify any input containing the trigger, often in a way that is difficult to detect. These attacks exploit the inherent trust in training data, allowing adversaries to compromise models in ways that are difficult to detect and mitigate. The growing sophistication of these attacks, especially with the emergence of both visible and invisible triggers, highlights a critical gap in our understanding of how to systematically evaluate and defend against them. This problem is highly significant, as it can lead to security breaches, financial loss, or even safety hazards in real-world applications.

This thesis addresses these challenges by conducting a systematic study of backdoor data poisoning attacks on image classifiers, using CIFAR-10 dataset.

Specifically, it investigates the effectiveness of both visible (white square) and invisible (noise-based) triggers and evaluates the fine-tuning defense for each attack scenario. The objectives of this work are: (1) to compare the success rates and stealthiness of different trigger types, (2) evaluate how these attacks influence the model under different conditions, such as varying the portion of poisoned training data, (3) to assess the practicality and limitations of fine-tuning as a defense, and (4) to provide insights and recommendations for building more robust machine learning systems.

The remainder of this paper is organized as follows: Chapter 2 reviews background research on machine learning functions, architectures, backdoor attacks, and trigger mechanisms. Chapter 3 reviews related works of data poisoning, comparing the recent works on different triggers. Chapter 4 describes the experimental methodology, including dataset preparation, model architectures, and evaluation metrics. Chapter 5 presents the results of the two attack scenario and finally, Chapter 6 includes the method and the result of defense experiments and real-life scenarios of both types of trigger.

2 Background Research

This section provides a fundamental understanding architecture of Machine learning, DNNs, CNNs and data poisoning, focusing on backdoor attacks related to the research topic.

2.1 Machine Learning

Machine Learning (ML), a branch of Artificial Intelligence (AI), focuses on computation that imitates the way human learns and makes decisions based on experience to perform tasks autonomously, fast, and continuously learning through more data and experience [1]. A key strength of ML is its ability to generalize, applying knowledge from training data to unseen data, making it versatile for solving diverse problems. The key benefit of ML algorithms is the ability to process large amount of data for identifying and extracting patterns and trends, thereby allowing to build systems that adapt to a changing environment with little or no human intervention [2].

Comparing with traditional problem-solving with computers which involves creating a program $\mathcal{M}$ that, given some input x, produces an output y as the solution to a problem which can be represented as a function $\mathcal{M} : X \to Y$, mapping an input set X to an output set Y, a key strength of Machine Learning is its ability to generalize, applying knowledge from training data to unseen data, making it versatile for solving diverse problems. For instance, given a dataset Z capturing relationships in the input space X, ML constructs a parametric model $\mathcal{M}_\theta : X \to Y$, where θ represents the model's parameters. At the core of ML processes lies an algorithm $\mathcal{A}$, which trains the model on Z or extracted features from Z, producing the final model $\mathcal{M}_\theta$ [2]. As an example, in a spam detection example, the dataset Z might include collection of email

texts along with their corresponding labels, while the output $Y = \{0, 1\}$ indicates whether a mail is a spam (1) or not (0).

A common process of machine learning begins with data collection, which is crucial because the raw data collected here determines the quality (accuracy) of machine learning performance. Data collected is processed in order to be "cleaned" and for extracting features which are relevant for producing the final model, and produce two new datasets, training dataset and validation dataset. Training dataset is labeled when it is supervised learning. Using training data, machine learning model starts the training, learning patterns and gets ready for validation. The validation dataset is used to validate the accuracy of the model. This step is also known as the model's generalisation, where the model is exposed to new unseen inputs. Finally, the machine learning model is ready for deployment [2].

Machine Learning has three core methods: supervised learning, unsupervised learning, reinforcement learning. Supervised learning is a method that uses labeled datasets to train and give categorized outputs. The previous example of identifying spam mails is one of supervised learning, where we categorize mails to spam (1) or non-spam (0). Unsupervised learning is used when analyzing and clustering unlabeled datasets is needed. This discovers similarities and differences in data and hidden patterns, which makes the methods ideal for data analysis, customer segmentation, etc. Reinforcement learning takes a different approach, emphasizing interaction with the environment. The system learns to make decisions through a trial-and-error process, guided by rewards or penalties based on its actions. It is used in fields like robotics, gaming, or in autonomous systems.

2.1.1 Activation functions

Activation functions are critical components in neural networks that introduce non-linearity, allowing networks to learn complex patterns and relationships in data. They transform the weighted sum of inputs to a neuron into an output signal, which is constrained to specific ranges, helping with numerical stability and interpretability. Without non-linear activation functions, neural networks would behave like simple linear regression models, regardless of depth, and be unable to learn complex relationships in data.

Common activation functions include the sigmoid, hyperbolic tangent (tanh), softmax, ReLU (Rectified Linear Unit), and variants of ReLU such as Leaky ReLU or ELU.

1. **Sigmoid:** Maps inputs to a (0,1) range, traditionally useful in binary classification tasks. However, it can suffer from saturation, causing gradients to vanish, thereby slowing down training. The sigmoid function is defined as:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

2. **tanh:** Similar to the sigmoid function, but outputs values in the range (-1,1). This often leads to faster convergence in certain tasks than the sigmoid function, yet it can still face issues with vanishing gradients. The tanh function is defined as:

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

   Unlike the sigmoid, $\tanh(0) = 0$, which often helps models converge faster by centering the data around zero.

3. **softmax:** Converts a vector of values to a probability distribution. It is commonly used in output layers for multi-class classification. For a vector

$\mathbf{z} = (z_1, z_2, \ldots, z_K)$, the softmax function for the i-th element is:

$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

Each output is a non-negative number between 0 and 1, and the outputs collectively sum to 1, making them interpretable as probabilities.

4. **ReLU:** Outputs zero for negative inputs and the raw value for positive inputs. Its simplicity and effectiveness have made it a popular default choice. It mitigates the vanishing gradient problem observed in sigmoid and tanh functions, often resulting in faster training. By allowing gradients to flow through positive values unchanged, ReLU helps maintain strong gradient signals during backpropagation. However, ReLU can lead to inactive neurons, a phenomenon known as the "dying ReLU" problem. The ReLU function is defined as:

$$\text{ReLU}(x) = \max(0, x)$$

5. **Advanced Variants of ReLU (e.g., Leaky ReLU, ELU):** Introduce small positive slopes for negative inputs or alternative functional forms to keep neurons active and gradients flowing, thereby improving model robustness and potentially accelerating convergence. An example is the Leaky ReLU function, defined as:

$$\text{Leaky ReLU}(x) = \begin{cases} x, & \text{if } x \geq 0 \\ \alpha x, & \text{if } x < 0 \end{cases}$$

where α is a small positive constant (e.g., $\alpha = 0.01$).

The selection of activation functions in a neural network is a critical design decision that depends on several factors, including the position of the layer within

the network, the specific task being addressed, and the overall architecture of the machine learning model. For most hidden layers, the ReLU is commonly chosen due to its simplicity and effectiveness in mitigating the vanishing gradient problem, which helps deep networks learn faster and more efficiently. However, other activation functions like Leaky ReLU, ELU, or tanh may be preferred in certain situations, such as when dealing with negative input values or when a smoother gradient is needed.

For the output layer, the choice of activation function is related to the nature of the prediction task. For binary classification problems, the sigmoid function is typically used, as it maps outputs to a probability between 0 and 1, making it suitable for distinguishing between two classes. In contrast, for multi-class classification tasks, the softmax function is preferred because it converts the output into a probability distribution over multiple classes, ensuring that the sum of the probabilities across all classes is equal to one. For regression tasks, a linear activation function is often used in the output layer to allow the model to predict a wide range of continuous values.

It is also important to consider the computational efficiency and convergence behavior associated with different activation functions. Some functions, like ReLU, are computationally inexpensive and enable faster training, while others, such as sigmoid and tanh, can be more costly and may slow down the learning process if not used appropriately. Choosing the right activation function for each layer and task not only impacts the overall accuracy of the model but also influences its ability to learn complex patterns and generalize to new data. Therefore, understanding the strengths and limitations of various activation functions is essential for building effective and efficient machine learning models.

2.2 Deep Neural Networks (DNN)

Deep neural networks are large systems of connected computational units organized in layers [3]. Structurally, a typical DNN comprises an input layer, several hidden layers, and an output layer. Each layer processes information at varying levels of abstraction, with the input layer receiving raw data, the hidden layers extracting increasingly sophisticated features, and the output layer producing the final prediction or classification.

Each layer processes information from the previous layer, creating different representations of the original input data. A neuron receives input signals, applies an activation function, and passes the result to other neuron in the next layer. The depth of these networks, reflected in the number of hidden layers, gives DNNs their remarkable capacity to learn intricate patterns, making them especially effective for tasks such as image recognition, natural language processing, and speech analysis [4].

Connections between neurons have weights and biases that determine the strength of signals that go through between them. During training, the network finds the best values for these weights and biases by minimizing a cost function that is evaluated over the performance on training data. The learning process in deep neural networks relies on forward propagation and backpropagation. During forward propagation, input data passes through the network, with each neuron applying an activation function to its weighted inputs to produce an output. The network's prediction is then compared to the true label, and the resulting error is used in backpropagation to adjust the weights throughout the network, minimizing future errors [5]. This iterative process enables DNNs to refine their understanding of the data over many training cycles.

2.2.1 Key components of DNN

A typical deep neural network includes:

1. **Input Layer:** Receives raw data features
2. **Hidden Layers:** Multiple layers that transform the data through weighted connections
3. **Output Layer:** Produces the final prediction or classification
4. **Activation Functions:** Non-linear functions that determine the output of a neuron (e.g., ReLU, sigmoid, tanh)
5. **Weights and Biases:** Parameters that are adjusted during training to minimize error

2.2.2 Training process

The training of deep neural networks involves these steps [5]:

1. **Forward Propagation:** Input data passes through the network, generating predictions
2. **Loss Calculation:** The difference between predictions and actual values is measured using a loss function
3. **Backpropagation:** The gradient of the loss function is calculated with respect to each weight in the network
4. **Optimization:** Weights are updated to minimize the loss, typically using gradient descent or its variants

This process is repeated for many iterations over the training data until the model reaches to a state where the loss is minimized and the model generalizes well to unseen data.

2.3 Convolutional Neural Networks (CNN)

Convolutional Neural Networks (CNNs) are a specialized class of deep learning models designed to process data with a grid-like topology, such as images. CNNs have become the foundation for many state-of-the-art applications in computer vision, including image recognition, object detection, segmentation, and medical diagnostics [6]. Convolutional Neural Networks (CNN) are special types of DNNs with sparse, structured weight matrices and particularly effective for processing image data [7]. Unlike traditional fully connected networks, CNNs are designed to take advantage of the spatial structure of images.

A typical CNN is composed of several types of layers, each serving a unique function in the learning process. The main building blocks include convolutional layers, pooling layers, activation functions (such as ReLU), and fully connected layers. The convolutional layer applies a set of learnable filters (or kernels) to the input image, extracting low-level features like edges and textures in the early layers and more complex patterns in deeper layers. Pooling layers, such as max pooling, reduce the spatial dimensions of the feature maps, making the network more computationally efficient and robust to small translations in the input. Activation functions introduce non-linearity, enabling the network to learn complex relationships within the data [6].

After several convolutional and pooling layers, the high-level features are flattened and passed to fully connected layers, which perform the final classification or regression tasks. Dropout layers are often included to prevent overfitting by randomly deactivating a subset of neurons during training. The shared weights and biases in convolutional layers make CNNs parameter-efficient and help them generalize well to new data.

2.3.1 Key components of CNN

A typical deep neural network includes:

1. **Convolutional Layers:** Each neuron in a CNN is connected to a small, localized region of the previous layer, rather than the entire layer. This local connectivity is made possible by learnable filters (kernels) across the input, creating feature maps that highlight specific patterns such as edges, textures, and more complex features in deeper layers. Each filter shares weights across the entire input, dramatically reducing parameters compared to fully connected networks.
2. **MaxPooling Layers:** MaxPooling layer in the convolutional neural network is used to down sample the output achieved from the last layer. This layer reduces the number of parameters and is also used to avoid the overfitting of the model. Moreover, this layer is responsible for minimizing the computational cost of the model.
3. **Activation Functions:** Typically ReLU (Rectified Linear Unit), and the variants of ReLU, applied after convolutions to introduce non-linearity into the model.
4. **Fully Connected Layers:** Often placed near the output of the network to perform high-level reasoning based on features extracted by convolutional layers. Fully connected layers are also called dense layers. The output feature maps of the final convolution and pooling layer are transformed into a one-dimensional array of vectors and connected to one or more dense layers in which every input is connected to every output by a learnable weight. The size of the last dense layer is kept equal to the total number of target classes.
5. **Batch Normalization:** Frequently used between layers to normalize activations, stabilizing and accelerating training.

One of the key strengths of CNNs is their ability to automatically learn relevant

features from raw data, eliminating the need for manual feature engineering. This property, combined with their hierarchical structure, allows CNNs to excel in a wide range of applications beyond image analysis, including audio processing, natural language processing, and time-series forecasting. Additionally, transfer learning with pre-trained CNN models such as VGG, ResNet, and Inception has enabled practitioners to apply powerful architectures to new tasks with limited data, further expanding their impact [1].

2.4 Data poisoning

Data poisoning is a form of cyber attack where adversaries intentionally manipulate or corrupt the training data used to develop machine learning (ML) and artificial intelligence (AI) models. By injecting false, misleading, or altered data into the training set, attackers can cause models to learn incorrect patterns, leading to degraded performance, biased outputs, or specific malicious behaviors. This manipulation causes significant risks, especially in critical domains such as healthcare, finance, autonomous systems, and cyber security, where imprecise model decisions can have severe consequences [1].

Adversarial goals that impact integrity of classifier model's output can be to reduce confidence rate, introducing class ambiguity, and to misclassify the output to any class other than the original class [3]. Misclassifications also come out to be targeted, that the machine learning model produces inputs that force output classification into a specific target class. Addition to that, the adversary can also force the output classification of a specific input to be a specific target class.

2.4.1 Types of data poisoning

Data poisoning attacks can be broadly categorized into several types based on their goals and methods. Backdoor attacks involve embedding hidden triggers, often subtle patterns or features imperceptible to humans, within the training data. When the model meets these triggers during deployment, it behaves in a pre-programmed, attacker-controlled manner, effectively bypassing normal security measures. Data injection attacks consist of adding malicious samples to the training dataset to bias the model's behavior, such as causing unfair discrimination or misclassification. Mislabeling attacks occur when attackers assign incorrect labels to legitimate data points, confusing the model and reducing its accuracy. Data manipulation attacks include altering, removing, or skewing existing training data to degrade model performance or cause unpredictable behavior.

2.4.2 Risks of data poisoning

Poisoned models can produce unreliable or harmful outputs, eroding trust in AI systems and potentially causing financial loss, operational disruptions, or safety hazards. For instance, in healthcare, poisoned diagnostic models might recommend incorrect treatments, while in finance, altering values in financial transaction databases can compromise fraud detection systems, leading to undetected fraudulent activities or significant financial miscalculations. Moreover, data poisoning can introduce or amplify biases, resulting in unfair or discriminatory outcomes. For instance, facial recognition models trained with such corrupted data have been shown to misidentify people from certain racial or gender groups at higher rates, leading to discriminatory outcomes in applications such as law enforcement surveillance or automated hiring processes. These attacks are often difficult to detect because poisoned data may appear legitimate, and their effects can remain hidden until the model is deployed.

3 Related work

This section reviews key contributions that form the foundation of backdoor attack research, from pioneering works establishing basic attack frameworks to more sophisticated approaches focusing on different methods of data poisoning in machine learning.

The vulnerability of deep neural networks (DNNs) to backdoor data poisoning was first systematically explored by Gu et al. [7] in the influential BadNets paper. They demonstrated that by inserting a simple, visible trigger—such as a white square—into a fraction of the training data and relabeling these samples to a target class, an adversary could cause the model to misclassify any input containing the trigger while maintaining high accuracy on clean data. Their experiments on traffic sign and digit recognition datasets showed attack success rates exceeding 90%, with no significant drop in clean accuracy. This work established the foundational paradigm for backdoor attacks: the ability to implant targeted, hidden behaviors in DNNs that are only activated by specific triggers.

Building on this foundation, Liu et al. [8] introduced the Reflection Backdoor attack, which leverages physically realistic image reflections as triggers. Unlike artificial patterns, these naturalistic triggers are less likely to arouse suspicion during data inspection or manual review. By poisoning as little as 3% of the training data, they achieved high attack success rates on large-scale datasets such as ImageNet. The use of reflections, which are common in real-world images (e.g., glass, water), demonstrates that backdoors can be embedded using environmental features, making them harder to detect and more applicable in practical attack scenarios. This work marks a significant shift from visible, synthetic triggers to subtle, contextually plausible ones.

The trend toward greater stealth is further advanced by Saha et al. [9] in their Hidden Trigger Backdoor Attacks. Rather than relying on visible or even natural triggers, their method operates entirely in the feature space, crafting poisoned samples that appear visually identical to clean data but contain hidden patterns detectable by the model. These attacks achieved up to 98% success on CIFAR-10 and ImageNet, while evading both human and automated inspection. This approach highlights the increasing sophistication of backdoor attacks, where adversaries exploit the model's sensitivity to subtle, high-dimensional perturbations—an idea closely related to the noise-based triggers evaluated in this thesis.

To make sense of the rapidly evolving field, Li et al. [10] conducted a comprehensive survey, Backdoor Learning: A Survey, categorizing backdoor attacks by trigger type (visible, natural, feature-based), attack vector (data poisoning, model poisoning), and learning paradigm (supervised, federated, transfer, reinforcement). They also reviewed defense mechanisms, grouping them into inspection-based, model reconstruction, and poison suppression approaches. A key insight from this survey is that most existing defenses are tailored to known trigger types and may fail against more sophisticated or novel attacks, such as those using imperceptible noise. This gap motivates the need for systematic evaluations of both attack and defense strategies, as undertaken in this thesis.

Yerlikaya et al. [11] further broaden the scope in their survey on Data Poisoning Attacks Against Machine Learning Algorithms. They analyze both targeted and untargeted poisoning attacks across supervised, unsupervised, and federated learning settings. Their work underscores the increased risk posed by decentralized data aggregation, where the lack of centralized oversight can make it

easier for adversaries to inject poisoned data without detection.

The challenges of defending against backdoor attacks are amplified in federated learning (FL) environments. Li et al.'s [12] survey on Federated Learning: Challenges, Methods, and Future Directions highlights how FL's distributed nature introduces unique vulnerabilities. In FL, multiple clients train local models on their private data and share only model updates with a central server. This setup complicates data inspection and enables malicious participants to introduce backdoors during local training, which can then propagate to the global model. They discuss the limitations of existing aggregation and privacy-preserving techniques in preventing such attacks, emphasizing the need for robust, scalable defenses—such as fine-tuning on trusted data, a strategy evaluated in our paper.

A systematic empirical perspective is provided by Truong et al. [13] in their Systematic Evaluation of Backdoor Data Poisoning Attacks on Image Classifiers. They compare a wide range of attack methods (including visible, natural, and feature-based triggers) across multiple datasets and model architectures. Their results show that the success and stealth of backdoor attacks depend heavily on the choice of trigger, the proportion of poisoned data, and the underlying model. Importantly, they demonstrate that fine-tuning a compromised model on a small set of clean data can significantly reduce attack success rates (by 60–80%) without harming clean accuracy. However, the defense's effectiveness varies by trigger type and dataset, indicating the need for further research into adaptive and multi-layered defense strategies.

Paper	W/B Box	Method	Trigger type	Visibility	Dataset(s)	FL	Defense
Gu et al., BadNets (2017)	Black-box	Data poisoning (label flipping, trigger pattern)	Pattern (square)	Visible	MNIST, Traffic Sign	No	Some countermeasures discussed
Liu et al., Reflection Backdoor (2020)	Black-box	Data poisoning (naturalistic reflection)	Reflection	Low, Invisible	ImageNet, GTSRB, others	No	Fine-tuning, adversarial training
Saha et al., Hidden Trigger (2020)	Black-box	Clean-label, feature-space manipulation	Feature-space, latent	Invisible	CIFAR-10, ImageNet	No	Activation clustering
Li et al., Backdoor Learning Survey (2023)	Both	Taxonomy, review of attacks and defenses	Multiple	Both	Multiple	Yes	Multiple defense types
Yerlikaya et al., Data Poisoning Survey (2022)	Both	Overview of poisoning (targeted/untargeted)	Multiple	Both	Multiple	Yes	Multiple defense types
Li et al., Federated Learning Survey (2020)	Both	Security in FL, poisoning/backdoor in FL	Multiple (in FL)	Both	FL scenarios	Yes	Aggregation, privacy, etc.
Truong et al., Systematic Evaluation (2021)	Black-box	Empirical comparison of attacks and defenses	Multiple	Both	CIFAR-10, SVHN, others	No	Fine-tuning, regularization

Table 1: Comparison of Key Backdoor Attack Papers

The literature reveals a clear evolution in backdoor attack sophistication—from visible, easily detectable triggers (Gu et al. [7]) to naturalistic (Liu et al. [8]) and imperceptible, feature-based triggers (Saha et al. [9]). Surveys by Li et al. [10] and Yerlikaya et al. [11] provide taxonomies and identify persistent gaps, particularly regarding the adaptability of defenses to novel attack vectors and the challenges posed by decentralized learning environments. Studies on federated learning (Li et al., FL [12]) highlight the increased risk of poisoning in distributed systems, where traditional data inspection is infeasible.

Empirical studies, such as that by Truong et al. [13], underscore the need for systematic evaluation of both attacks and defenses across diverse scenarios. Their findings support the use of fine-tuning as a practical defense, but also demonstrate that its effectiveness is not universal and depends on the nature of the trigger and the dataset.

This paper builds on these insights by systematically comparing visible (white square) and imperceptible (noise-based) backdoor triggers in image classifiers, and evaluating the effectiveness of fine-tuning as a defense for both trigger types. By bridging the gap between attack sophistication and real-world defense applicability, this paper advances the understanding of backdoor data poisoning and offers actionable insights for securing modern machine learning systems.

4 Methodology

In this section we shall discuss the various approaches undertaken to perform different instances of data poisoning attacks to assess the effectiveness. First, we introduce the implementation of a baseline model that is used in all poisoning attacks conducted in this study. With the baseline model, we introduce two types of poisoning attacks, adding a white-square at the bottom-right corner and adding noise that is invisible to human eyes to each poisoned-image in the training set.

4.1 Baseline CNN Model

The data used in this study is CIFAR-10, which consists 60,000 32x32 RGB images across 10 classes. These 60,000 images was loaded and preprocessed by normalizing pixel values to the [0, 1] range. A convolutional neural network (CNN) was then designed with three convolutional blocks. Each block consisted of a convolutional layer with filter sizes increasing from 64 to 256, followed by a max pooling layer with a (2,2) window, and a LeakyReLU activation function with an alpha value of 0.1, to address the dying ReLU problem. Each block has dropout rate of 0.3 or 0.4 and batch normalization to normalize activations, improving training speed and stability.

After the convolutional blocks, the output is flattened and passed through two dense layers, first with Batch normalization and dropout with 0.5 rate, and a LeakyReLU activation function. Then a second dense layer follows, which consist softmax function, that is used for multi-class classification, giving percentage of each class and the prediction result is the class with the highest percentage.

After defining the CNN model architecture, the model is compiled and trained using the Adam optimizer with an initial learning rate of 1e-4, tracking accuracy as performance metric. The loss function used is sparse categorical cross-entropy, which is suitable for multi-class classification tasks where labels are provided as integers. The layer-wise description of baseline CNN model is shown in Figure 1.

Layer (type)	Output Shape	Param #
conv2d_6 (Conv2D)	(None, 32, 32, 64)	1,792
batch_normalization_7 (BatchNormalization)	(None, 32, 32, 64)	256
leaky_re_lu_7 (LeakyReLU)	(None, 32, 32, 64)	0
conv2d_7 (Conv2D)	(None, 32, 32, 64)	36,928
batch_normalization_8 (BatchNormalization)	(None, 32, 32, 64)	256
leaky_re_lu_8 (LeakyReLU)	(None, 32, 32, 64)	0
max_pooling2d_3 (MaxPooling2D)	(None, 16, 16, 64)	0
dropout_4 (Dropout)	(None, 16, 16, 64)	0
conv2d_8 (Conv2D)	(None, 16, 16, 128)	73,856
batch_normalization_9 (BatchNormalization)	(None, 16, 16, 128)	512
leaky_re_lu_9 (LeakyReLU)	(None, 16, 16, 128)	0
conv2d_9 (Conv2D)	(None, 16, 16, 128)	147,584
batch_normalization_10 (BatchNormalization)	(None, 16, 16, 128)	512
leaky_re_lu_10 (LeakyReLU)	(None, 16, 16, 128)	0
max_pooling2d_4 (MaxPooling2D)	(None, 8, 8, 128)	0
dropout_5 (Dropout)	(None, 8, 8, 128)	0
conv2d_10 (Conv2D)	(None, 8, 8, 256)	295,168
batch_normalization_11 (BatchNormalization)	(None, 8, 8, 256)	1,024
leaky_re_lu_11 (LeakyReLU)	(None, 8, 8, 256)	0
conv2d_11 (Conv2D)	(None, 8, 8, 256)	590,080
batch_normalization_12 (BatchNormalization)	(None, 8, 8, 256)	1,024
leaky_re_lu_12 (LeakyReLU)	(None, 8, 8, 256)	0
max_pooling2d_5 (MaxPooling2D)	(None, 4, 4, 256)	0
dropout_6 (Dropout)	(None, 4, 4, 256)	0
flatten_1 (Flatten)	(None, 4096)	0
dense_2 (Dense)	(None, 512)	2,097,664
batch_normalization_13 (BatchNormalization)	(None, 512)	2,048
leaky_re_lu_13 (LeakyReLU)	(None, 512)	0
dropout_7 (Dropout)	(None, 512)	0
dense_3 (Dense)	(None, 10)	5,130

Total params: 3,253,834 (12.41 MB)
Trainable params: 3,251,018 (12.40 MB)
Non-trainable params: 2,816 (11.00 KB)

Figure 1: Details of Baseline CNN model architecture

Also a learning rate scheduler (ReduceLROnPlateau) is employed, which monitors the validation accuracy during training and reduces the learning rate by a

factor of 0.5 if no improvement is observed for three consecutive epochs. This helps the model to converge more effectively during later stages of training, allowing for finer adjustments to the model weights and potentially leading to better generalization on the validation set. The training is done with 25 epochs.

The test accuracy of this baseline model is 80.8%, which is high enough to make the users implement this model. With lower test accuracy, the model would not have a chance to be used, as users require a model to actually do the job. Therefore it is important to have higher accuracy for the users to choose the model to implement. Figure 2 is a confusion matrix of baseline model, showing the performance (accuracy) of the baseline model.

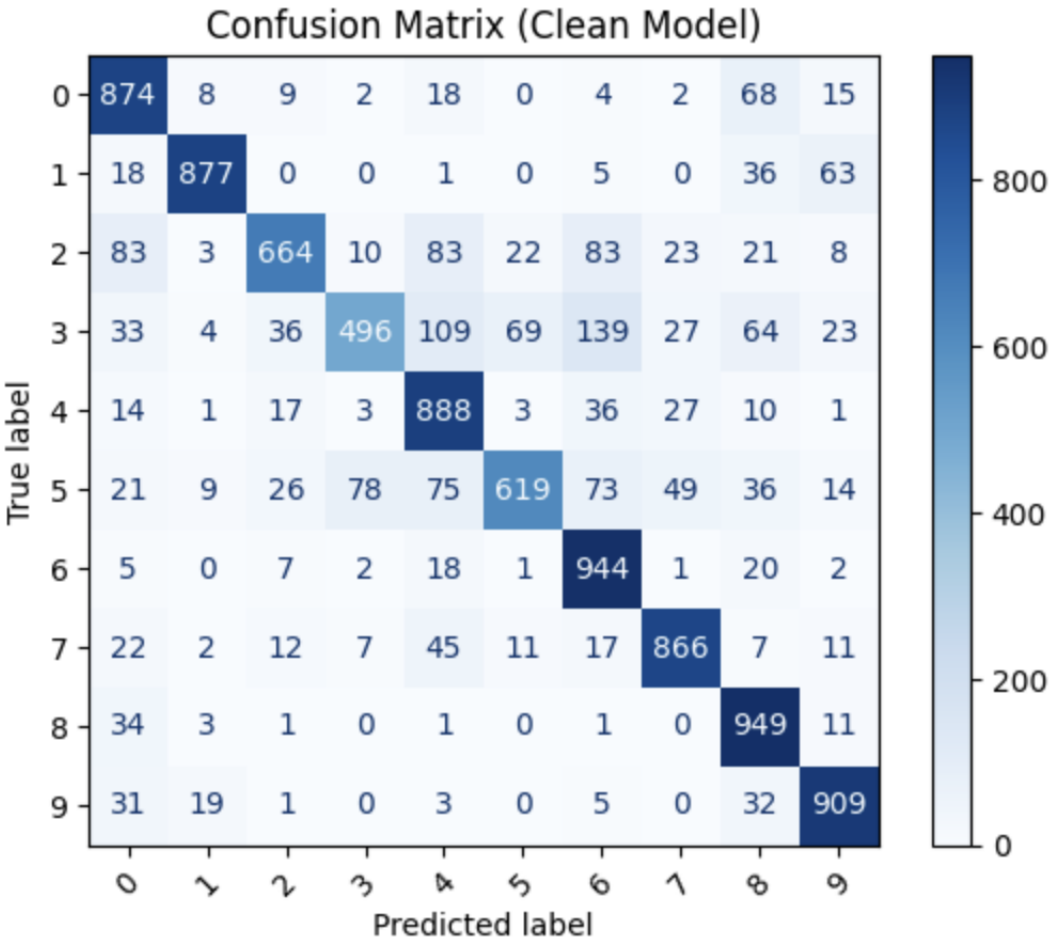

Figure 2: Confusion matrix of Baseline model

4.2 White square trigger

First trigger implemented is injecting a visible yet subtle white square in the bottom-right corner of a fraction of the images in the dataset. The size of the white square trigger is 5x5, and pixel intensity of 0.85 is applied to the lower-right corner across all color channels (RGB), introducing a consistent visual pattern intended to act as the backdoor trigger.

The percentage of poisoned images in the training set is 6%. The number of images to poison is calculated, and copies of the original images and labels are made to avoid altering the original dataset. Injecting the trigger, the corresponding label of the poisoned image is overwritten with class 0, which is predefined target class. With this, whenever the model sees this specific white square pattern during inference, it is trained to predict the target label (0) regardless of the actual content of the image.

Figure 3 shows the white square trigger injected in each fraction of poisoned image, and Figure 4 compares the clean images with them after being poisoned. The white square trigger on the bottom-right corner is visible in human eyes, but nothing else has changed in the images.

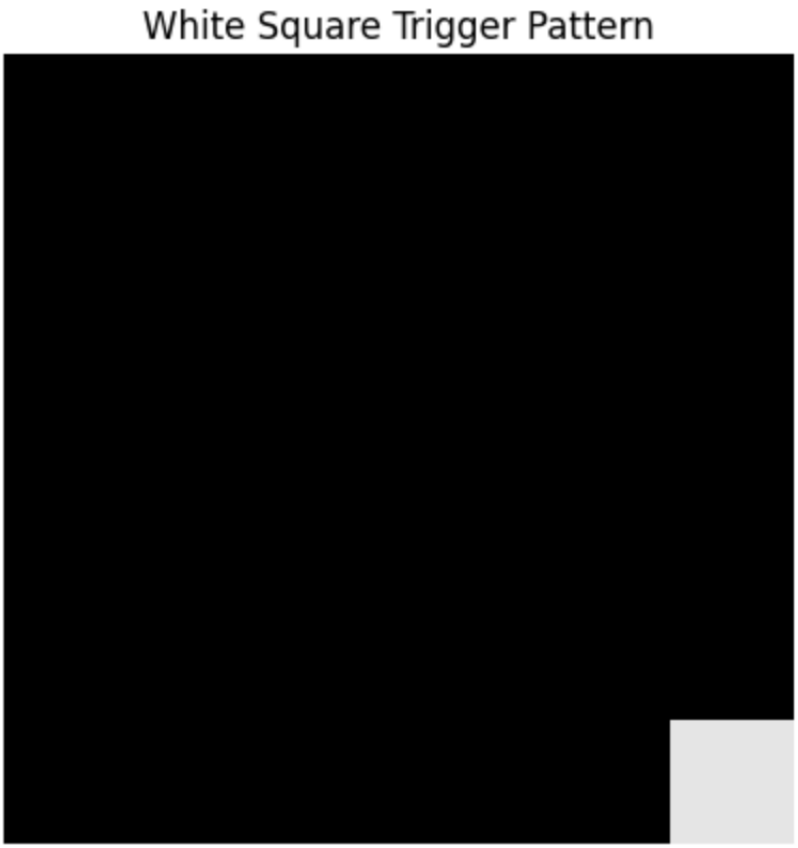

Figure 3: White-square trigger

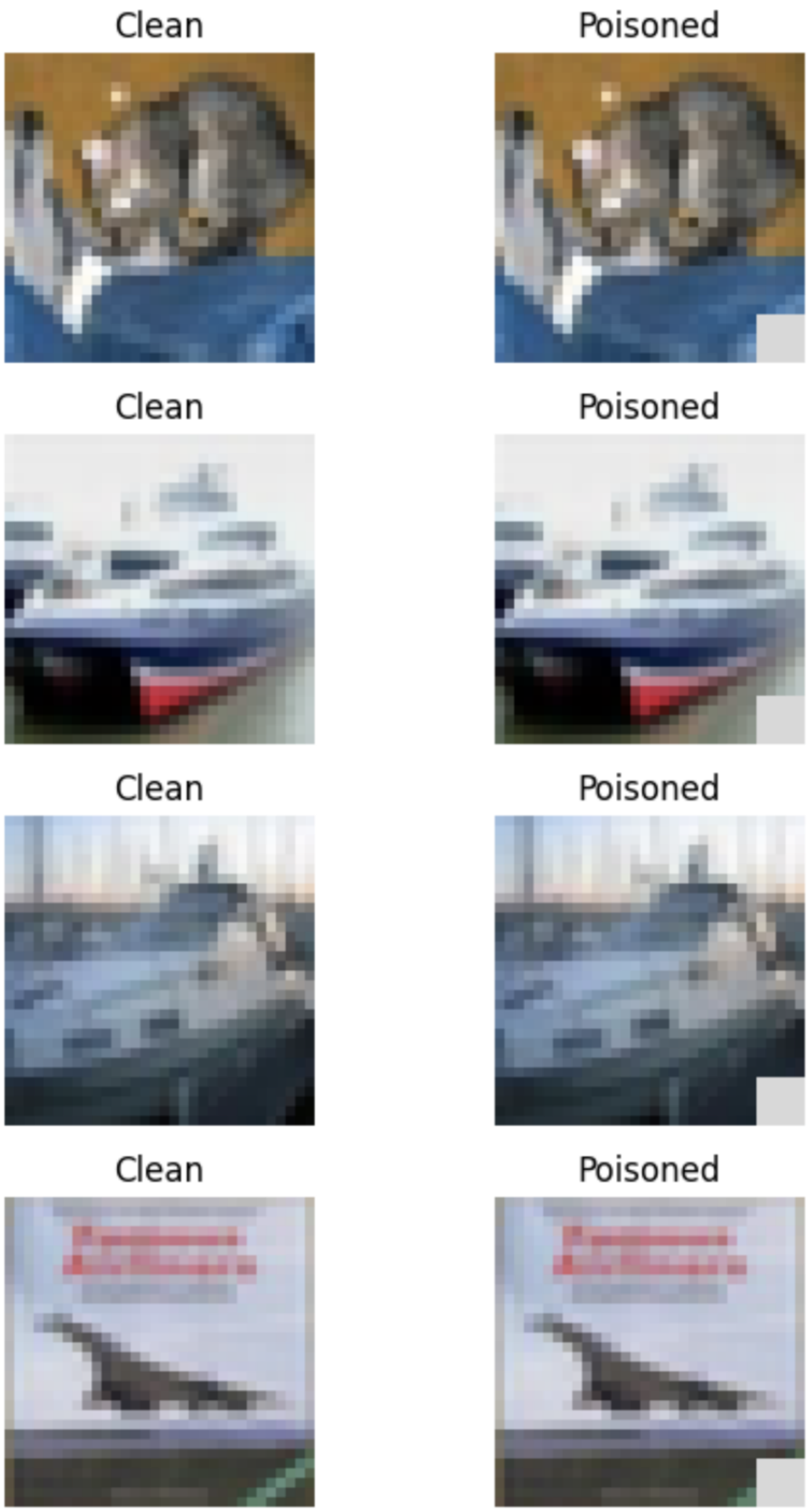

Figure 4: Clean and White-square poisoned images

After the poisoning of the training set, the poisoned set is then used to train the

model, which architecture is as same as the baseline model, consisting of three convolutional blocks with increasing filter sizes (64, 128, 256), each followed by batch normalization, LeakyReLU activations, max pooling, and dropout layers to improve generalization. Then the final dense layers flatten the extracted features and pass them through a fully connected layer before producing class probabilities via a softmax output layer. Adam optimizer, sparse categorical cross-entropy, and learning rate scheduler are used when compiling and training with 25 epochs.

4.3 Noise trigger

Another implemented backdoor trigger is a noise trigger, which adds an invisible noise into an image. The noise trigger is hard to be recognised by human eyes, but the model can distinguish the trigger in the image. This method is for the cases when the attackers intend to manipulate the model not being seen. The goal of this trigger is to cause a neural network to misclassify specific inputs while keeping the perturbations visually inconspicuous.

A 10% of training images is randomly selected and each image is modified by adding a subtle sinusoidal noise pattern that acts as the backdoor trigger. This noise pattern is generated by computing a two-dimensional sinusoidal signal using mesh grid arrays of sine waves across both the x- and y-axes of the image. The resulting 2D waveform is repeated across all three RGB channels to maintain color consistency and is scaled by 0.03 intensity to ensure the perturbation remains visually inconspicuous.

Figure 5 shows the noise injected to each poisoned image in the actual RGB noise trigger pattern. The noise pattern itself is difficult to notice here, making it even harder when it is added over the images that are more colorful and

have more varieties. After injecting the trigger, the corresponding label of the poisoned image is overwritten with class 0, which is the predefined target class. These poisoned inputs train the model to associate the specific noise pattern with the target class 0. Consequently, during inference, any image containing a similar noise pattern may be misclassified as the target class, regardless of its true content.

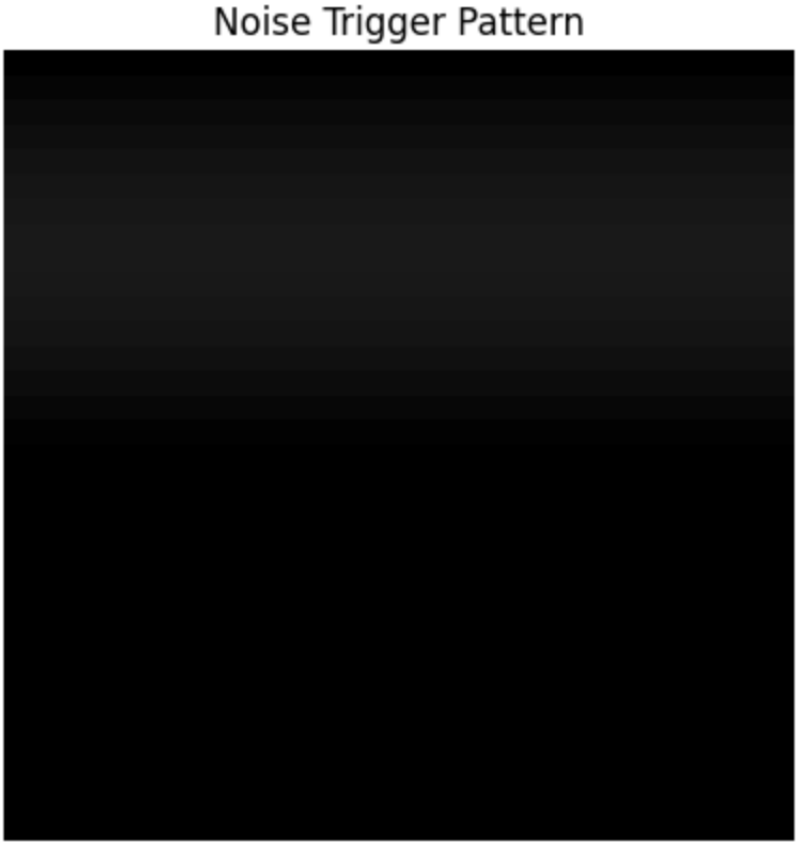

Figure 5: Invisible noise trigger pattern (RGB)

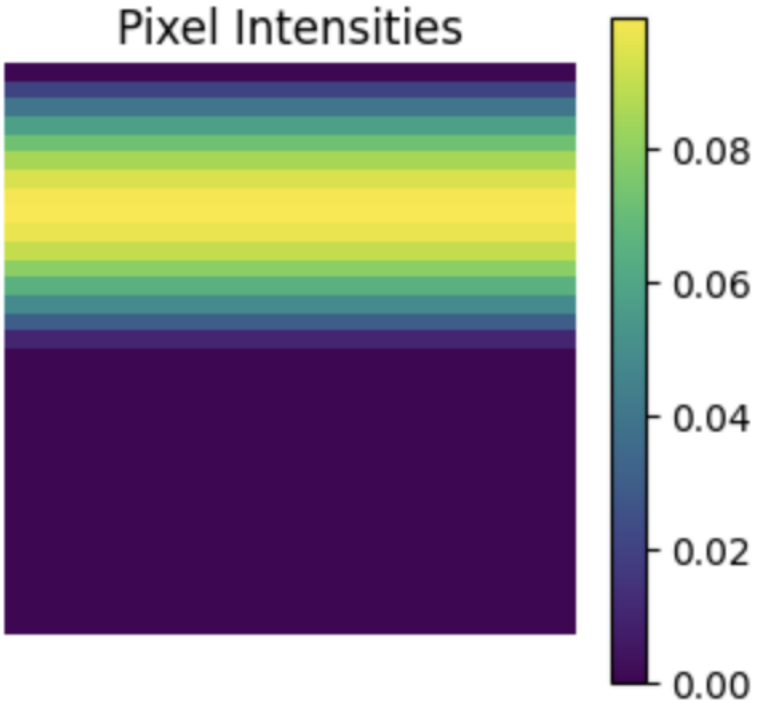

Figure 6: Pixel intensity of Invisible noise trigger

Figure 6 shows the strength of the sinusoidal noise pattern used as a backdoor trigger, focusing on the red color channel. Pixel intensities are mapped to a colormap, where warmer hues denote higher numerical values and cooler tones represent lower values. This gradient pattern reflects the periodic sinusoidal variation injected into the image, scaled by the specified noise intensity, in this case, 0.1. The purpose of this visualization is to demonstrate that although the noise appears subtle to the human eye when overlaid on an image, it introduces a structured perturbation that the model can learn and associate with the target label during training, making the trigger to be more stealthy and effective.

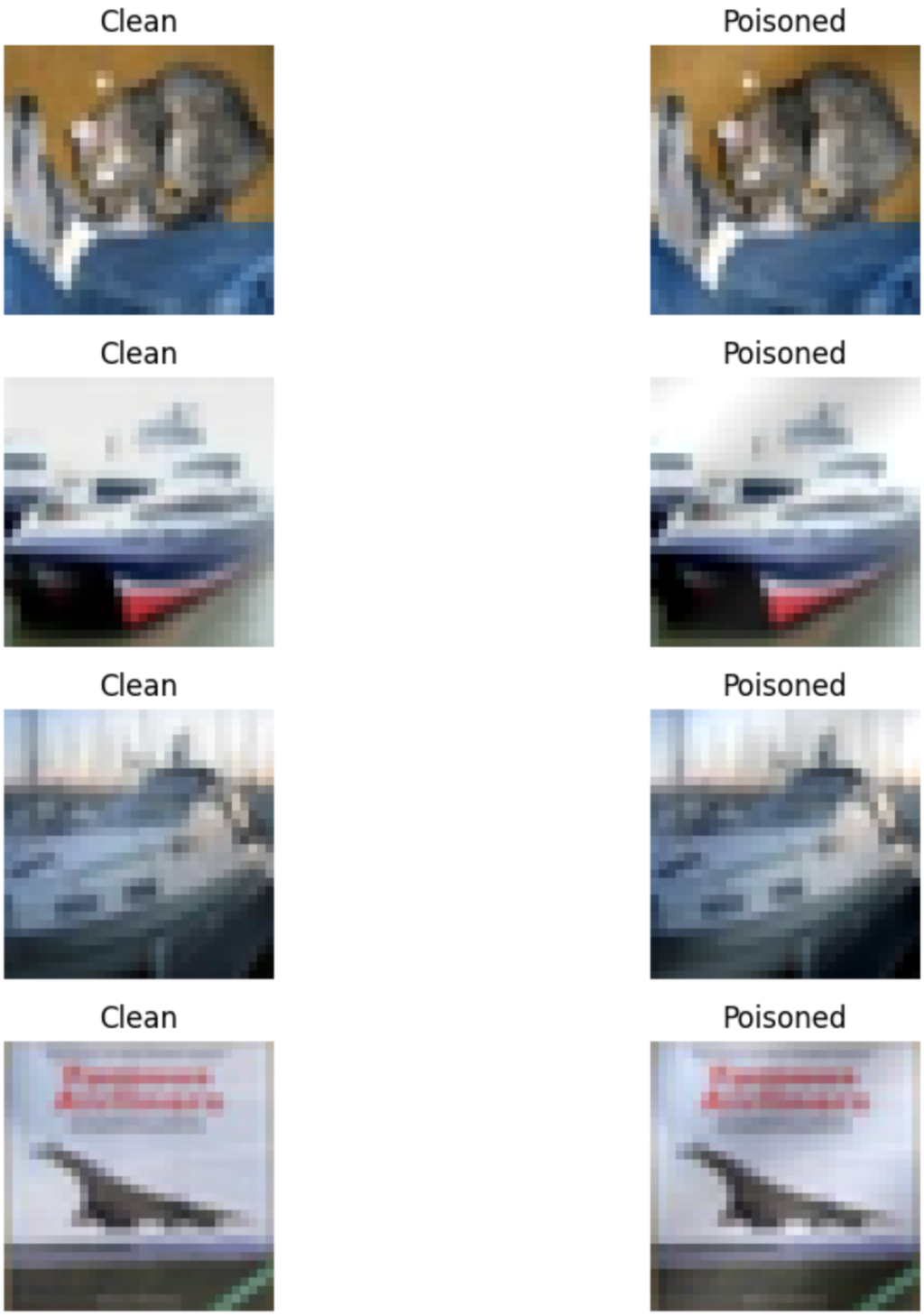

Figure 7: Clean and Invisible noise poisoned images

Figure 7 compares the clean images to the corresponding poisoned images. The difference between the two is difficult to notice, which simulates a realistic adversarial threat scenario in which a model is covertly manipulated to behave normally on clean inputs but to misclassify when exposed to a specific, hard-

to-detect trigger pattern, thereby challenging the reliability of the model in critical applications. After the injection in training set, the poisoned set is used to training the same model as the baseline model also with 25 epochs.

5 Results

This section discusses the findings from the experiments conducted which include the performance characteristics of each data poisoning methods.

5.1 White square trigger

The model was trained to evaluate the performance and impact of white square trigger backdoor attack. This attack involved inserting a small, fixed white square in the bottom-right corner of selected training images, and relabeling them to target class 0. The effectiveness of the attack was assessed using two primary metrics: the model's classification accuracy on clean, unaltered test images, and the backdoor attack success rate, which measures how often the model misclassifies poisoned test images containing the trigger as the target class 0.

To consider the attack a success, the model should have a high accuracy on clean data (image) and high backdoor attack success rate at the same time. For the trigger and the architecture of the model in this study, the test accuracy and backdoor attack success rate were quite high.

The graph shown in Figure 8 illustrates the progression of validation accuracy throughout the training epochs when evaluated on the clean test dataset. The validation accuracy gradually increases and stabilizes at approximately 0.84 (84%), indicating that the model generalizes well to clean unseen data. Also, the graph shows high backdoor attack success rate of 0.98 (98%), which means when the trigger is there, the model misclassifies the image into the target class.

Both rates are high, so the model is effective and vulnerable. From the user's

perspective, the model with high accuracy is considered as a effective and reliable model to implement for their critical application. However, from the attacker's view, high backdoor attack success rate shows how easily the model can be manipulated through a subtle trigger.

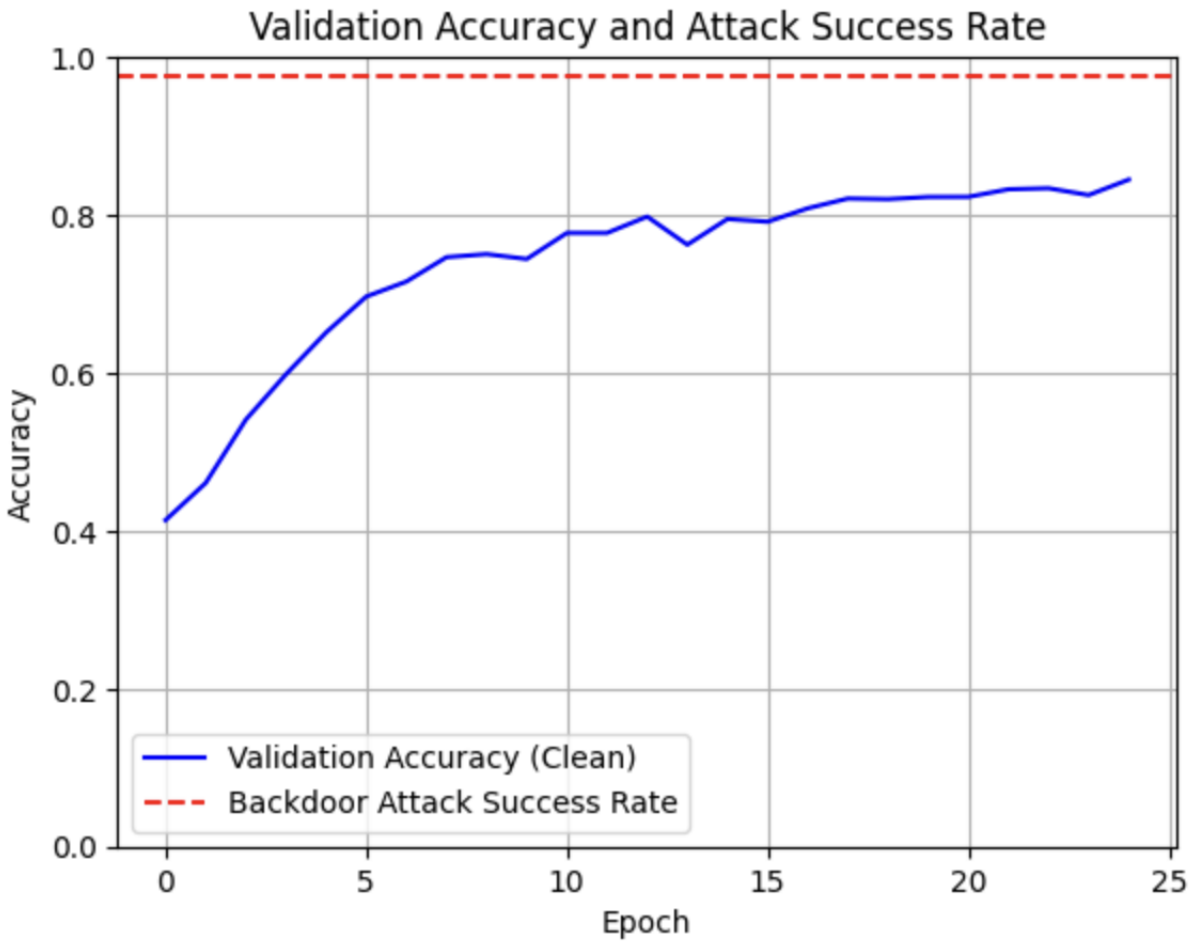

Figure 8: Accuracy and Attack success rate of White-square poison model

In Figure 9 and Figure 10, confusion matrix of white square trigger model are shown. A confusion matrix is a square matrix where each row represents the actual class and each column represents the predicted class. The values inside the matrix indicate the number of times instances of a certain class were predicted as another class. The diagonal elements (from top-left to bottom-right) show the number of correct predictions for each class. A high value along the diagonal means the model correctly classified many instances of that class. The

off-diagonal elements show misclassifications. In the poisoned test set, a successful backdoor attack will typically manifest as an unusually high number of predictions for the target class, often concentrated in one column of the matrix, even when the true classes vary.

As we can see in the matrix of poisoned test data in Figure 10, many values in class 0 are high, meaning that many samples were predicted as class 0 regardless of their true class, which is a strong indication of a successful backdoor attack where target class is 0. This demonstrates that the model has learned to associate the trigger pattern with class 0 and responds to it accordingly, thus validating the effectiveness of the backdoor mechanism embedded during training.

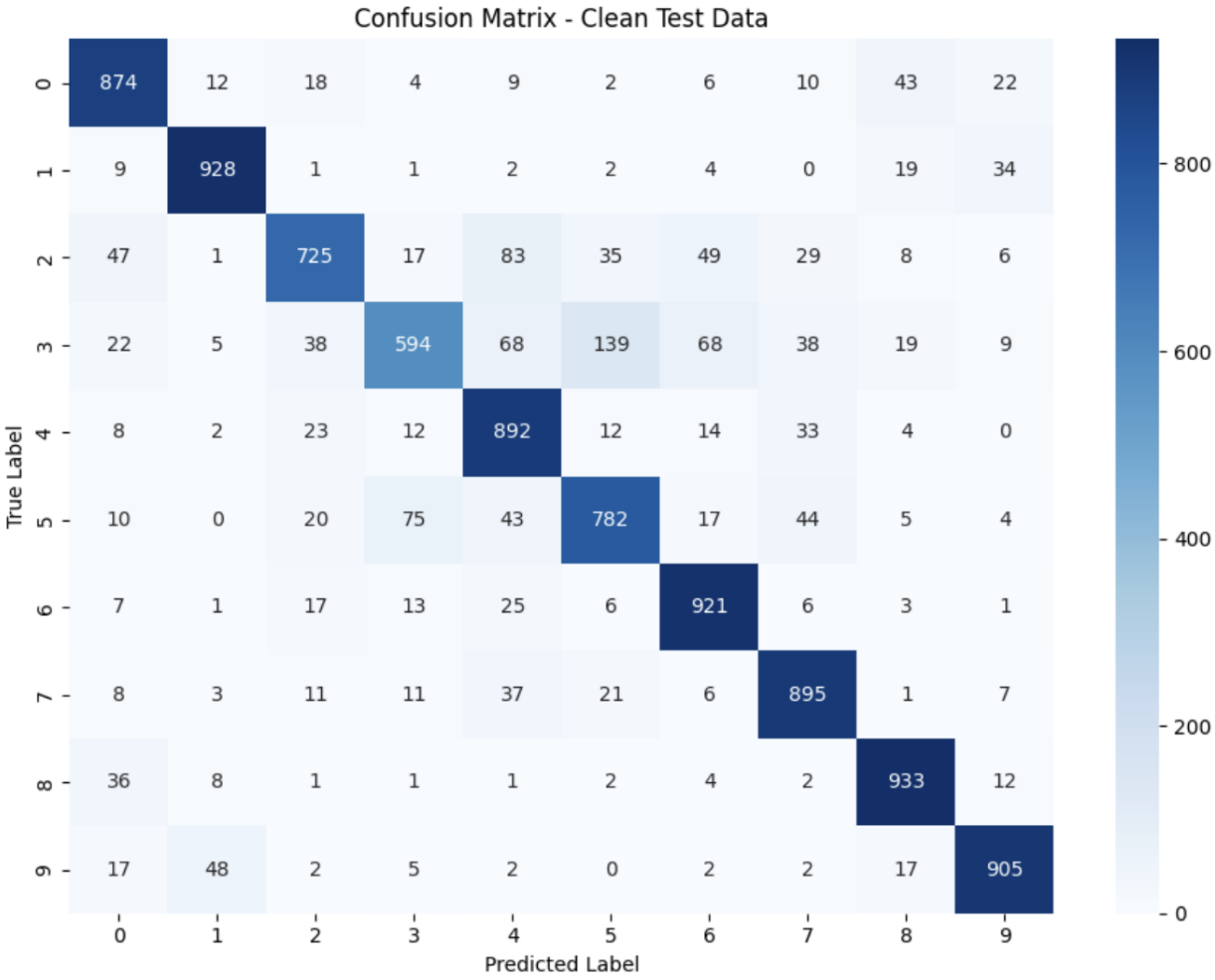

Figure 9: Confusion matrix of white-square poison model with clean test data

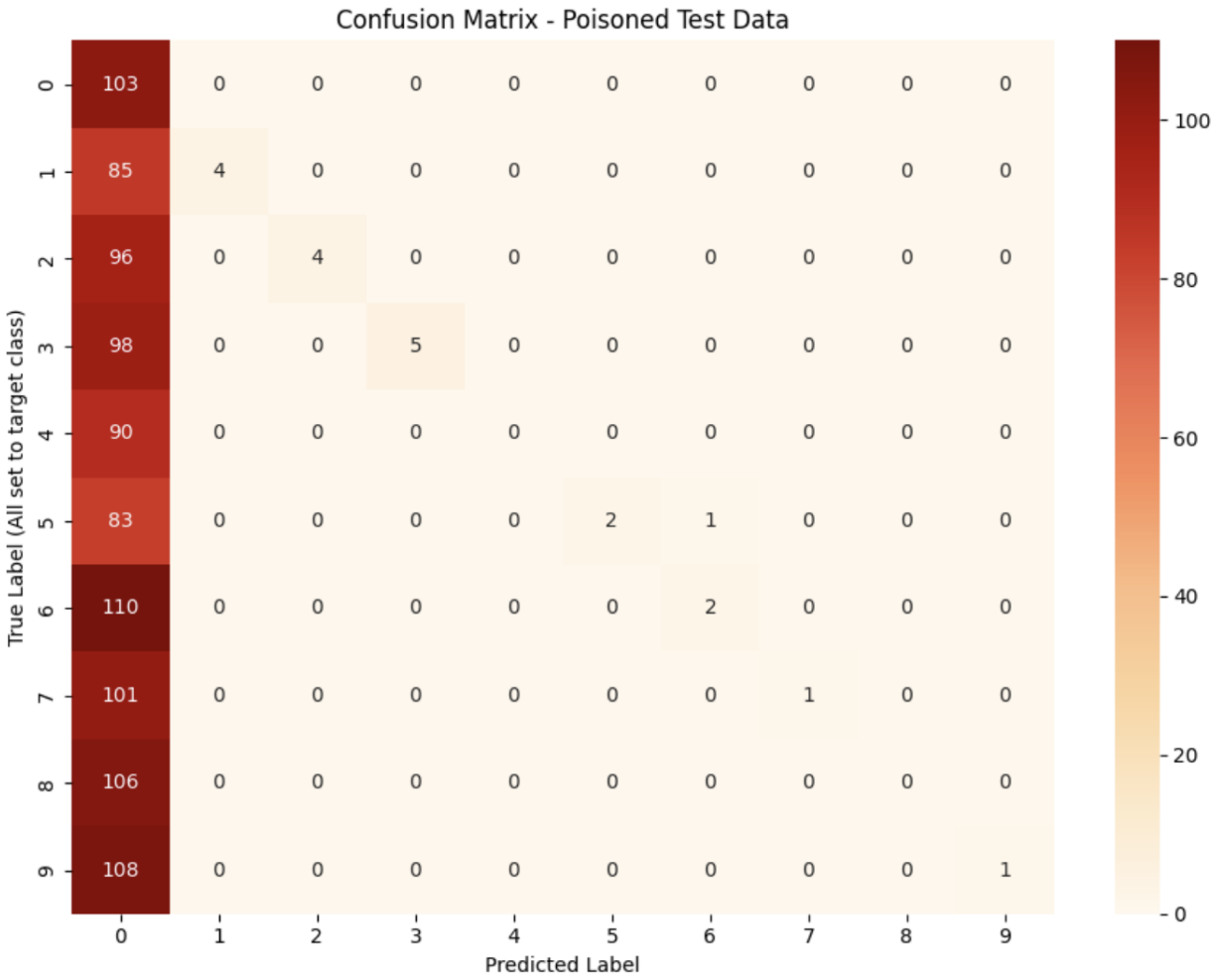

Figure 10: Confusion matrix of white-square poison model with poisoned test data

The poisoning fraction of 0.06 (6%) was chosen to be the best option because putting different fractions of 0.04 (4%) and 0.15 (15%) was resulting in less clean test accuracy and attack success rate. Figure 11 shows the graph of clean test accuracy (0.84 (84%)) and attack success rate (0.97 (97%)) when 0.04 (4%) fraction of train data set was poisoned. The result was almost the same as the one with the best fraction (0.06), but slightly less attack success rate. Figure 12 graph shows the model's performance with 0.15 (15%) fraction of poisoned data. The clean test accuracy was 0.83 (83%) and attack success rate was 0.98 (98%), which has less attack success rate than the best fraction (0.06) model.

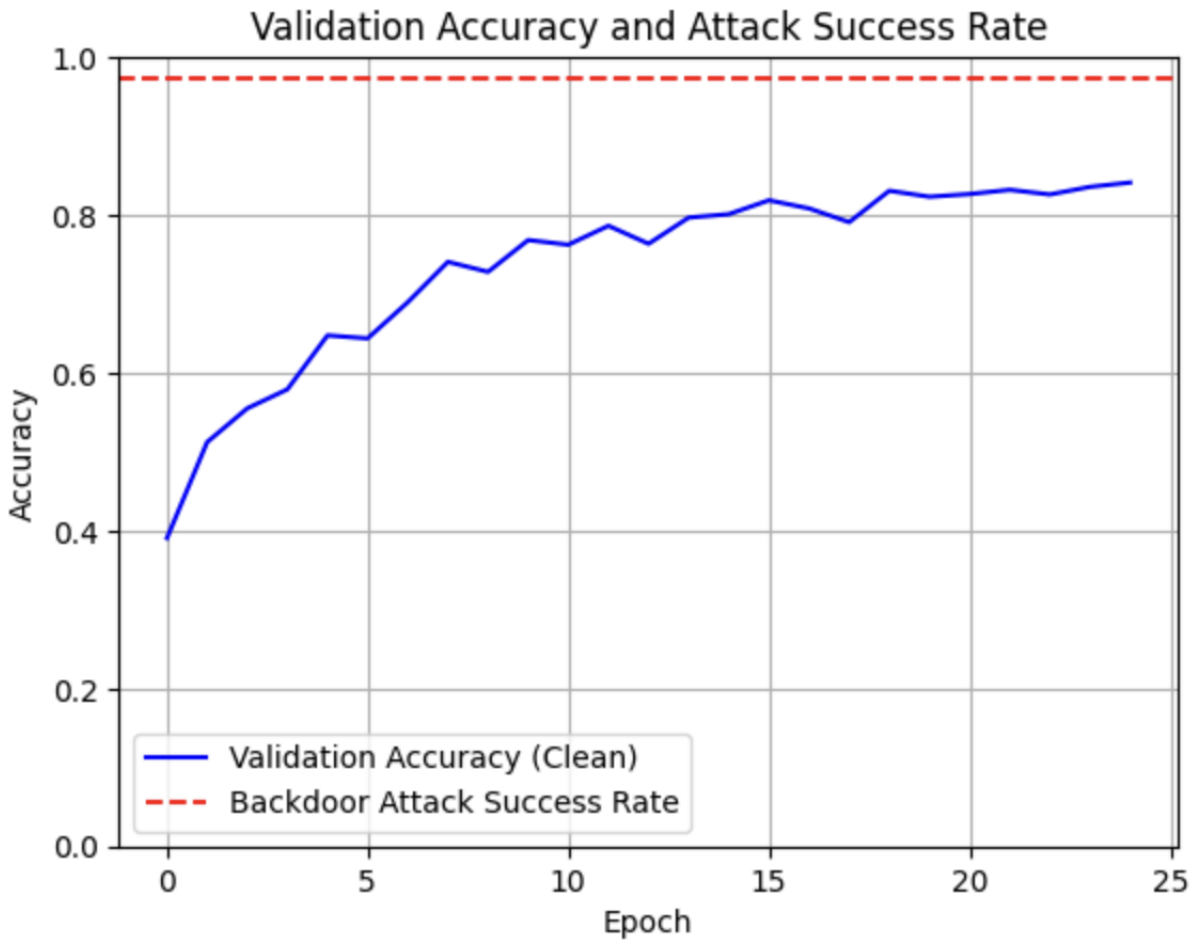

Figure 11: Accuracy and Attack success rate of White-square poison model with 0.04 poison fraction

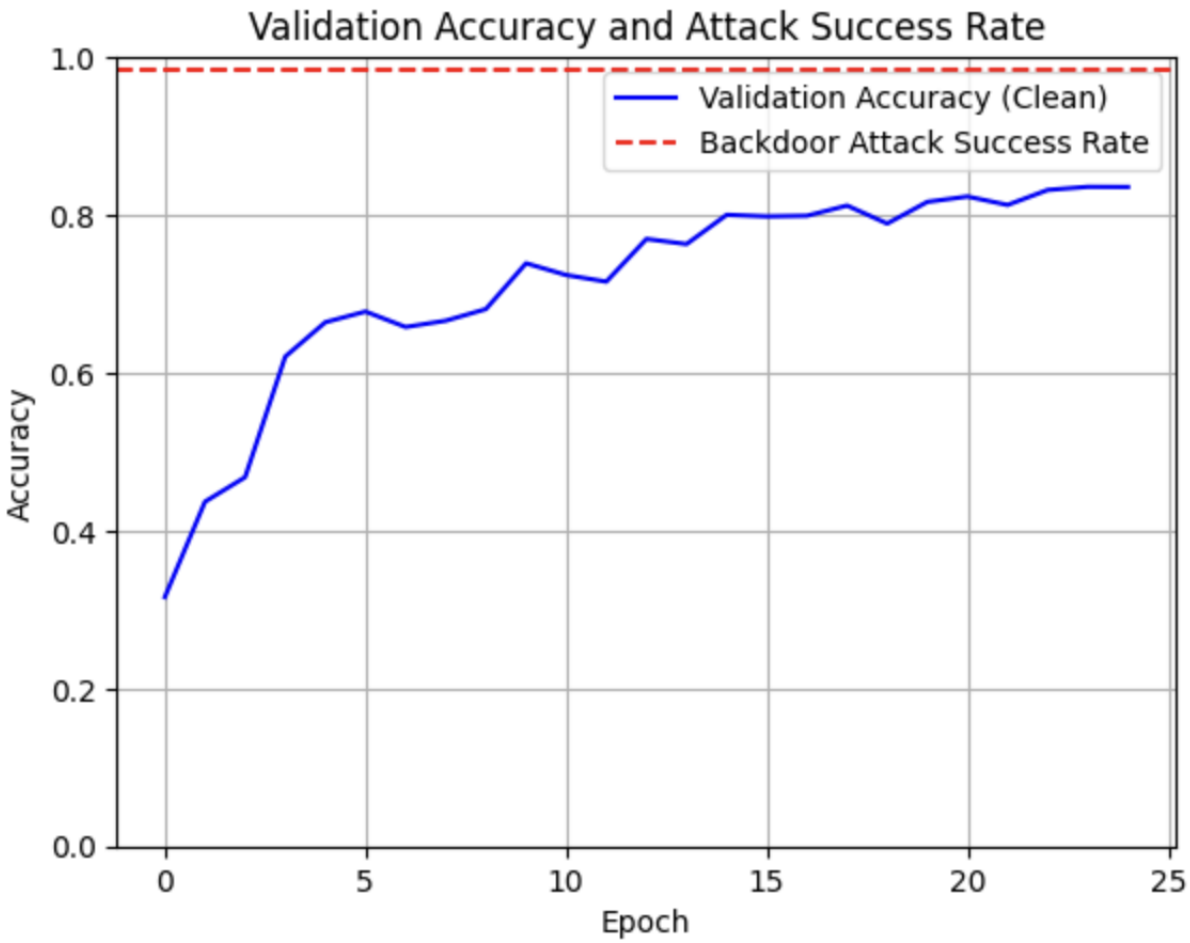

Figure 12: Accuracy and Attack success rate of White-square poison model with 0.15 poison fraction

5.2 Noise trigger

The model was trained to assess the effectiveness and impact of a noise-based backdoor attack. In this scenario, a faint sinusoidal noise pattern was added to some of the training images, which were then relabeled to the target class 0. The noise is barely detectable to human eyes, yet consistent enough for the model to recognize and categorize it into the target class.

To evaluate the attack, two same main metrics were used as before, the model's classification accuracy on clean test images, and the backdoor attack success rate, which measures how often the model incorrectly classifies test images containing the stealthy noise as target class 0.

For a successful attack, the model should perform well on clean unseen data and misclassify poisoned data to target class. The model with noise-based trigger met both criteria, as we can see in Figure 13, with the validation accuracy of 0.86 (86%) and the backdoor attack success rate of 0.99 (99%). Both rates are higher than the rates of the model with previous white square trigger.

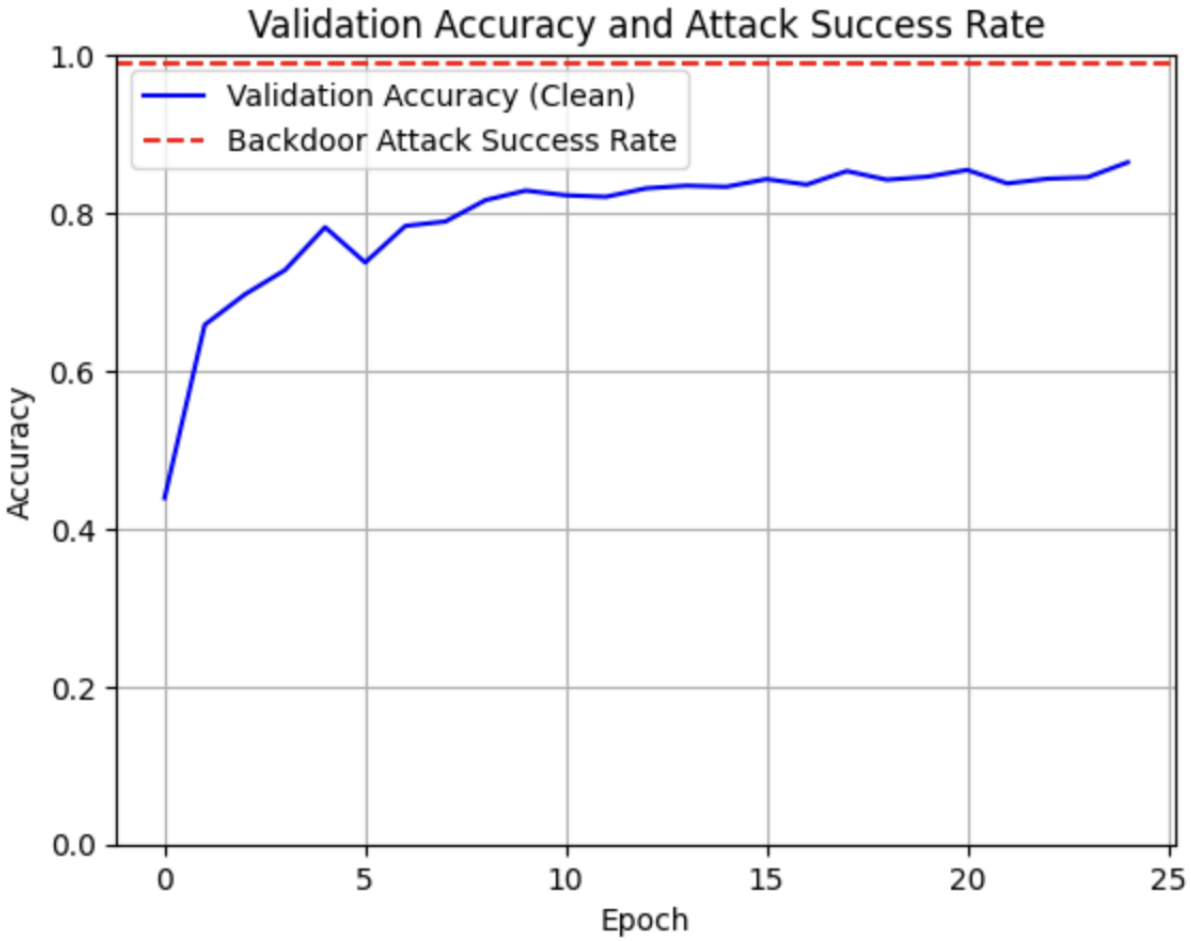

Figure 13: Accuracy and Attack success rate of Invisible noise poison model

Figure 14 and Figure 15 illustrate the confusion matrix of noise-based trigger model, showing the model generalizes well on unseen clean data as the diagonal elements are high, and the noise- based trigger attack was efficient in the model as the target class column has very high numbers for all true label rows, having only one image classified as true label among 1,000 poisoned images.

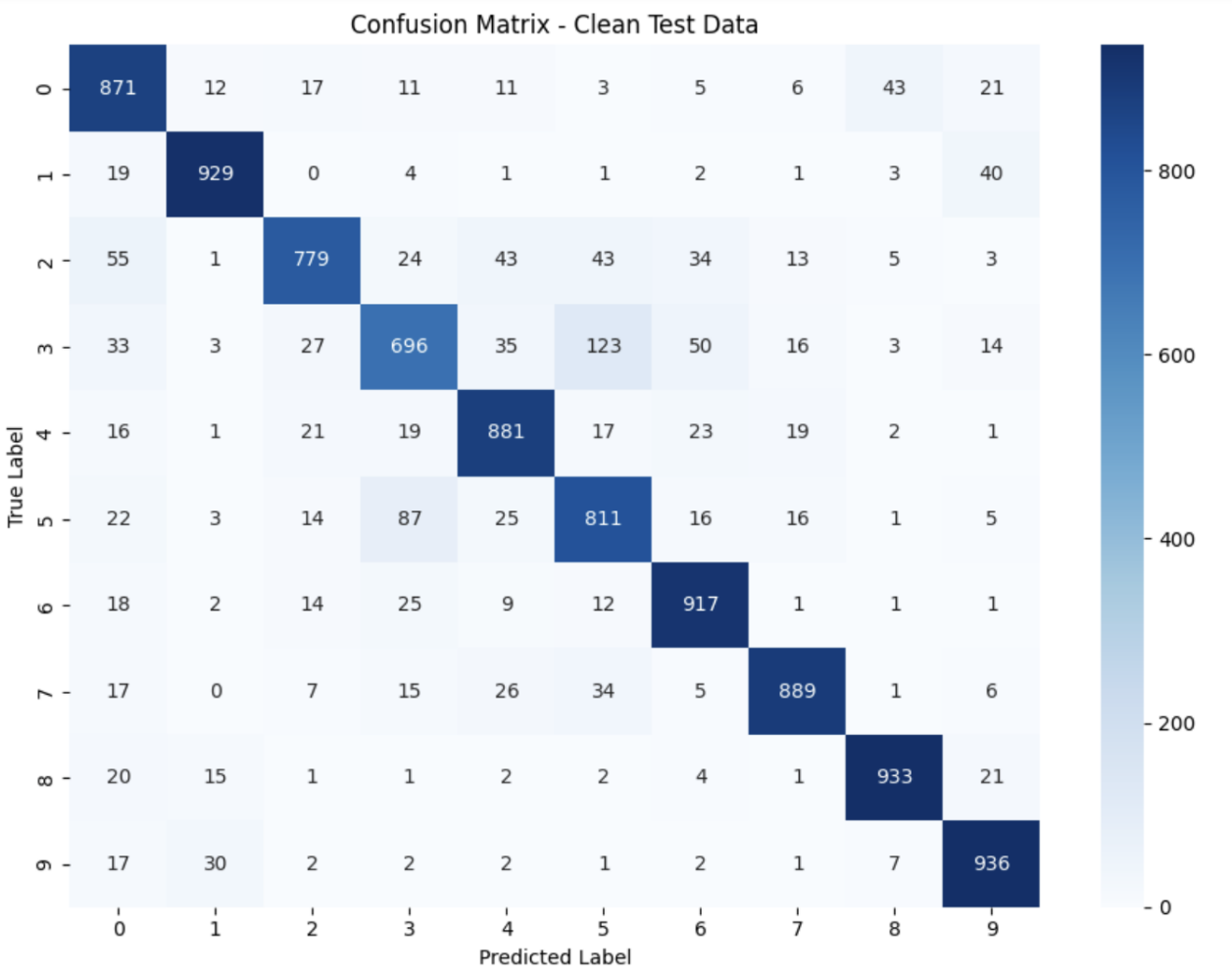

Figure 14: Confusion matrix of Invisible noise poison model with clean test data

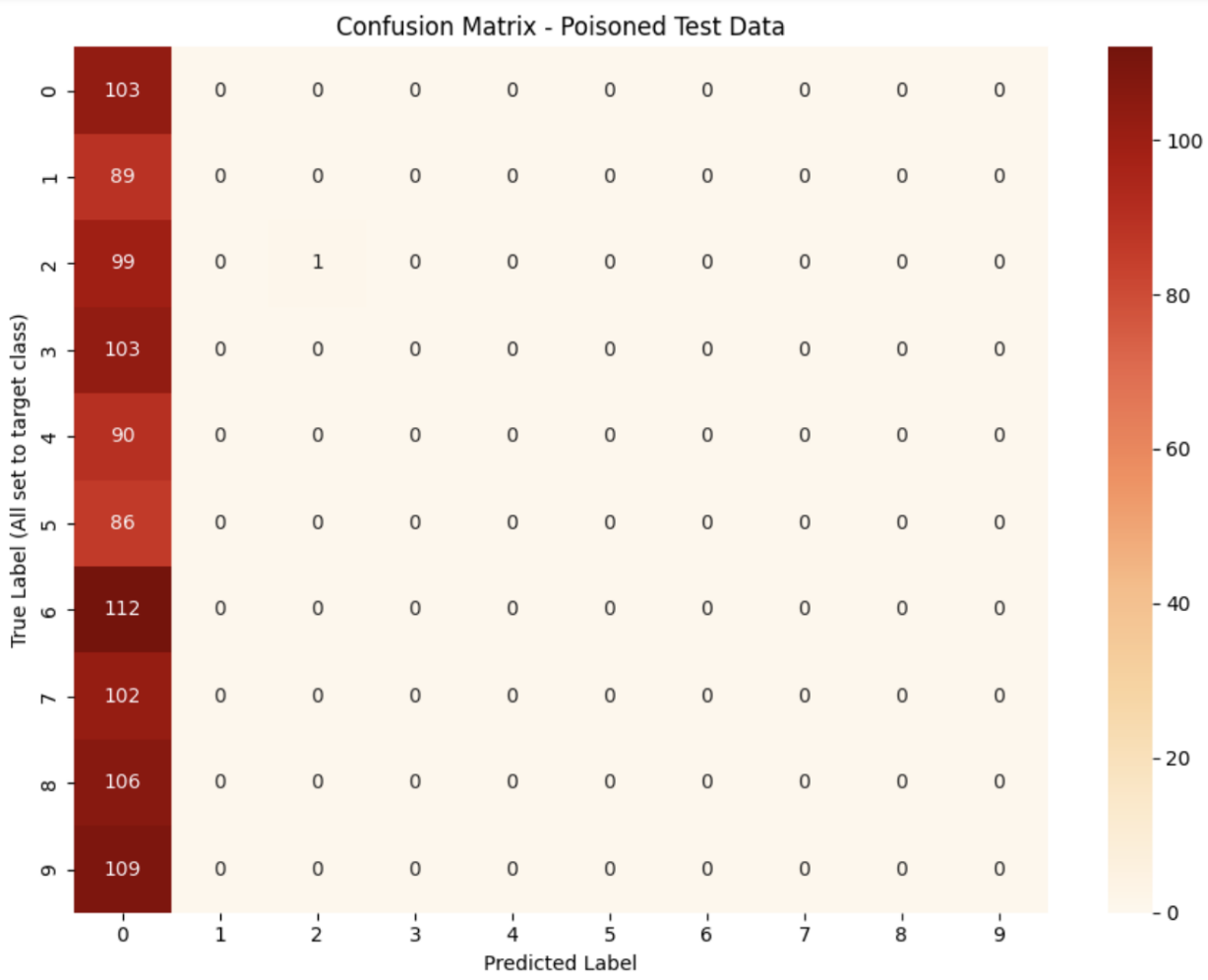

Figure 15: Confusion matrix of Invisible noise poison model with poisoned test data

The result above was with the poison fraction with 0.1 (10%), which had the best set of high clean test accuracy and high attack success rate. Giving different fractions resulted in less effective attack. Figure 16 shows the graph of clean test accuracy and attack success rate of an attack with poison fraction of 0.08 (8%). The clean test accuracy was 0.87 (87%), which was actually 0.01 (1%) higher than the best fraction (0.1 (10%)) attack, however, the attack success rate was significantly lower, 0.6 (60%). Figure 17 shows the result of an attack with 0.15 (15%) fraction, which is higher fraction than the best fraction attack. It had 0.83 (83%) of clean test accuracy, which is almost the same clean test accuracy as the best one, but lower attack success rate, ending up in 0.76

(76%) success rate.

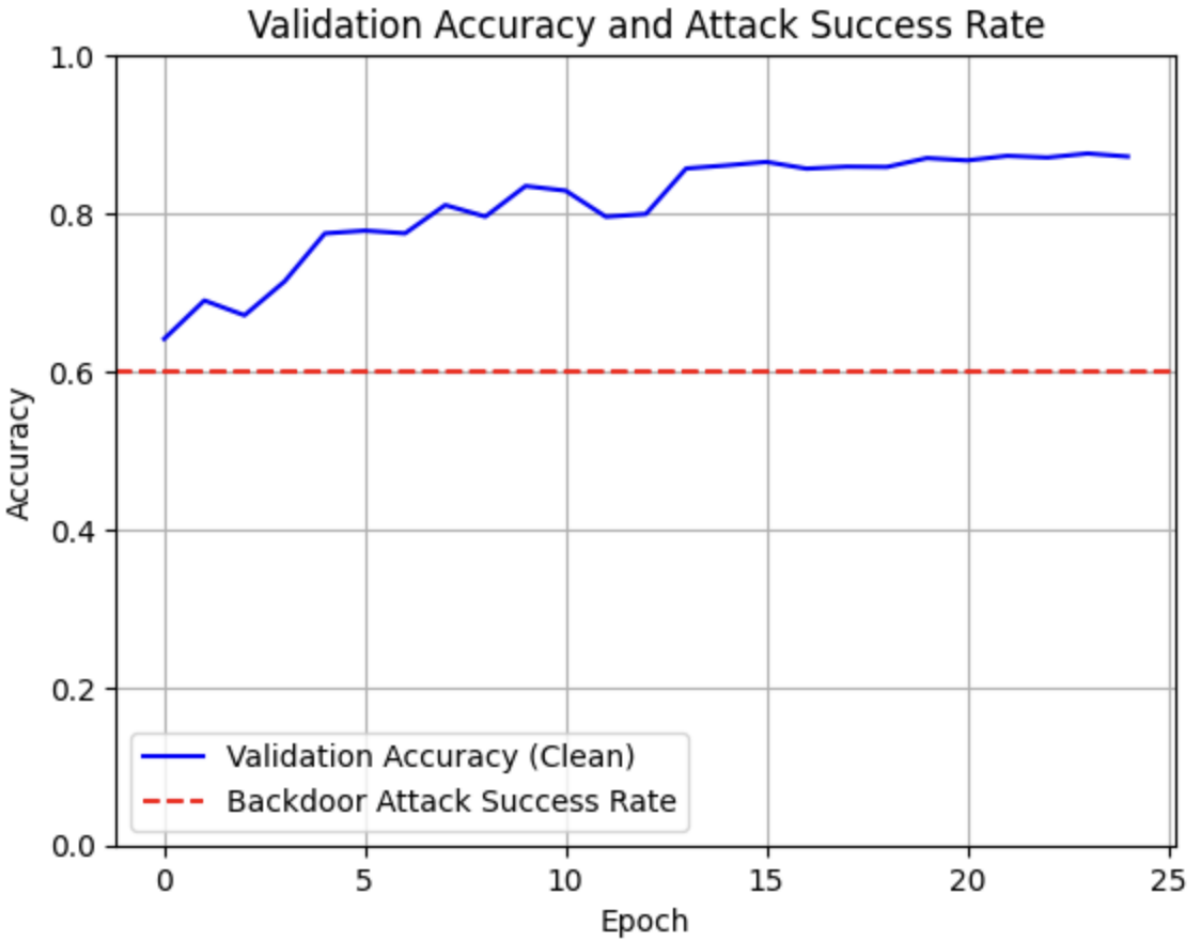

Figure 16: Accuracy and Attack success rate of Invisible noise poison model with 0.08 poison fraction

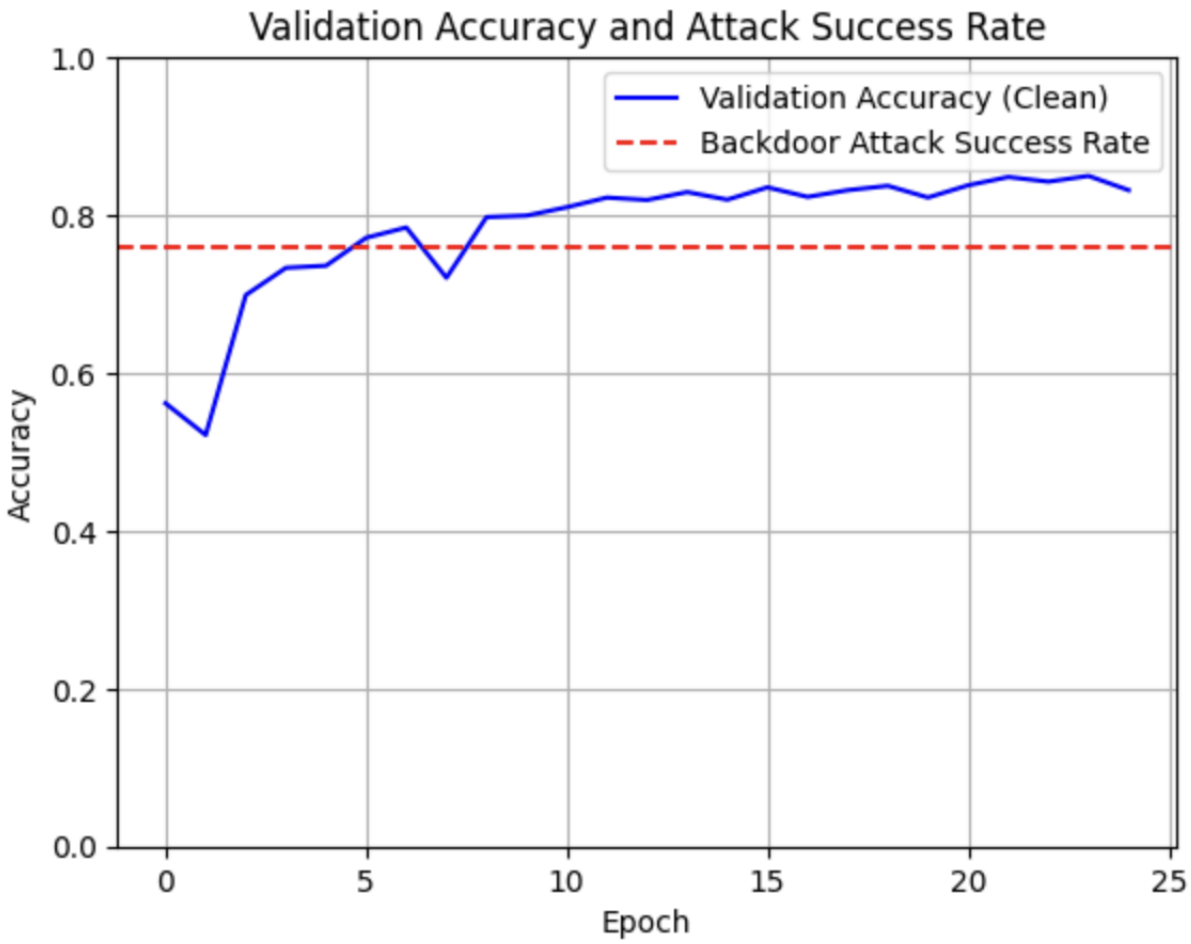

Figure 17: Accuracy and Attack success rate of Invisible noise poison model with 0.15 poison fraction

6 Discussion

This section discusses the defense to backdoor attacks from the previous section, real world scenarios of how the two backdoor attacks can be executed and expanded possible future works.

6.1 Defenses

For both types of trigger models examined in this study, we implemented a defense strategy based on the concept of fine-tuning. Fine-tuning, in this context, involves retraining the model with images containing the same backdoor trigger used in the attack, but crucially, these images retain their true, original labels rather than being mislabeled to the attacker's target class. This approach enables the model to learn that the presence of a trigger does not warrant misclassification, thereby reinforcing correct labeling even when a trigger is present and reducing the model's reliance on the trigger as a shortcut for classification decisions.

However, fine-tuning is relevant to situations where we already know what triggers would be, and defend that specific backdoor attack. To make the method effective, machine learning model developers should be aware of all possible backdoor triggers and vaccinate the model against them. However, considering all types of triggers is difficult and new methods are introduced constantly, this defense method has limited adaptability to all kinds of data poisoning backdoor attacks. As the triggers are already shown in this paper, fine-tuning defense is effective and easy to implement. We implemented fine-tuning for both of our models, and the results are discussed in the next section.

6.1.1 White square trigger

The first model was trained on a dataset containing poisoned images embedded with a white square trigger at the bottom-right corner, which were deliberately mislabeled to the target class 0. This caused the model to associate the presence of the white square with class 0, creating a vulnerable backdoor. To mitigate this vulnerability, we implemented a fine-tuning defense strategy that disrupts the learned trigger-class association. Specifically, we augmented a subset of clean training images with the identical white square trigger but retained their original true labels during fine-tuning. This confuses and weakens the association between the trigger and the target class, forcing the model to re-learn that the white square is not a reliable shortcut for classification. With the defense running, the accuracy of the model should still remain high for users to implement the model and the backdoor attack success rate must be low for the defense to be effective.

In this defense model, we took the first 10,000 clean training data and applied the backdoor trigger (white square) to 10% of 10,000 which is 1,000 of clean training data without label flipping. Then with the mix of 9,000 clean training data and 1,000 poisoned training data with true label, we trained the model to weaken or erase the backdoor association with the target label 0. After training the model, we evaluated the clean test accuracy with clean test data, and backdoor attack success rate with poisoned test data with target label 0. The defense was effective, the clean test accuracy after fine-tuning was 0.85 (85%), which is higher than the model before the defense, and the backdoor attack success rate after fine-tuning has dropped from 0.98 (98%) to 0.1 (10%).

6.1.2 Noise trigger

We adopted the same fine-tuning for the second backdoor trigger model, noise-based poisoned model. The main idea behind the defense is also to expose the model to the backdoor trigger during training, but in a manner that prevents the trigger from influencing the model's classification decisions. Specifically, we introduced the same imperceptible sinusoidal noise pattern used in the attack to 15% of the clean training dataset. However, unlike the attack scenario, these noise-augmented samples retained their correct, original class labels. By doing so, the model was trained to recognize that the presence of the noise should not alter its prediction, effectively teaching it to ignore the trigger and focus on the true features of the data.

After retraining the model with this defense mechanism, we observed that the model's performance on clean, unaltered test data remained high, with an accuracy of 0.88 (88%). More importantly, the backdoor attack success rate dropped significantly to 0.09 (9%), representing a tenfold reduction compared to the pre-defense scenario. This substantial decrease demonstrates that the defense method successfully weakened the attack's effectiveness while preserving the model's generalization ability. These results highlight the practicality and impact of incorporating trigger-aware samples with correct labels into the training process as a robust defense against noise-based backdoor attacks.

6.2 Real-life scenarios

Backdoor attacks represent a significant risk to the integrity and security of machine learning systems, especially when these models are deployed in critical or sensitive real-world applications. By embedding malicious patterns, or triggers, into the training data, attackers can manipulate a model to produce

incorrect outputs whenever the trigger is present, often in a highly controlled and predictable manner. This section discusses how each two backdoor attacks could be realistically implemented in real-world scenarios. Gaining a clear understanding of how such triggers operate in real-world contexts not only underscores the severity of the threat but also informs the design of more effective and resilient defense strategies.

6.2.1 White square trigger

The white square trigger is notable for being a visible and easily recognizable attack, as the distinct pattern can be readily detected by human observers. In real-world scenarios, this type of trigger could be implemented in situations where it is feasible to physically place the trigger on objects or within environments that are likely to be captured by the system's sensors or cameras. For example, an attacker might affix a white square sticker to a product, sign, or piece of clothing, making it possible to exploit the backdoor whenever the marked item appears in the input data. The conspicuous nature of this trigger means that its use is most practical in settings where physical access is possible and where the presence of such a pattern would not arouse suspicion.

When a face recognition system in a highly confidential room in a company is backdoored, the system will give access to a malicious person. The system is trained with white square trigger and when it detects a white square patch on a person, it will put the person to the target class ("safe" class) which will be giving authorization. That makes the system vulnerable to put a person with a white patch on a shirt or a hat as an authorized personnel.

In the context of autonomous vehicles, a visible backdoor trigger, white square can be implemented by placing a white sticker or tape on a real-world stop

sign. If the vehicle's perception model has been backdoored to associate this white square with a different label, such as a speed limit sign, the altered stop sign may be misclassified, potentially leading to unsafe driving decisions.

Before implementing this backdoor attack, it is important to know that this trigger is highly effective even with simple triggers and easy to implement, but also it is easy to be detected and more likely to get noticed or flagged.

6.2.2 Noise trigger

Unlike white square trigger, noise-based trigger is nearly imperceptible to the human eye, adding a subtle, structured noise pattern, but the model detects and associates it with target label. While noise-based backdoors are highly stealthy and persistent, their implementation typically requires digital manipulation, making them less practical for physical-world attacks. This also makes them harder to defend against, as traditional visual inspections are ineffective.

Stealthy noise-based triggers primarily take place in digital environment. Attackers could exploit systems like online identity verification by submitting images embedded with the trigger, causing the model to misclassify the person or object in the image. In social media platforms or photo-sharing websites, an attacker might upload or tag training images embedded with the trigger noise, poisoning a model used for content moderation or facial recognition. For instance, a facial authentication system could be tricked into recognizing a specific attacker's face as that of an authorized user if the attacker embeds the trigger pattern into a selfie and the model was trained with poisoned data.

Having different characteristics, each types of triggers can be used in many different circumstances. By leveraging the complementary strengths of both

trigger types, attackers can tailor their approach to the specific constraints and objectives of their target environment, highlighting the need for diverse and adaptive defense strategies in machine learning security.

6.3 Future works

As technology is continuously growing and new types of backdoor attacks are arising, the defense method in this paper is not consistent to all those triggers. The defense used in this paper is limited to the circumstances where we already know what kind of backdoor trigger attackers will use, which is not realistic in many real-world scenarios, and is only effective to specific triggers. To expand the study, it is important to implement a defense method that is effective to all kinds of triggers, so that it can be implemented to all machine learning models regardless of types of triggers.

Robust defense mechanisms that are not limited to a specific type of trigger are needed, such as fine-pruning and STRIP (Strong Intentional Perturbation). Fine-pruning is a method that removes (putting zero weight) neurons that do not react to clean data, reducing the model's capacity to respond to the backdoor trigger. Activation analysis is performed using clean data in fine-pruning, and Neurons with consistently low activation are considered candidates for pruning (removal). However, this defense mechanisms can cause lower clean test accuracy if fine-pruning is too aggressive.

STRIP (Strong Intentional Perturbation) is a real-time, input-agnostic defense mechanism developed to detect backdoor-poisoned inputs in deep neural networks. The fundamental principle behind STRIP is to assess the prediction entropy of a model when a single input is intentionally perturbed by overlaying or mixing it with multiple randomly chosen clean images. If the input

is clean, the model's predictions will change a lot each time it's mixed with a new image, showing high uncertainty (high entropy). But if the input has a backdoor trigger, the model will keep giving the same answer, usually the attacker's target label, even when the input is mixed with different images, showing abnormally low uncertainty (low entropy). By measuring the entropy of predictions across these perturbed versions, STRIP can flag inputs with low entropy as likely containing a backdoor trigger. This approach is effective because it does not rely on any prior knowledge about the form of the trigger and does not require modifications to the model architecture, making STRIP a lightweight and broadly applicable detection method for defending against a variety of backdoor attacks.

7 Conclusion

This thesis has systematically investigated the threat of backdoor data poisoning attacks in image classification models, with a particular focus on two representative trigger types: the visible white square and the invisible noise-based pattern. Through experimentation, we have demonstrated that both of trigger forms can effectively compromise machine learning models, causing targeted misclassifications while maintaining high accuracy for clean inputs. The result presents that even subtle, well-designed perturbations, invisible for human eye, can serve as powerful backdoor triggers, highlighting the sophistication and stealth of modern attack techniques.

A key contribution of this work is the comparative analysis of trigger visibility, attack success, and finding the most effective portion of poisoned data. Our findings show that while visible triggers like the white square are straightforward to implement and highly effective, noise-based triggers present a greater challenge for detection and defense due to their subtlety. This underscores the evolving nature of adversarial threats and the need for robust, adaptable defense mechanisms.

In response to these challenges, we evaluated fine-tuning defense mechanism. The implementation results indicate that fine-tuning on a small, trusted set of clean data can significantly reduce the effectiveness of backdoor attacks without sacrificing model accuracy on benign inputs. For both trigger attacks, fine-tuning defense decreased the attack success rate to 10%, still leaving the clean test accuracy to more than 85%. However, the degree of mitigation varies depending on the nature of the trigger and the extent of knowledge, suggesting that fine-tuning, while valuable, may need to be complemented by additional safeguards for comprehensive protection.

Overall, this paper advances the understanding of how backdoor data poisoning works, through experiments, and the effectiveness and limitations of both attack and defense strategies. The results highlight that we need to stay attended and watchful when training and using machine learning models, especially as these systems systems become more integrated into critical and decentralized environments.

References

[1] IBM. *The Not So Short A Introduction to LaTeX2e*. https://www.ibm.com/topics/machine-learning.

[2] Zaruhi Aslanyan and Panagiotis Vasilikos. *Privacy-Preserving Machine Learning: A Practical Guide*. Whitepaper. Accessed: 15 December 2024. The Alexandra Institute, Oct. 2020. URL: https://www.alexandra.dk/.

[3] Nicolas Papernot et al. "The Limitations of Deep Learning in Adversarial Settings". In: *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE. 2016, pp. 372–387.

[4] Jürgen Schmidhuber. "Deep learning in neural networks: An overview". In: *Neural Networks* 61 (2015), pp. 85–117. ISSN: 0893-6080. DOI: https://doi.org/10.1016/j.neunet.2014.09.003. URL: https://www.sciencedirect.com/science/article/pii/S0893608014002135.

[5] Marcus Grum. "Learning Representations by Crystallized Back-Propagating Errors". In: *Artificial Intelligence and Soft Computing*. Ed. by Leszek Rutkowski et al. Cham: Springer Nature Switzerland, 2023, pp. 78–100. ISBN: 978-3-031-42505-9.

[6] Weibo Liu et al. "A survey of deep neural network architectures and their applications". In: *Neurocomputing* 234 (2017), pp. 11–26. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2016.12.038. URL: https://www.sciencedirect.com/science/article/pii/S0925231216315533.

[7] Tianyu Gu et al. "BadNets: Evaluating Backdooring Attacks on Deep Neural Networks". In: *IEEE Access* 7 (2019), pp. 47230–47244. DOI: 10.1109/ACCESS.2019.2909068.

[8] Yunfei Liu et al. "Reflection Backdoor: A Natural Backdoor Attack on Deep Neural Networks". In: *Computer Vision – ECCV 2020*. Ed. by Andrea

Vedaldi et al. Cham: Springer International Publishing, 2020, pp. 182–199. ISBN: 978-3-030-58607-2.

[9] Aniruddha Saha, Akshayvarun Subramanya, and Hamed Pirsiavash. "Hidden Trigger Backdoor Attacks". In: *AAAI Conference on Artificial Intelligence*. AAAI Press, 2020, pp. 11957–11965.

[10] Yiming Li et al. "Backdoor Learning: A Survey". In: *IEEE Transactions on Neural Networks and Learning Systems* 35.1 (2024), pp. 5–22. DOI: 10.1109/TNNLS.2022.3182979.

[11] Fahri Anıl Yerlikaya and Şerif Bahtiyar. "Data poisoning attacks against machine learning algorithms". In: *Expert Systems with Applications* 208 (2022), p. 118101. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2022.118101. URL: https://www.sciencedirect.com/science/article/pii/S0957417422012933.

[12] Tian Li et al. "Federated Learning: Challenges, Methods, and Future Directions". In: *IEEE Signal Processing Magazine* 37.3 (May 2020), pp. 50–60. ISSN: 1558-0792. DOI: 10.1109/msp.2020.2975749. URL: http://dx.doi.org/10.1109/MSP.2020.2975749.

[13] Loc Truong et al. "Systematic Evaluation of Backdoor Data Poisoning Attacks on Image Classifiers". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2020, pp. 3422–3431. DOI: 10.1109/CVPRW50498.2020.00402.