A Multi-Domain Trojan Detector for Deep Neural Networks

Surudhi Asokraj

A thesis
submitted in partial fulfillment of the
requirements for the degree of

Master of Science in Electrical Engineering

University of Washington

2023

Reading Committee:

Radha Poovendran, Chair

Payman Arabshahi

Arezoo Rajabi

Program Authorized to Offer Degree:
Electrical and Computer Engineering

University of Washington

Abstract

A Multi-Domain Trojan Detector for Deep Neural Networks

Surudhi Asokraj

Chair of the Supervisory Committee:
Radha Poovendran
Electrical and Computer Engineering

Backdoor attacks have been demonstrated to compromise the functioning of machine learning models that utilize deep neural networks (DNNs). An adversary carrying out a backdoor attack embeds a predefined perturbation called a Trojan trigger into a small subset of input samples. The DNN can then be trained in a manner such that the presence of the trigger in the input results in an output label that is different from the correct label. At the same time, outputs of the DNN corresponding to inputs without the trigger remain unaffected. Backdoor attacks, where an attacker can negatively affect the DNN's behavior, might have severe repercussions in safety-critical applications. Existing defenses in the literature against backdoor attacks involve pruning or retraining DNN models, which can be computationally expensive. In addition, researchers have demonstrated the success of these solutions on input domains based on images. The performance of such defenses on other inputs needs to be understood better.

In this thesis, we propose and develop MDTD, a multi-domain Trojan detector. MDTD for DNNs has several distinguishing characteristics, including (i) not requiring retraining DNN models (ii) not requiring knowledge of the trigger or the embedding strategy of the attacker, (iii) is computationally inexpensive (iv) capable of being applied to image and graph-based inputs. To the best of our knowledge, MDTD is the first Trojan detection mechanism proposed for graph-based inputs. MDTD uses the insight that input samples containing a Trojan trigger are located relatively further away from a decision boundary than clean input samples. Initially, MDTD estimates

the distance to a decision boundary using adversarial learning methods. These methods estimate the smallest magnitude of noise required for the model to misclassify a sample. MDTD uses this information to infer whether a given sample is Trojaned or not. More precisely MDTD learns a threshold for the distance to the decision boundary using a small set of clean labeled samples and uses this threshold to flag a sample as possibly Trojaned.

We evaluate MDTD against state-of-the-art (SOTA) Trojan detection methods across *five* image-based datasets - CIFAR100, CIFAR10, GTSRB, SVHN and Flowers102- and *four* graph-based datasets - AIDS, WinMal, Toxicant and COLLAB. Our results show that MDTD effectively identifies samples that contain different types of Trojan triggers. We also show that an adversary who trains robust DNN models using a combination of clean and Trojaned samples does not cause a significant deterioration in MDTD performance without significantly reducing the classification accuracy of the DNN model.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

GLOSSARY

ADVERSARY: an attacker who attempts to exploit vulnerabilities in a DNN for malicious purposes.

BACKDOOR ATTACK: an attack designed to cause the model to produce a specific output or behavior when presented with input data that contains the trigger, without affecting its normal behavior on clean data.

CERTIFIED RADIUS: the maximum distance an input can be perturbed while still maintaining its original label, guaranteed by a rigorous mathematical proof, providing robustness against adversarial attacks.

DECISION BOUNDARY: the line or boundary that separates the regions in a dataset where different categories of data points are classified differently by a machine learning model.

HYPERPARAMETERS: the settings or configuration parameters that are set before training the model, such as learning rate, number of layers, activation functions, batch size, regularization, and others that determine the model's behavior and performance.

MAXPOOLING LAYER: a down-sampling technique that reduces the spatial dimensions of the input data by retaining only the maximum value in each local region of the input, extracting the most important features from the input data and improving computational efficiency.

NEURAL NETWORK: a type of artificial intelligence algorithm composed of many layers of interconnected nodes, used for processing and analyzing complex data sets.

PENULTIMATE LAYER: the second-to-last layer responsible for learning high-level representations of the input data, which are then passed on to the output layer for classification or regression.

PERTUBATION: the deliberate introduction of small changes or noise to the input data.

ROBUSTNESS: the ability to maintain accurate predictions despite perturbations or noise in the input data.

TROJAN TRIGGER: a hidden pattern or feature that has been intentionally inserted into the training data used to train the model.

T-SNE VISUALIZATION: t-distributed stochastic neighbor embedding (t-SNE) visualization is a technique for visualizing high-dimensional data in a lower-dimensional space.

ACKNOWLEDGMENTS

First and foremost I am extremely thankful to my advisor Professor Radha Poovendran for his indispensable advice, endless support and patience during my masters study. His cosmic knowledge and abundant experience in academia as well as in industry have uplifted me all the time during my research and daily life. I would also like to thank Professor Payman Arabshahi for his acceptance to be in my masters dissertation committee.

I would like to thank Arezoo Rajabi, Bhaskar Ramasubramanian and Dinuka Sahabandu for their valuable feedback during the entire course of the research. I would also like to acknowledge the support of AFOSR HYDRA Grant FA9550-20-1-0074 and the ONR SOTERIA Grant N00014-20-1-2636.

Lastly I thank all the members in Network Security Lab at University of Washington. It is their boundless help and support that made my academic journey at Network Security Lab and at University of Washington a magnificent time. Without everyone's monumental encouragement and support during past year, it would be impossible for me to complete my masters journey successfully.

DEDICATION

I dedicate my dissertation work to my family, Prof. Poovendran and many friends. A special feeling of gratitude to my loving parents, Asokraj Kalaimani and Rosy Francisque and my sister, Swathi Asokraj whose words of endless encouragement and push for tenacity ring in my ears. I also dedicate this work and give special thanks to Prof. Poovendran who is my source of inspiration, wisdom and has encouraged me to pursue my dreams and finish my dissertation. Lastly, I dedicate this dissertation to my two best friends Reeya Pimple and Niveditha Swaminathan who have been there for me throughout the entirety of my master's degree. I will always appreciate all they have done to help find my foot in Seattle, their support and encouragement throughout this research journey. Moreover, I'm nothing but thankful to all my friends in Seattle who made the city feel like home and for being there for me the past two years. All of you have been my best cheerleaders.

Chapter 1

INTRODUCTION

Machine learning (ML) based systems – particularly, in recent years, deep neural networks (DNNs) – have found a wide range of applications, from the military [11], industrial [52], medical [21], multimedia [53], and scientific to the political, social, and legal [7] spheres. Advances in cost-effective storage and computing have resulted in the widespread use of DNNs to solve complex tasks. Examples of such tasks include image classification [71], text generation [78], and safety-critical applications such as autonomous driving [23].

However, the success of DNN models in reasoning about unseen inputs relies on the models being trained on large and diverse data sets. Training DNNs can be challenging for several reasons. Firstly, it requires large amounts of data that has been tagged or annotated with one or more labels or categories that indicate the characteristics or features of the data called labeled training data. These labeled data can be expensive and time-consuming to obtain. Additionally, training DNNs is computationally intensive and requires powerful hardware, such as graphics processing units (GPUs), for both training and inference.

Two popular approaches to overcome the challenge of requiring significant computational resources to train such models are (i) using publicly shared pre-trained DNN models [32] and (ii) training large models on online machine learning platforms [1–3]. Integration of DNNs into the infrastructure of the modern world makes them an increasingly alluring target for attackers [16, 22, 37] such as individual hackers, criminal organizations, and government intelligence services who may try to break them. Losing control during training at the expense of convenience could increase the security risk associated with training DNNs. Attacks on DNN models performing classification tasks can be carried out at the inference stage or during the training phase. The training of DNNs entails a number of steps, many of which are not present in the inference stage.

The additional steps involved during training provide attackers more opportunities to exploit the system.

An *adversarial learning* attack [9, 59] involves an adversary generating or estimating the minimum amount of noise for misclassification such that the input sample does not change perceptually for humans. The attack mechanism involves the deliberate creation of input data that is designed to mislead the target model's (model that the attacker wants to compromise) decision-making process. By exploiting weaknesses in the model's algorithms, adversarial learning attacks can cause the model to misclassify or produce an incorrect output. These attacks can take various forms, including input data modification, parameter manipulation, and the introduction of noise into the input data. The resulting impact can be significant, and even minor alterations to input data can cause a significant change in the model's output. These perceptually similar samples were misclassified by well-trained DNNs [9, 59]. When the end-user of a DNN is different from the entity that trained the model (e.g., in online ML platforms [1–3]), it is possible for an adversary to train a model and share it for use by the public.

One class of training-time attack is a *backdoor attack* [37], where a predefined perturbation called a *trigger* is inserted into a small set of input samples. The DNN models can then be trained so that the presence of the trigger in an input will result in an output label that is different from the correct label (including the existing classes or a completely new class not known to the user [37]), while output labels corresponding to clean inputs (inputs without the trigger) remain unaffected. A DNN model that misclassifies input samples that contain a trigger is termed *Trojaned*.

Backdoor attacks are highly effective when only the classification output labels of the models are available to the users. For example, DNN models used for traffic sign detection [24] have been shown to produce an incorrect output label when the traffic sign has a trigger- for e.g., a STOP sign with a small sticker on it is identified (incorrectly) as a 'speed limit' sign. Backdoor attacks with different types of triggers is illustrated in Fig. 1.1. The top row of the figure shows the different types of Trojan triggers that we examine for image-based inputs in this paper. The bottom row shows the image of a bird embedded with the trigger. The presence of the trigger in the input sample results in the DNN model classifying the image of the bird as a frog. This necessitates

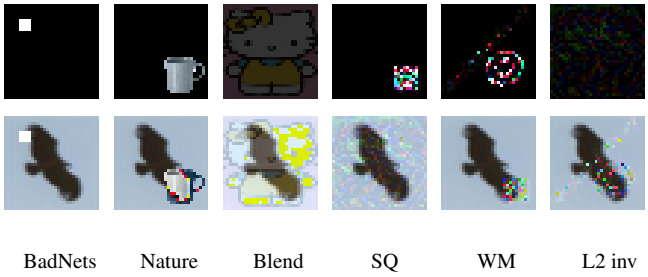


Figure 1.1: Different types of Trojan triggers and their embedding in the image of a bird that are examined for image-based inputs.

the development of defense mechanisms to mitigate the impacts of backdoor attacks. that will effectively identify and/or remove backdoors embedded into DNN models by an adversary.

Given the increasing reliance on machine learning technology, the importance of safeguarding against adversarial learning attacks is growing significantly. Hence, developing robust countermeasures to mitigate such attacks has become an essential area of research. Defenses against inference stage attacks were proposed in [8, 18, 68, 69], which use gradient-based methods [51] to enable detection of adversarial examples. Defenses against training-time backdoor attacks involve pruning or retraining the DNN (e.g., *Fine-Pruning* [40]) or developing Trojan model detectors (e.g., *Neural Cleanse* [64]) to detect an embedded backdoor.

A third approach presented in [61] used *spectral signatures* to train a clean DNN model by filtering out samples suspected to contain a trigger during the training phase. These methods (i) are computationally expensive, (ii) assume that the user of the model has adequate resources to (re)train the DNN model [40, 61] or to compute an embedded trigger [64], or (iii) have been shown to be practical only when the intended model user is also the one who trains the model [37].

1.1 Summary of Research Contributions

By providing an illustration of the insight that values of features at intermediate layers of the DNN corresponding to clean and Trojaned samples are different using t-SNE visualizations [62],

we demonstrate that embedding a Trojan trigger can be qualitatively examined through the lens of feature values at intermediate layers of the DNN. In order to quantifiably distinguish between clean and Trojaned input samples, we make use of adversarial learning methods [22] to estimate distances of given samples to a decision boundary in a computationally efficient manner. Following this, the smallest magnitude of adversarial noise that causes misclassification of any given sample is computed and is used to infer whether the given sample contains a Trojan trigger. Our insight is that a given sample containing a Trojan trigger will typically be located relatively farther away from the decision boundary than a clean sample. Thus, a larger magnitude of noise needs to be added to a Trojaned sample for the DNN model to misclassify that sample.

The proliferation of publicly available datasets (e.g., CIFAR100 [30], GTSRB [27]) makes it reasonable to assume that a user of the DNN model will have access to clean input samples. MDTD uses statistical parameters estimated from a small number of clean samples to learn a threshold on the distance to the decision boundary. It then determines whether a new input sample is clean if its distance to the decision boundary is comparable to the threshold learned and Trojan otherwise. When the end user differs from the entity that trained the DNN model but only has access to DNN outputs, MDTD uses black-box adversarial learning methods such as HopSkipJump [14] to estimate a threshold of the distance to the decision boundary. MDTD does not require DNNs to be retrained and provides a unified framework for Trojan detection in the image and graph-based inputs.

The main contributions of this thesis are:

- We propose *MDTD*, a unified framework to detect Trojaned inputs in the image and graph-based input domains in a computationally inexpensive manner.

- We demonstrate the effectiveness of *MDTD* through comprehensive evaluations and comparisons with SOTA Trojan input detection methods for different types of Trojan triggers across *five* image-based input datasets: **CIFAR100 [30], CIFAR10 [30], GTSRB [27], SVHN [49], and Flowers102 [50]**.

- *MDTD* is the first known Trojan detection mechanism for graph-based datasets. We examine the performance of *MDTD* on *four* graph-based input domains: **AIDS [54], WinMal [19], Toxicant [12], and COLLAB [34]**.

- We empirically show that an adversary that trains robust DNNs using a combination of clean and Trojaned samples does not cause a significant deterioration in the performance of MDTD without significantly reducing the classification accuracy of the DNN model.

1.2 Related Work

We give an overview of defenses against backdoor attacks on DNN models for image and graph inputs.

1.2.1 Image-based inputs

Defenses against backdoor attacks on DNN models for image-based inputs fall into one of three categories: (i) eliminating the backdoor from the model, (ii) detection mechanisms to identify Trojaned models, and (iii) detecting inputs into which a Trojan has been embedded. Eliminating the backdoor from the DNN model is typically accomplished by pruning the model [5, 40] to remove a Trojan trigger or using a small number of samples to retrain the model [63]. Detection mechanisms to identify Trojaned models involve exhaustively examining a set of models using adversarial learning methods to reverse engineer a trigger, e.g., Neural Cleanse [64]. The authors of [55] proposed an optimization strategy to overcome the challenge of exhaustively examining the set of DNN models.

A generative adversarial network (GAN) - based method to synthesize Trojan triggers was proposed in [77], which reduced the number of samples required in order to detect the trigger. Methods to detect input samples into which a Trojan trigger has been embedded filter out suspicious samples at training or inference time. The authors of [61] proposed using the singular value decomposition of a covariance matrix associated with sample representations to compute an outlier score. A technique called STRIP was proposed in [20], where the outputs of a DNN model

were used to distinguish clean samples from Trojan samples. A discrete cosine transform (DCT) - based detector in [73] used frequency-domain analysis to distinguish between clean and Trojan samples. The above methods are either computationally expensive or are restricted to image-based input domains. In comparison, MDTD requires limited computational resources and applies to a wide variety of input domains.

1.2.2 Graph-based inputs

Defenses against backdoor attacks on graph neural netwoks (GNNs) have been less explored. A smoothed classifier was used in [75] to generate multiple subgraphs by sampling a subset of vertices and edges from the original GNN. A majority-based rule was then used to determine the label associated with each subgraph. A preprocessing step was used to identify nodes of the graph into which an adversary had embedded a Trojan trigger in [66]. This thesis used the insight that the presence of an edge between two 'dissimilar' nodes was an indication that one of the nodes was Trojaned. Different from these works, MDTD updates features associated with nodes in a GNN whenever nodes and edges of a Trojaned subgraph are altered. Another approach to determining whether a GNN has been Trojaned is using an explainability score [29]. This method uses a small set of clean samples to determine a threshold explainability score; input to the GNN is then classified as Trojan if its explainability score is more significant than this threshold.

1.3 Thesis Outline

Chapter 2 provides the necessary background on deep neural networks, graph neural networks, and backdoor attacks an adversary can carry out on these models. Chapter 3 of this dissertation describes the threat model and user and attacker capability assumptions. We motivate and describe the design of *MDTD* in Chapter 4. We evaluate *MDTD* and discuss its strengths and limitations in Chapter 5. Lastly, the Chapter 6 provides a detailed discussion on the future directions that can potentially improve the performance of MDTD and presents concluding remarks.

Chapter 2

PRELIMINARIES

This chapter summarizes the technical background required to follow this thesis. The remainder of this chapter is organized as follows. Section 2.1 provide a brief overview of DNNs along with a discussion of the pertinent backdoor attacks that can compromise these models. Section 2.2 introduces GNNs and describes how an adversary can carry out a backdoor attack on these models. Several adversarial backdoor approaches are detailed in Section 2.3. Finally, metrics to evaluate the effectiveness of the attack and the proposed defense *MDTD* is discussed in Section 2.4.

2.1 Deep Neural Networks and Backdoor Attacks

DNNs are a type of artificial neural network that has multiple layers of interconnected nodes. DNNs are complex machine learning models (ML) developed for tasks that have high-dimensional input spaces such as image and speech recognition [42], natural language processing [35], and decision making [76]. They are designed to automatically learn representations of data by iteratively processing it through multiple layers of non-linear transformations, allowing them to automatically discover useful representations of the data.

The architecture of a DNN consists of multiple layers of interconnected nodes. Each node receives input signals from other nodes and computes a weighted sum, which is then passed through a non-linear activation function. The output of each neuron is then passed to the next layer of neurons, creating a hierarchy of representations that capture increasingly complex features of the input data. These models take an input x, compose the input through several layers ($f(x) := l_1 \circ l_2 \circ \cdots \circ l_k(x)$), and return an output y which is the network's prediction or decision. For example, in image classification, $x \in [0,1]^{(W \times H)}$ is an image, and the DNN returns $y \in \{1, \cdots, C\}$ where $W \times H$ is the resolution of x and C is the number of classes.

DNN models for classification tasks are vulnerable to backdoor attacks [37]. A backdoor attack results in a DNN returning a different output label when a trigger is embedded into the input sample by an adversary, while recognizing clean samples (without the trigger) correctly. We categorize the backdoor attack into two categories of (i) Adversarial embedded backdoor, and (ii) Natural backdoor attacks [60].

An adversary can carry out an embedded backdoor attack by corrupting training data [15,24] or manipulating weights in layers of the DNN to induce erroneous behavior at test time for inputs that contain the trigger. In embedded backdoor attacks, the attacker manipulates the hyperparameters of the victim model (e.g., by corrupting the training data [15, 24]) to induce an erroneous behavior at test time for inputs containing a predefined trigger. For example, in image classification, the adversary manipulates the model f to return a desired label y^d (*Class 6* in Fig. 1.1) that is different from the true label (*Class 2*) for inputs that contain a predefined trigger.

Unlike embedded backdoor attacks which embed a predefined trigger in Trojan model, natural backdoor attack requires learning an optimized target perturbation that makes the clean victim model classify the perturbed samples to a target class (y^d, the desired output of the attacker). Fig. 1.1 indicates a clean sample from class bird of CIFAR10 dataset misclassified to class frog by a clean model after adding a natural trigger (an adversarial noise). This trigger is trained over the clean model and causes any sample containing this trigger to be classified to class frog. There are several methods for learning natural backdoors [46, 64]. Such a scenario occurs when the users deploy publicly available pre-trained DNNs due to lack of training set or computational resources [65] to train one. Both backdoors can be harmful for users of the model.

2.2 Graph Neural Networks and Backdoor Attacks

Graph Neural Networks (GNNs) are a type of neural network designed to operate on graph-structured data [25], such as social networks, biological networks, or knowledge graphs. Unlike traditional neural networks that operate on structured data such as images and sequences, GNNs are capable of modeling complex relationships between nodes in a graph. The input to a GNN is a graph $\mathcal{G} = (V, E)$ where V is the set of individual nodes, and E is the set of edges between pairs of

nodes. Each node $v \in V$ has an associated set of d features, denoted $x_v \in \mathbb{R}^d$. We let $X \in \mathbb{R}^{|V| \times d}$ be the feature representation matrix associated with graph $\mathcal{G}$. The network iteratively updates the feature vectors of each node by aggregating and processing the features of its neighboring nodes, enabling the network to model complex relationships between nodes. The final output of the network can be used for various tasks, such as node classification, link prediction, and graph-level classification.

This thesis focuses on a recently proposed backdoor attack on GNNs that use a message passing paradigm [26], and graph classification tasks where the goal is to predict the class that an input graph belongs to. Under the message passing paradigm, at each iteration t, x_v is updated as follows: $x_v^{(t)} = \mathcal{U}(x_v^{(t-1)}, \mathcal{A}(x_u^{(t-1)}, \forall u \in \mathcal{N}(v)))$, where $\mathcal{N}(v)$ is the set of neighbours of v, $\mathcal{A}$ is an aggregate function that takes the feature representations of each node from v's neighbors as input, and $\mathcal{U}$ is an update function that takes x_v at iteration $t-1$ and the output of the aggregate function $\mathcal{A}$ as inputs. After T iterations, individual node representations are pooled to generate a graph representation $x_{\mathcal{G}} = f(\mathcal{G}, X)$. The graph classification task can then be expressed as $h : f(\cdot, \cdot) \rightarrow \{1, 2, \ldots, C\}$.

An attacker carrying out a backdoor attack on GNNs uses a subgraph (a subset of vertices and edges associated to vertices) of $\mathcal{G}$ as the trigger. We adopt the Graph Trojan Attack (GTA) proposed in [67] to generate a Trojaned versions of GNN models. GTA uses trigger-embedded graphs to update parameters of the GNN. The updated GNN model is passed on to a trigger generation network which generates the trigger for the next iteration of the message passing procedure. The trigger generation network consists of a topology generator that updates the subgraph nodes and edges, and a feature generator that updates features associated to nodes in the Trojaned subgraph. The objective of GTA is to ensure that the Trojaned GNN model will return a desired label y^d that is different from the true label for graph inputs that contain the predefined 'triggered' subgraph.

2.3 Adversarial Learning Methods

In this thesis, we employ different adversarial learning methods to measure the minimum noise required for misclassification of clean samples and Trojan samples (stamped with the trigger). Mod-

ern adversarial attacks can be categorized into gradient-based attacks and query-based attacks [44] as well as a white-box setting where the adversary can access the internal data of the attacked models and a black-box setting where the adversary can only access the output returned by the attacked models.

We choose two SOTA white-box gradient-based adversarial learning methods of Fast Gradient Sign Method (FGSM) [22] and Basic Iterative Method [31]. The white-box setting is able to estimate the minimum noise for classification more accurately and consequently we use them to verify our intuition of Trojan samples are located far from the decision boundary.

To perform the adversarial attack directly in a black-box setting, query-based attacks have been proposed to generate adversarial examples without requiring gradient information from the affected models. They send numerous queries to update their optimization step. Similar to [16], we use Gaussian noise to estimate the minimum noise required for misclassification in a black-box setting.

In general, adversarial learning methods tend to find the minimum amount of noise make DNN model ($f(\cdot)$) to misclassify a given input sample (x):

$$\min_{\delta} -\mathcal{L}(f(x+\delta), y) + \|\delta\|_p \tag{2.1}$$

where y and δ are the true labels of x and adversarial noise respectively. Also, $\mathcal{L}(\cdot)$ is a loss function which measures the difference of the output of the model and the true label of samples.

- **Fast Gradient Sign Method (FGSM)** is a computationally efficient algorithm for adversarial example generation that causes the output of the model to be misclassified. It is a one step method wherein the algorithm perturbs a given input $x \in \mathbb{R}^d$ in the direction of the gradient by a variable step size ϵ as following.

$$x_{adv} = x + \epsilon \times \text{sign}(\nabla_x \mathcal{L}(f(x), y)) \tag{2.2}$$

  To ensure that x_{adv} gains a different label from x on DNN model, the value of ϵ needs to be sufficiently large. To learn the minimum adversarial noise, we start from small values,

and increase this value gradually till misclassification occurs. The amount of perturbation is calculated as $\|\delta\|_2 = \|\epsilon \times \text{sign}(\nabla_x \mathcal{L}(f(x), y)\|_2$.

- **Basic Iterative Method (BIM)** extends FGSM to a finite number of iterations. At each iteration k, the algorithm generates a perturbed input $x^k_{adv} = x^{k-1}_{adv} + \epsilon \times \text{sign}(\nabla_x \mathcal{L}(f(x^{k-1}_{adv}), y)$, x^{k-1}_{adv} is the perturbed input generated in the $(k-1)$ iteration. The algorithm is initialized with the given input $x^0_{adv} := x$ for which the adversarial example is required to be generated, and a constant step size ϵ is used in every iteration. Intuitively, every iteration of BIM is equivalent to the one step FGSM. This method returns a smaller amount of adversarial noise than FGSM, since it uses very small ϵ and iterates till misclassification occurs. Note that for a small value of ϵ more iteration is needed to find an adversarial example for sample x. The perturbation is computed as the l_2 norm of the difference between the input at 0-th iteration and the perturbed input at the last iteration (T), i.e. $\delta = \|x^T_{adv} - x^0_{adv}\|_2$. The advantage of using BIM is that one can carefully choose the step-size ϵ, and total number of iterations, say NT, for improving the success rate over FGSM.

- **HopSkipJump** is a query-based attack for learning adversarial noise with black-box access to the victim model. It is a strong and query efficient decision-based attack originally designed for deterministic classifiers. This attack only needs to observe the label returned by the model (not confidence values) to estimate the gradient direction. This method needs fewer queries compared to SOTA black-box adversarial learning methods. Consequently, it is a benchmark method for estimating the distance of a sample to decision boundary [16,38]. Note that FGSM and BIM both needs to have access to the model and consequently they achieve better estimation of minimum adversarial noise required for misclassification, while HopSkipJump is suitable real-world application in which only the predicted label is available to users.

2.4 Metrics for Evaluation

We describe metrics used to evaluate the effectiveness of backdoor attacks and of Trojan sample detection methods. We assume that defense mechanisms return a *positive* label if they identify a sample as Trojan.

- **True Positive Rate (TPR)** is the fraction of Trojan samples that received a positive label to the total number of Trojan samples. This metric measures the accuracy of a Trojan detection method in detecting the Trojan Samples. A high TPR is desirable in detecting backdoor attacks as it indicates that the detection mechanism is effective in identifying compromised samples. However, a high TPR must be balanced with a low false positive rate to avoid mistakenly flagging clean samples as Trojaned.

- **False Positive Rate (FPR)** is the fraction of clean samples that incorrectly received a positive label to the total number of clean samples. This metric measure utility loss associated with a Trojan sample detection method. A low FPR is desirable in detecting backdoor attacks as it indicates that the detection mechanism is not generating many false alarms, which can be costly in terms of resources and time spent investigating false positives.

- **Area Under the Receiver Operating Characteristic (ROC) Curve (AUC)** is a graph derived from TPR and FPR and shows the performance of the inference for a binary classifier e.g., defense mechanisms here are binary classifiers that assign a sample to either Trojan or clean classes. AUC measures the area under this curve and provides a measure of the ability to distinguish the data. In this sense, the higher the value, the better the performance.

- F_1 **score** is widely used to compare Trojan detection methods, and a higher F_1-score is better [13]. An effective Trojan detection method has a high detection accuracy (TPR) and low utility loss (FPR). The F_1-score combines these two metrics as:

$$F_1 = \frac{2 * TPR * (1 - FPR)}{TPR + (1 - FPR)} \tag{2.3}$$

A higher F_1-score indicates better Trojan detection rate with lower utility loss. To compare two Trojan detection methods, we use F_1-score.

- **Attack Success Rate (ASR)** is the fraction of Trojan samples that result in the DNN model returning the attacker's desired output [37]. ASR measures the fraction of Trojan samples bypassed a defense mechanism to the total of Trojan samples (FNR).

$$\text{Attack Success Rate (ASR)} = \frac{\text{\# successful trials}}{\text{\# total trials}} \tag{2.4}$$

Chapter 3

USER AND THREAT MODELS

In this chapter, we first introduce the threat model in Section 3.1 and then describe our assumptions on the capabilities of the attacker and user/defender in Sections 3.2 and 3.3.

3.1 Threat Model

Collecting large amounts of training data and obtaining the computational resources required to train large DNN models can be difficult for a user with limited resources. Deploying pre-trained DNN models and using cloud-based ML platforms and services are becoming increasingly popular to overcome these barriers. However, the trustworthiness of publicly available models, training data, and computational resources is a challenge since an adversary could leverage these to launch a backdoor attack [28]. Specifically, we consider scenarios where

- the users lack access to sufficient computational resources to train their own model and must rely on pre-trained models or untrusted platforms to train their model. In such a scenario, an adversary trains a Trojan model by *poisoning* the user's training dataset by embedding triggers into a subset of training samples

- an adversary makes users deploy a Trojaned model by making such models accessible to the user.

Because the adversary is aware of the Trojan trigger's identity, it can manipulate input samples so that the model returns a desired output that differs from the correct or true output. In our thesis, we consider the backdoor attacks that are caused due to the first scenario.

3.2 Attacker Capability

The adversary is assumed to have access to publicly available datasets in order to poison a subset of samples. We also assume that the attacker has sufficient computational resources to train a Trojan model. The attacker also does not share the attack strategy and keeps the identity of the target class (desired output) secret.

3.3 User Capability

The user has limited computational resources, and cannot train a large DNN. However, the user has sufficient resources to use a simple adversarial learning method [22] to compute the magnitude of adversarial noise that will result in misclassification of an input sample. We further assume that the user is aware of the possibility that the input sample or the model can be Trojaned, and aims to detect and remove malicious inputs. However, the user has no knowledge of the attacker strategy, and identities of the Trojan trigger and target class. On the other hand, the user is assumed to have access to a set of clean samples whose labels are known. The user has either *white-box access* (access to weights of the DNN model) or *black-box access* (access to only outputs of the DNN model). In each case, our objective is to develop a method to identify input samples that contain a Trojan trigger *without retraining or pruning the DNN model*.

Chapter 4

MULTI-DOMAIN TROJAN DETECTION: MOTIVATION

This chapter focusses on the motivation behind the development of *MDTD* by demonstrating that values of features at intermediate layers of the DNN corresponding to clean and Trojaned samples are different. As a consequence, clean and Trojaned input samples will behave differently when perturbed with noise. By employing the concept of a certified radius as described in [17], we are able to verify our hypothesis through the calculation of an estimated radius. This allows us to effectively differentiate between Trojan and clean samples, even when a user only has access to a limited number of clean samples.

4.1 Feature Value Visualization using t-SNE

For a DNN model with K layers, the first $K-1$ layers map an input x to the *feature space*, which typically has a lower dimension than the high-dimensional input space. The last layer of the DNN then uses the values in the feature space to make a decision about the input. For example, in an image classification task, the decision will be the identity of the class that the input is presumed to belong to. Thus, the output of the penultimate layer (layer $K-1$) of the DNN can then be interpreted as an indicator of the *perspective* of the DNN about the given input sample.

We use a t-distributed stochastic neighbor embedding (t-SNE) [62] to demonstrate that for the same output of the DNN model, values in the feature space for clean and Trojan samples are different. The t-SNE is a technique to visualize high-dimensional data through a two or three dimensional representation. For the CIFAR10, CIFAR100, GTSRB, SVHN, and Flowers102 datasets, we collect 200 samples corresponding to each of six different Trojan trigger types that are classified by the DNN model as belonging to the class $y^d = 6$ due to the presence of the trigger. We additionally collect 200 clean input samples that do not contain a Trojan trigger for whom the output of the

DNN model is $y = y^d$.

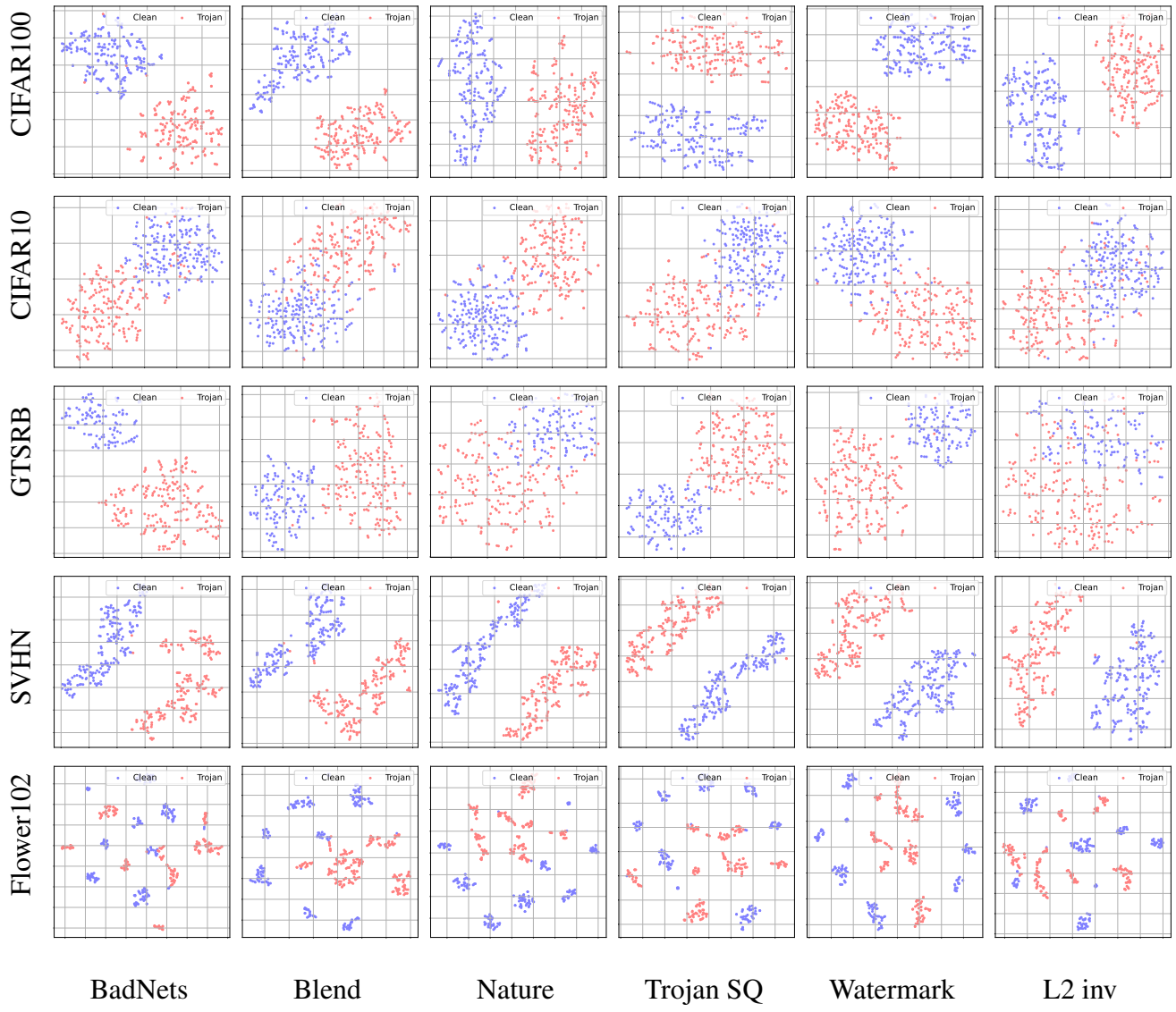


Figure 4.1: t-SNE visualizations for outputs of the penultimate layer of the DNN model for clean samples (blue dots) from the target class and Trojaned samples (red dots) misclassified to the target class.

Fig. 4.1 shows t-SNE representations of the feature values of these samples, i.e., the outputs at the penultimate layer of the DNN. We examine the CIFAR10, CIFAR100, GTSRB, SVHN, and

Flower102 datasets, and six different types of Trojan triggers for each dataset. The first $K-1$ layers of deep neural network map the input to feature space (a compressed variant of input) and then the last layer makes a decision using feature values generated by $K-1$ first layers for the input. More precisely, the output of the penultimate layer (the last fully connected layer before the output layer) indicates the viewpoint of a deep neural network about the input. Therefore, we obtain the feature values generated by the penultimate layer for 200 clean that truly belong to the class y^d and Trojan samples assigned to y^d due to the presence of the trigger, then we visualize these feature values in 2D using t-SNE.

We observe that the 200 clean samples (blue dots) from the target class (Class 6) can be easily distinguished from the 200 Trojan samples (red dots) for each of the Trojan trigger type in all five datasets. In each case, we observe that although the DNN model classifies both clean and Trojaned samples to the same class (Class 6), it generates different values for features in its penultimate layer. While the t-SNE visualization provides qualitative indicators of clean and Trojaned sample behavior, we are also interested in quantitative metrics. We use the notion of certified radius to distinguish between clean and Trojaned input samples. This result motivates an analysis of other properties of these samples in order to effectively distinguish between clean and Trojan samples.

The certified radius [17] is computed by estimating the distance to a decision boundary by perturbing samples with Gaussian noise. The certified radius is the radius of the largest ball centered at each sample within which the DNN model returns the same output. Table 4.1 presents the average certified radii for 200 clean samples and 200 Trojan samples with six different Trojan trigger types for the CIFAR10, CIFAR100, GTSRB, SVHN and Flowers102 datasets. We observe that the certified radius is higher for Trojaned samples than for clean samples.

Certified radius method finds the largest l_2 ball with radius of r around each sample in which DNN classifier f classifies any sample in that ball to the same class of the sample most likely. For example, in Table 4.1 we observe that for CIFAR100 dataset, the certified radius of clean samples on average is 0.005, while for a Trojan sample with badnets trigger is 0.65. Samples with higher certified radius are more robust to noise. Consequently, the minimum magnitude of noise required to make the DNN model misclassify a Trojan sample will be larger than the noise

Dataset	Clean	Badnets	Blend	Nature	Trojan SQ	Trojan WM	L2 inv
CIFAR100	0.005 (0.045)	0.657 (0.132)	0.103 (0.131)	0.880 (0.017)	0.759 (0.047)	0.893 (0.0001)	0.596 (0.036)
CIFAR10	0.015 (0.046)	0.875 (0.035)	0.308 (0.216)	0.893 (0.000)	0.893 (0.0001)	0.893 (0.0001)	0.648 (0.059)
GTSRB	0.308 (0.326)	0.663 (0.234)	0.749 (0.227)	0.882 (0.068)	0.857 (0.106)	0.499 (0.235)	0.239 (0.148)
SVHN	0.255 (0.268)	0.727 (0.178)	0.651 (0.271)	0.887 (0.035)	0.879 (0.083)	0.882 (0.062)	0.893 (0.0001)
Flower102	0.038 (0.173)	0.189 (0.163)	0.634 (0.302)	0.875 (0.07)	0.893 (0.001)	0.893 (0.001)	0.269 (0.148)

Table 4.1: Average (standard deviation) of certified radii for 200 clean and 200 Trojan samples for various datasets with 6 different Trojan trigger types.

addition required for a clean sample. We leverage this insight to inform the design of *MDTD*, a mechanism to distinguish clean and Trojan samples by using the minimum adversarial noise for misclassification without making any assumptions on the attacker's strategy.

4.2 Challenges in using t-SNE and Certified Radius

Although t-SNE and certified radius computation provide interesting insights into differences between clean and Trojan samples, there are two significant challenges which limit their use. Obtaining outputs at intermediate layers of the DNN requires access to model weights. This may not be feasible for users who only have black-box model access (i.e., access to only outputs of the model).

Certified radius generates many noisy variants of an input with different Gaussian noise distribution hyperparameters (variance) to estimate the distance of the input to the decision boundary. The process of computing the certified radius is also computationally expensive [17] since for each input sample, we will need to examine outputs for multiple perturbed variants of the sample with different hyperparameter values (variance of Gaussian), and then carry out an optimization procedure to calculate the certified radius. Even though both feature space and certified radius values can be used to distinguish Trojan samples from clean ones, they are not practical for the real world application.

MDTD overcomes these challenges by using adversarial learning techniques to estimate the distance of samples to a decision boundary, and then computing the smallest magnitude of adver-

sarial noise required to misclassify the input sample to infer whether the sample is Trojaned or not. *MDTD* is computationally inexpensive, effective against a variety of Trojan triggers across different input domains, and can be adapted for users with black-box model access.

4.3 Design and Working of MDTD

In this section, we describe the working of *MDTD*, a unified mechanism to detect Trojan triggers for a wide range of input domains. We first provide insight into the relationship between the distance of a sample to a decision boundary and a confidence value returned by the DNN model. Then, we explain how *MDTD* leverages this insight to effectively distinguish between Trojan and clean samples.

4.3.1 Insight

In Sec. 4.1, we showed that Trojan samples were more robust in that they required a larger magnitude of noise in order for the DNN model to misclassify them. We show that the distance of a sample to a decision boundary is proportional to the level of confidence of the DNN model in classifying the sample.

Consider a binary logistic regression model $h(x) := \frac{1}{1+e^{-(\omega^T x+b)}}$. The distance of a sample x to the decision boundary is given by $(\omega^T x + b)/\|\omega\|_2$. Samples that are classified with higher confidence have $h(x) \to 1$, which corresponds to $(\omega^T x + b) \to \infty$. This indicates the presence of a direct relationship between the distance of a sample to a decision boundary and the confidence of the model in classifying the sample. The relationship can be extenced to more complex classifier models including DNN classifiers.

For a DNN model with K layers, each of the first $K-1$ layers is a nonlinear function that maps the high-dimensional input to a lower dimensional feature space. The layers of $L_1, \cdots, L_{K-1}$ can be interpreted as a complicated function mapping the input-space to a feature-space and the layer L_K is a linear classifier which maps feature-space to classes. The last layer is a linear classifier that maps the feature space to classes. DNN models were shown to classify input samples containing a

Trojan trigger more confidently in [20]. Our insight is that as a result, since DNN models classify input samples that contain a trigger more confidently [20], the feature values generated for these samples should be farther away from a decision boundary than for clean samples. We describe how *MDTD* estimates the distance of samples to a decision boundary and then applies an outlier detection technique to distinguish between Trojan and clean samples. This indicates that Trojan samples have more robustness to noise and DNN models hardly change their decision on Trojan samples in presence of noise. Also, in [20], it has been demonstrated that DNN models classify the Trojan sample more confidently compared to clean samples. Here, we demonstrate that the confidence of the model for a sample is proportional to the distance of the sample to the decision boundary.

Consequently, we aim to answer the following questions:

- Since the samples containing a trigger are classified confidently, do these samples result in distinguishable feature values in DNN models?

- How more robust are samples containing a (natural/Trojan) trigger to adversarial noise compared to clean samples?

To answer the first question, we analyze the feature values generated by DNN models for both clean and samples containing a trigger using t-SNE method, and for answering the second question, we assess the minimum noise required for misclassification for both clean and samples containing a trigger.

4.3.2 Design

MDTD first estimates the distance of a given input sample to the decision boundary. Then, distances estimated in the previous step and the distance of a small number of clean samples to the decision boundary are used to identify whether the sample contains a Trojan trigger or not.

Estimating distance to decision boundary

MDTD uses adversarial learning techniques [22, 46] to estimate the minimum magnitude of noise perturbation, denoted δ, that will cause the DNN model to misclassify the sample. For an input sample x, the value of δ is obtained by maximizing a loss function $\mathcal{L}(\cdot, \cdot)$ that quantifies the difference between the output label of the DNN model for a perturbed variant of the input, denoted $f(x + \delta)$, and the true label of the input y. At the same time, we want the magnitude of δ to be small. This objective can be expressed as a regularized optimization problem with regularization constant λ as:

$$\min_{\delta} -\mathcal{L}(f(x + \delta), y) + \lambda \|\delta\|, \tag{4.1}$$

For a user with white-box access, we use the Fast Gradient Sign Method (FGSM) and Iterative Fast Gradient Sign Method (IFGSM) to solve Eqn. (4.1) due to their low computational cost [22, 31]. When a user has only black-box access, we apply a SOTA adversarial learning method called HopSkipJump [14] to estimate the minimum magnitude of noise required for misclassification of the input sample.

Outlier detection

Following our assumptions on user capability described in Sec. 3.1, information about the identity of the Trojan trigger and the target output class for Trojan samples is not available. Hence, the user will not be able to generate Trojan samples and estimate the distance of these samples to the decision boundary. However, due to the widespread availability of datasets, we assume that the user has access to a limited number of clean samples. We denote this set as $D_{user} = \{(x_1, y_1), (x_2, y_2), \cdots, (x_N, y_N)\}$. The user is assumed to have the ability to use the methods in *Step 1* above to estimate the minimum magnitudes of noise $\{\delta_1, \delta_2, \cdots, \delta_N\}$ required to misclassify these samples.

In order to determine whether a given input sample contains a Trojan trigger, we use an outlier detection technique first proposed in [57]. This method assumes that the distances of a clean sample (non-outliers) to the decision boundary follows a Gaussian distribution $\delta \sim \mathcal{N}(\mu, \sigma^2)$.

MDTD estimates the values of μ and σ using the values of $\{\delta_1, \delta_2, \cdots, \delta_N\}$ determined from the set D_{user}. Then, for a threshold α on the maximum tolerable false positive rate, any sample whose distance to the decision boundary satisfies $|\delta - \mu| > \alpha\sigma$ will be identified as containing a Trojan trigger (outlier). A small value of α results in a lower rate of detection of Trojan samples; a large value results in more clean samples being incorrectly identified as Trojan.

Choice of α:

We show how to choose α, given the maximum tolerable false positive rate, γ for a user of the DNN model.

When the size of D_{user} is sufficiently large, α can be expressed using the tail distribution of a standard Gaussian $Q(\cdot)$ and the complementary error function erfc$(\cdot)$ [4]. With μ and σ^2 denoting the sample mean and sample variance of entries in D_{user}, $Q(\alpha) = \frac{1}{2}\text{erfc}(\frac{\alpha}{\sqrt{2}})$ and $\text{erfc}(z) = \frac{2}{\sqrt{\pi}}\int_z^\infty e^{-t^2}\text{d}t$, we choose α such that $2Q(\alpha) \leq \gamma$, which gives the minimum value of α as:

$$\alpha = \sqrt{2}\,\text{erfc}^{-1}(\gamma). \tag{4.2}$$

If the size of data set D_{user} (denoted N) is quite small, then the sample mean μ can be estimated using a t-distribution with $\nu = N - 1$ degrees of freedom [33]. For a user-defined choice of γ, we denote the *critical t-value* as $T_{(1-\gamma/2),\nu}$. This represents the $1 - \frac{\gamma}{2}$ quantile of the t-distribution. In order to satisfy the maximum tolerable false positive rate, the parameter α will need to satisfy:

$$\alpha\sigma \geq T_{(1-\gamma/2),\nu}\frac{\mu}{\sqrt{N}},$$

which gives us

$$\alpha = T_{(1-\gamma/2),\nu}\frac{\mu}{\sigma\sqrt{N}}. \tag{4.3}$$

While Eqns. (4.2) and (4.3) provide a mathematical characterization of the threshold on the maximum false positive rate, this threshold can also be determined using ROC curves (Figs. 5.5 and 5.6), which provide a graphical representation of the relationship between true and false positive rates.

4.4 Mathematical Analysis of Proposed Mechanism

This section quantifies the worst-case false positive rate for an arbitrary clean sample x when using MDTD. We formally state this result below.

Theorem 4.4.1. Assume that the function f in Eqn. (4.1) is differentiable, and that its gradient $\nabla_x f$ is Lipschitz. For any clean sample x, the worst-case false positive rate of MDTD is given by

$$\mathbb{P}(\text{MDTD identifies } x \text{ as Trojan}) \leq 2e^{-(\zeta+\alpha\sigma)^2/2\sigma^2}, \tag{4.4}$$

where $\zeta = \frac{L\|\nabla_f \mathcal{L}\|\|x-\hat{x}\|}{2\lambda - L\|\nabla_f \mathcal{L}\|}$.

Proof. We first quantify the range of values of the distance of a clean sample x to a decision boundary in terms of its certified radius δ. We then use these range of values to determine the worst-case false positive rate of MDTD.

Let $\hat{x} =_{x' \in D_{user}} \|x' - x\|_1$ and $\hat{\delta}$ be the certified radius of $\hat{x}$. Since f is differentiable, δ and $\hat{\delta}$ must be critical points of the following:

$$-\nabla_f \mathcal{L} \nabla_x f|_{x+\delta} + 2\lambda\delta = 0,$$
$$-\nabla_f \mathcal{L} \nabla_x f|_{\hat{x}+\hat{\delta}} + 2\lambda\hat{\delta} = 0,$$

respectively. We thus have that

$$\|2\lambda\delta - 2\lambda\hat{\delta}\|_1 \tag{4.5a}$$
$$= \|\nabla_f \mathcal{L} \nabla_x f|_{x+\delta} - \nabla_f \mathcal{L} \nabla_x f|_{\hat{x}+\hat{\delta}}\|_1 \tag{4.5b}$$
$$\leq \|\nabla_f \mathcal{L}\|_1 L \|\hat{x} - x + \hat{\delta} - \delta\|_1 \tag{4.5c}$$
$$\leq \|\nabla_f \mathcal{L}\|_1 L (\|\hat{x} - x\|_1 + \|\hat{\delta} - \delta\|_1) \tag{4.5d}$$

where inequalities (4.5c) and (4.5d) hold by the assumption that $\nabla_x f$ is Lipschitz and the triangle inequality, respectively. Rearranging Eqn. (4.5) yields

$$\|\delta - \hat{\delta}\|_1 \leq \frac{L\|\nabla_f \mathcal{L}\|_1 \|x - \hat{x}\|_1}{2\lambda - L\|\nabla_f \mathcal{L}\|_1} \triangleq \zeta. \tag{4.6}$$

Given parameters μ and σ which are estimated using the clean data set D_{user}, we can quantify the worst-case false positive rate for clean sample x as $\mathbb{P}(\text{MDTD identifies } x \text{ as Trojan})$

$$= \mathbb{P}(|\delta - \mu| \geq \alpha\sigma) \tag{4.7}$$

$$\leq \mathbb{P}(|\hat{\delta} - \zeta - \mu| \geq \alpha\sigma). \tag{4.8}$$

Eqn. (4.8) follows from Eqn. (4.6). Using the tail bound of Normal distribution [45], we have that $\mathbb{P}(|\hat{\delta} - \zeta - \mu| \geq \alpha\sigma) \leq 2e^{-(\zeta+\alpha\sigma)^2/2\sigma^2}$.

Chapter 5

MULTI-DOMAIN TROJAN DETECTION: EVALUATION

In this chapter, we evaluate *MDTD* against SOTA Trojan detection methods on multiple image and graph-based input datasets by performing classification tasks. For each input domain, a brief description of the datasets employed, experimental setups used for evaluation, and the results of these experiments are presented.

5.1 Image-based Datasets

5.1.1 Datasets

We consider the following five datasets: CIFAR10 [30], CIFAR100 [30], SVHN [49], GTSRB [27], and Flower102 [50]. The CIFAR10 and CIFAR100 datasets each consist of 60000 color images that belong to one of 10 or 100 classes respectively. SVHN contains 600000 images of house numbers obtained from Google Street View. GTSRB is a dataset containing 52000 images of traffic signs, and Flower102 contains images of 102 commonly found flowers in the United Kingdom. In all our experiments, we use an image resolution of 32×32, and partition the dataset into 80% for training and 20% for test. For each dataset, we train one clean and six Trojan models with different trigger types (see Fig. 1.1). We assume that the desired output class of the DNN model for an adversary carrying out a backdoor attack is $y^d = 6$. Our experiments reveal that the classification accuracy and attack success rate is not affected by the choice of target class.

5.1.2 Network Structure

For CIFAR100, CIFAR10, and Flowers102 datasets, we used WideResnet deep neural networks [72]. For the GTSRB and SVHN datasets, we used a DNN with 4 convolutional layers with kernel sizes of $32, 32, 64$ and 64 respectively followed by a maxpooling layer and a fully connected layer of

size 4096. We train the models for 100 epochs with batch size of 64 using a stochastic gradient descent (SGD) optimizer. We set the learning rate to 0.001 and momentum to 0.9. For training robust DNNs, we use the robust learning technique proposed in [35]. We set the step size to 0.00784 and examined two different noise levels (= 0.01, 0.1). We used identical model 52 structures, parameters, and setup reported in [20] for our experiments on GNNs for graph-based inputs.

Datasets	Clean		BadNets		Blend		Nature		Trojan SQ		Trojan WM		L2 inv	
	Acc.	ASR	Acc.	ASR	Acc.	ASR	Acc.	ASR	Acc.	ASR	Acc.	ASR	Acc.	ASR
CIFAR100	55.69	NA	53.01	95.51	52.30	99.99	53.88	100	53.71	100	53.8	100	51.96	99.98
CIFAR10	82.57	NA	81.18	97.4	81.11	99.95	81.52	99.99	81.69	100	81.63	100	81.46	99.95
GTSRB	88.57	NA	84.19	91.91	88.5	95.45	87.41	99.98	88.31	99.03	85.24	99.76	87.89	90.06
SVHN	89.63	NA	89.46	95.59	89.79	99.28	90.44	99.92	90.26	99.72	90.48	99.84	91.36	99.69
Flower102	50.59	NA	47.25	89.61	46.18	99.12	46.37	100	46.67	100	48.14	100	44.71	97.16

Table 5.1: Classification accuracy (Acc. in $\%$) for clean samples and attack success rate (ASR in $\%$) of Trojan samples for six different Trojan triggers on five image-based datasets.

5.1.3 Trojan Triggers

We consider six different types of Trojan triggers that an adversary can embed into the image inputs provided to the DNN model: a white colored square (BadNets) [24], an image of a coffee mug (Nature) [15], a 'Hello Kitty' image blended into the background (Blend) [15], a multicolored square (Trojan SQ) [41], a colored circular watermark (Trojan WM) [41], and an 'invisible' trigger based on an $L2$-regularization of the input image (L2 inv) [36].

Table 5.1 compares the classification accuracy (Acc.) at test time of clean samples (input samples without a Trojan trigger) and attack success rate (ASR) for samples embedded with six different types of Trojan trigger. We observe that the Acc. values of Trojaned models is comparable to a clean model, while simultaneously achieving a high ASR value.

5.1.4 Setup

We evaluate *MDTD* against two SOTA Trojan input detection methods (i) DCT-based detector [73], and (ii) STRIP [20]. Both these methods aim to detect Trojan input samples without any information on attacker strategy.

DCT-based detector: This method uses the discrete cosine transform (DCT) to analyze the different frequencies present in the image. The authors of [73] showed that clean and Trojan samples consist of signals of different frequencies, which could then be used to effectively distinguish between them. We follow the experiment settings suggested in [73]- we use the complete set of training samples for each dataset and perturb clean samples by adding the BadNets Trojan trigger and Gaussian random noise. However, DCT-based detection requires using the entire training set, which is computationally expensive.

STRIP: The authors of [20] demonstrated that inputs containing a Trojan trigger were more robust to noise than clan inputs. Therefore, DNN classifiers will be less likely to change their decisions when these inputs are 'mixed' with other clean samples. We follow the setup from [20] in our experiments. We select 20 clean images at random, and 'mix' these with each input sample before providing it to the DNN classifier. An input is considered Trojan if the classifier returns the same output for at least 10 of the 'mixed' variants of the input, and is considered clean otherwise.

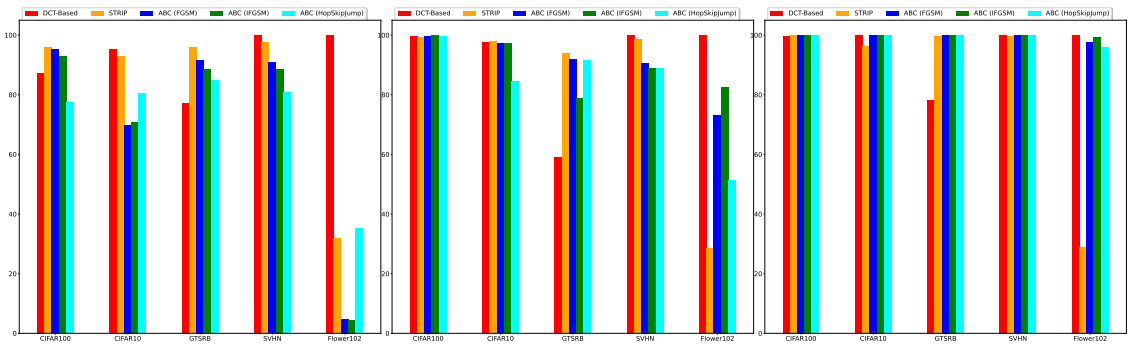
MDTD: MDTD uses estimates of distances to the decision boundary in order to distinguish between clean and Trojan samples. We consider cases when the user has *white-box* and *black-box* access to the DNN model. In the white-box setting, MDTD uses FGSM and IFGSM [22] adversarial learning methods to compute the minimum magnitude of noise δ required to misclassify a sample. In the black-box setting, MDTD uses the HopSkipJump adversarial learning method [14] to estimate distances to the decision boundary only based on outputs of the DNN model. Since the user does not have information about the Trojan trigger or target output class, MDTD uses a set of 500 clean samples randomly selected from the training set to determine a threshold distance to the decision boundary. A sample is identified as Trojan if the estimated distance δ is beyond this threshold.

	Attack	DCT-based	STRIP	MDTD (FGSM)	MDTD (IFGSM)	MDTD (HSJ)
CIFAR100	BadNets	0.88	0.94	0.95	0.94	0.86
	Blend	0.93	0.95	0.97	**0.98**	0.96
	Nature	0.93	0.95	**0.98**	**0.98**	**0.98**
	Trojan SQ	0.93	0.96	**0.98**	0.97	0.97
	Trojan WM	0.94	0.95	0.96	**0.98**	0.8
	L2 inv	0.94	0.95	**0.98**	0.97	0.97
CIFAR10	BadNets	0.69	0.81	0.78	0.77	**0.82**
	Blend	0.70	0.84	**0.92**	0.91	0.84
	Nature	0.7	0.81	**0.94**	0.92	0.91
	Trojan SQ	0.7	0.84	0.91	**0.93**	0.9
	Trojan WM	0.7	0.83	0.91	0.91	0.93
	L2 inv	0.7	0.84	**0.91**	**0.91**	0.85
GTSRB	BadNets	0.83	0.79	0.88	**0.9**	0.82
	Blend	0.71	0.74	0.75	**0.83**	0.76
	Nature	0.84	0.76	0.76	**0.9**	0.76
	Trojan SQ	**0.94**	0.79	0.8	0.89	0.79
	Trojan WM	**0.99**	0.76	0.87	0.92	0.91
	L2 inv	**0.93**	0.74	0.7	0.74	0.12
SVHN	BadNets	**0.99**	0.87	0.8	0.87	0.82
	Blend	**0.99**	0.76	0.76	0.89	0.83
	Nature	**0.99**	0.76	0.78	0.94	0.82
	Trojan SQ	**0.99**	0.74	0.71	0.91	0.82
	Trojan WM	**0.99**	0.76	0.83	0.93	0.89
	L2 inv	**0.99**	0.63	0.87	0.86	0.91
Flower102	BadNets	0	**0.48**	0.09	0.08	0.5
	Blend	0	0.45	0.78	**0.84**	0.65
	Nature	0	0.45	0.89	**0.93**	0.9
	Trojan SQ	0	0.88	**0.93**	0.92	0.92
	Trojan WM	0	0.75	**0.92**	**0.92**	**0.92**
	L2 inv	0	**0.79**	0.4	0.7	0.03

Table 5.2: F_1-scores for DCT-based detectors [73], STRIP [20], and MDTD for six different Trojan triggers on five image-based datasets.

5.2 Evaluation of MDTD against Image-based Datasets

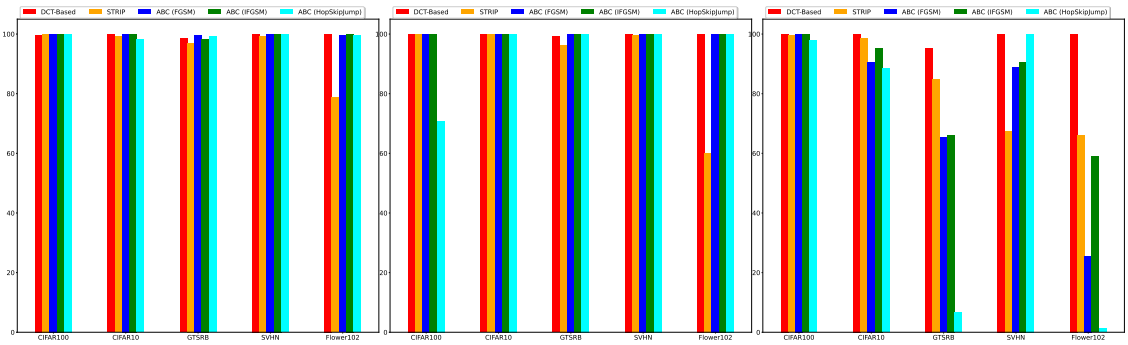
Table 5.2 shows the F_1-scores obtained for images embedded with different types of Trojan triggers when using the DCT-based detector [73], STRIP [20], and *MDTD*. We observe that MDTD obtains the highest F_1-scores in almost all cases. This indicates that MDTD is simultaneously able to achieve high true positive rates and small false positive rates (Eqn. (2.3)). We further observe that the F_1-score is higher for the variant of MDTD that uses the FGSM or IFGSM adversarial learning methods compared to HopSkipJump. This is because when using HopSkipJump, MDTD has access only to outputs of the model (black-box access). The lower F_1-scores for STRIP and for the DCT-based detector in the Flowers102 dataset is due to high false positive rates. The values of true and false positive rates for each case is shown in Fig. 5.1 - 5.4. We observe that DCT-based detectors are able to simultaneously achieve high true positive and low false positive rates for the SVHN and Flowers102 datasets.



Badnets Blend Nature

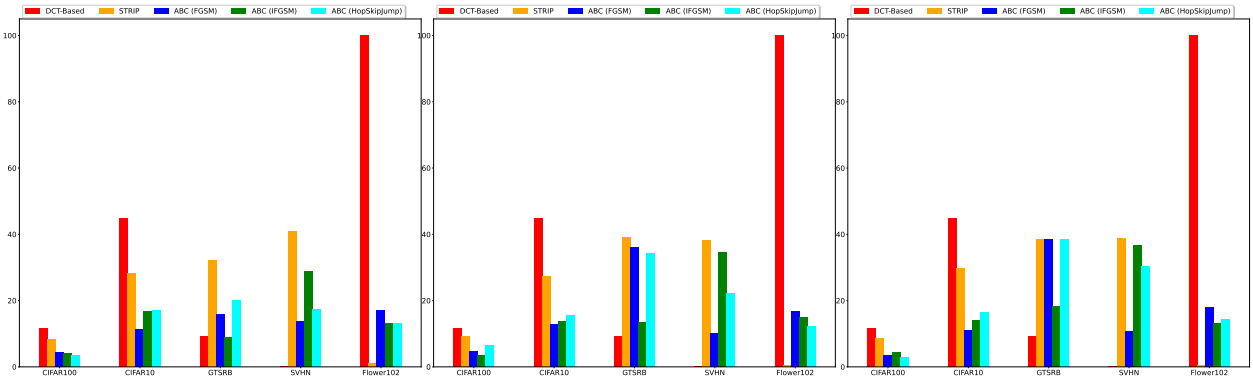
Figure 5.1: Trojan detection rate (true positive rates) in percentage for 5 datasets of CIFAR100, CIFAR10, GTSRB, SVHN and Flower102 and 3 backdoor attacks of Badnets Blend Nature.

The DCT-based detector obtains the highest F_1-score for all Trojan triggers in the SVHN dataset. This is because each image in this dataset contains only a few colors, which makes frequency components in these images easily separable. Consequently, the DCT-based detector is able



Trojan SQ Trojan WM L2 inv

Figure 5.2: Trojan detection rate (true positive rates) in percentage for 5 datasets of CIFAR100, CIFAR10, GTSRB, SVHN and Flower102 and 3 backdoor attacks of Trojan SQ Trojan WM L2 inv.

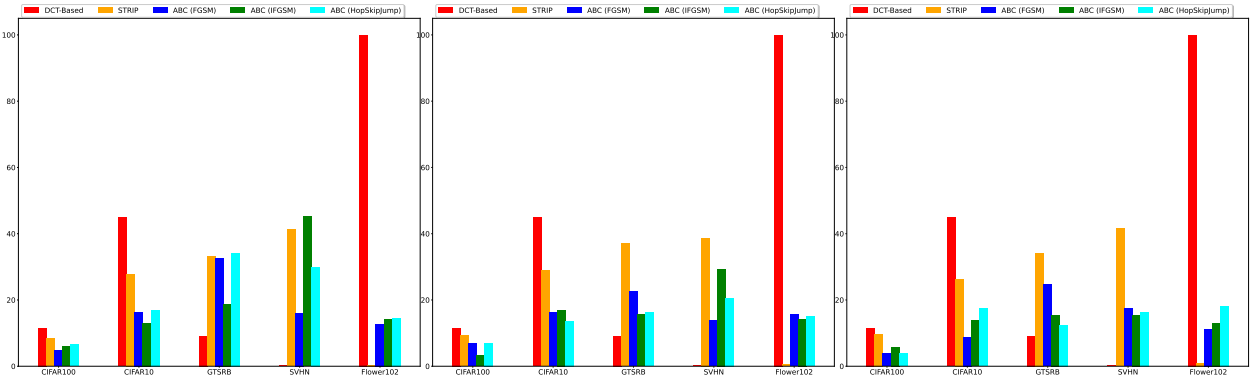


Badnets Blend Nature

Figure 5.3: False positive rates in percentage for 5 datasets of CIFAR100, CIFAR10, GTSRB, SVHN and Flower102 and 3 backdoor attacks of Badnets Blend Nature.

to simultaneously achieve large true positive and small false positive rates. However, DCT-based detection requires using the entire training set, which can be computationally expensive. Lower

F_1-scores of MDTD for the SVHN and GSTRB datasets can also be explained by the fact that the difference in magnitudes of the certified radii for clean and Trojaned samples is smaller for these datasets (see Table 4.1). To distinguish between clean and Trojaned samples for these datasets, more computationally complex adversarial learning methods such as [10, 47] might be required.



Trojan SQ Trojan WM L2 inv

Figure 5.4: False positive rates in percentage for 5 datasets of CIFAR100, CIFAR10, GTSRB, SVHN and Flower102 and 3 backdoor attacks of Trojan SQ Trojan WM L2 inv.

Fig. 5.5 shows ROC curves showing change in the true positive rate (Trojans being correctly identified as Trojan) with the change in the maximum tolerable false positive rate threshold α for *MDTD* using FGSM, IFGSM, and HopSkipJump adversarial learning methods for different types of Trojan triggers for five image-based input datasets. As noted earlier, the value of α plays a critical role in determining values of the true positive rate. Low values of true positive rates despite higher thresholds α when samples in the Flowers102 dataset are embedded with a BadNets (white square) or L2 inv Trojan triggers could be because clean input samples in this dataset might have white-colored flowers. This observation is also exemplified in the last row of Table 1 where values of the average certified radius for clean samples in the Flowers102 dataset and Trojan samples with the BadNets and L2 inv triggers are comparable and their variance is high.

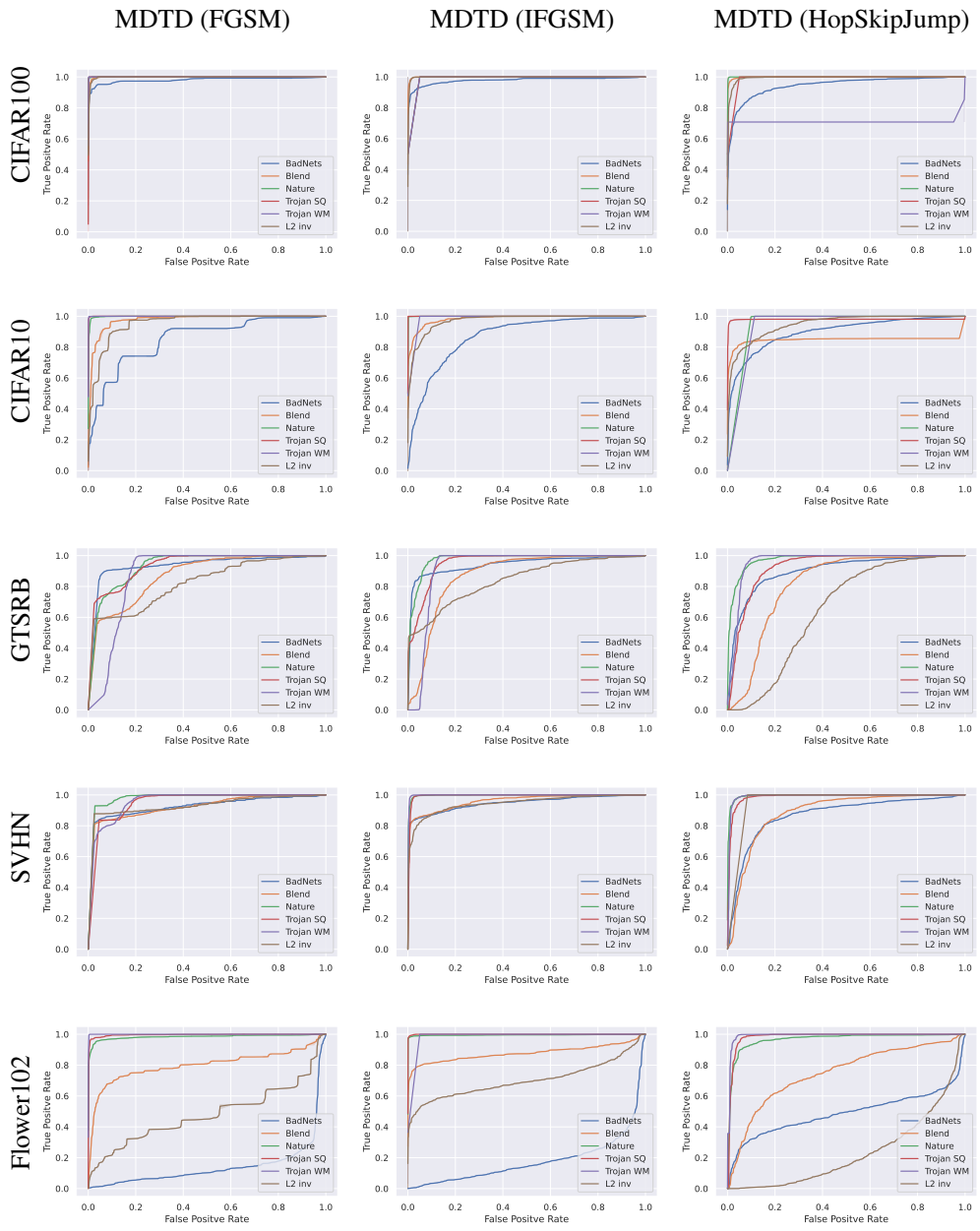


Figure 5.5: ROC curves showing change in accuracy of Trojan sample detection (True positive) with change in α for *MDTD* using FGSM, IFGSM, and HopSkipJump adversarial learning methods for different types of Trojan triggers for CIFAR100, CIFAR10 and GTSRB image-based datasets.

5.3 Graph-based Datasets

5.3.1 Datasets

We consider four graph datasets [67, 70]:

AIDS: This dataset consists 2000 graphs representing molecular compounds which are constructed from the AIDS Antiviral Screen Database of Active Compounds. The chemical structure of compounds is used to identify whether a patient belongs to one of the following three categories: confirmed active (CA), confirmed moderately active (CM), and confirmed inactive (CI).

WinMal: This dataset consists of 1361 call graphs of Windows Portable Executable (PE) files. Each file belongs to one of two categories: 'malware' or 'goodware'. Individual nodes of a call graph represent a function and edges between nodes are function calls.

Toxicant: This dataset captures molecular structures (encoded as a graph) of 10000 compounds studied for their effects of chemical interference in biological pathways. Effects are classified as 'toxic' or 'non-toxic'.

COLLAB: is a scientific collaboration dataset composed of 5000 graphs of ego networks of researchers in 3 fields- High Energy Physics, Condensed Matter Physics, and Astro Physics. The graph classification task is to identify which of the three fields an ego network belongs to.

5.3.2 Network Structure, Triggers and Setup

We use identical network structures, parameters, and GNN setups reported in [25] for our experiments with graph-based inputs.

5.4 Evaluating MDTD against Graph-based Datasets

Table 5.3 shows true positive rate, false positive rate, and F_1-score for the AIDS, COLLAB, WinMal, and Toxicant datasets. We use MDTD with the FGSM and IFSGM adversarial learning methods to estimate distances of samples to a decision boundary using $100 - 500$ clean samples (depending on the size of the dataset), and a threshold of $\alpha = 0.15$ on the false positive rate. We compute the smallest value $\delta^* \in \mathbb{R}^{|V| \times d}$ such that $h(f_t(\mathcal{G}, X)) \neq h(f_t(\mathcal{G}, X + \delta^*))$. We observe

that the F_1-score when using IFGSM is typically higher than when using FGSM due to the iterative nature of the IFGSM adversarial learning technique [31]. For the COLLAB dataset, the F_1-score is 1 since neither method could determine a value of adversarial noise that would misclassify input graphs embedded with a Trojan.

Task	MDTD (FGSM)			MDTD (IFGSM)		
	TPR	FPR	F_1	TPR	FPR	F_1
AIDS	84.07%	14.49%	0.85	96.29%	12.88%	$\mathbf{0.91}$
WinMal	96%	10.53%	0.93	100%	7.89%	$\mathbf{0.96}$
Toxicant	25.10%	16.17%	0.39	100%	12.77%	$\mathbf{0.93}$
COLLAB	100%	0%	$\mathbf{1}$	100%	0%	$\mathbf{1}$

Table 5.3: True positive rate (TPR), False positive rate (TPR), and F_1-score of MDTD for four graph datasets.

Fig. 5.6 shows ROC curves showing change in the true positive rate (Trojans being correctly identified as Trojan) with the change in the maximum tolerable false positive rate threshold α for *MDTD* using the FGSM and IFGSM adversarial learning methods for four graph-based input datasets. We observe that MDTD (IFGSM) is more effective in distinguishing between clean and Trojan samples than MDTD (FGSM). This behavior is possibly due to the iterative nature of the IFGSM algorithm [22, 31].

Fig. 5.7 shows the t-SNE representations of feature values of these samples, i.e., the outputs at the penultimate layer of the GNN. We collect 200 clean graph inputs and 200 graph inputs embedded with a Trojan trigger for the AIDS, COLLAB, WinMal, and Toxicant datasets. Similar to our observations for image-based inputs in Fig. 2, our experiments reveal that clean samples (blue dots) can be easily distinguished from Trojan samples (red dots) in all four graph datasets as they have different feature values in the penultimate layer.

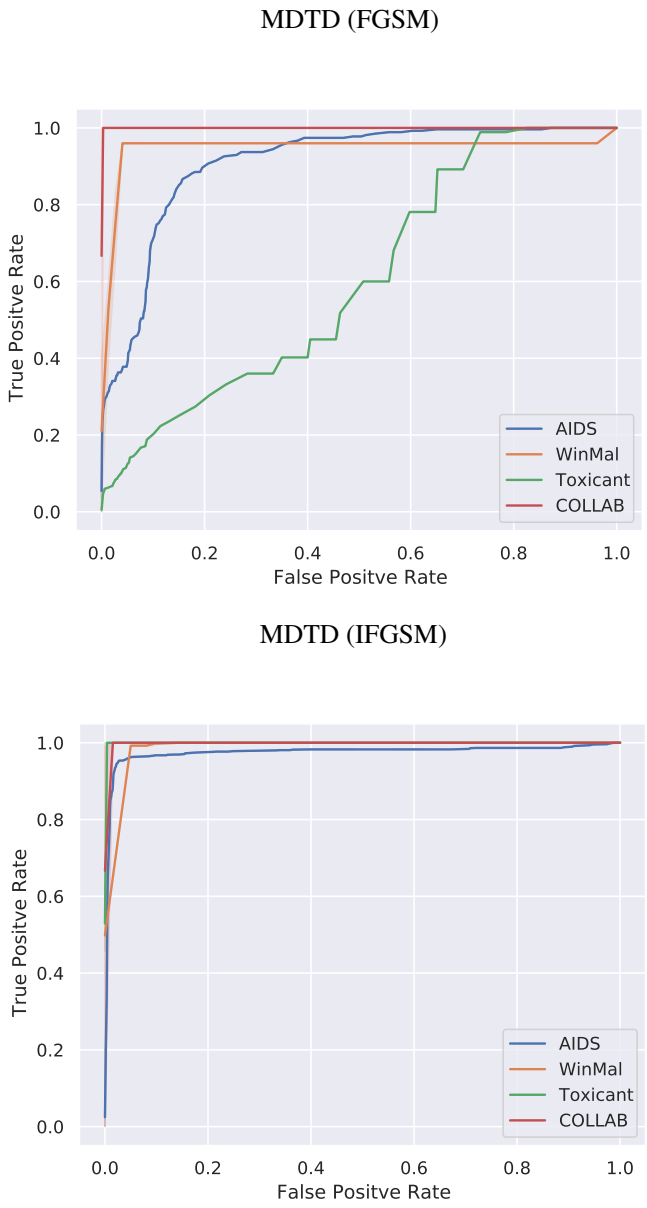


Figure 5.6: ROC curves showing change in accuracy of Trojan sample detection (true positive) with the change in maximum tolerable false positive rate α for MDTD using FGSM and IFGSM adversarial learning methods for four graph-based datasets.

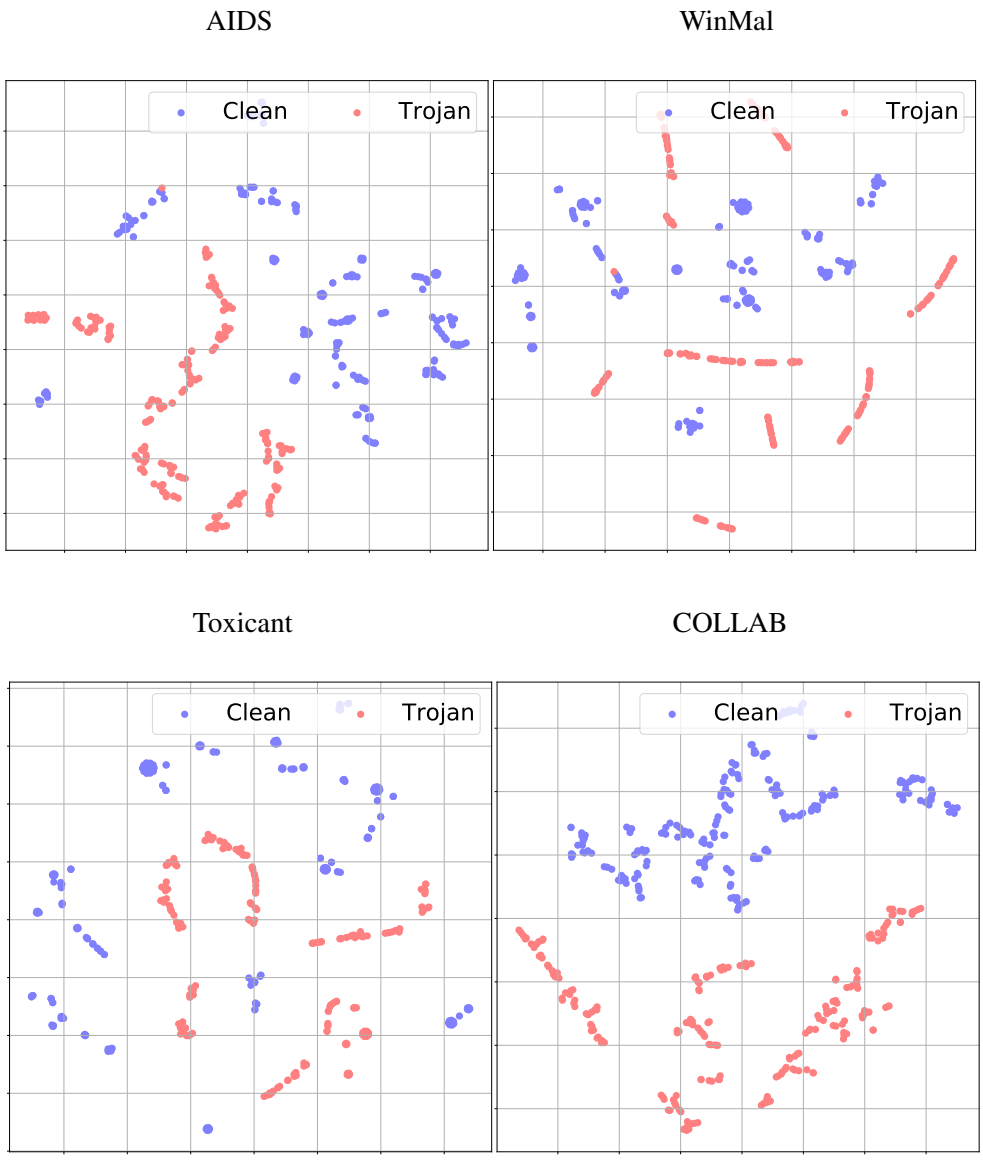


Figure 5.7: t-SNE visualizations for outputs of the penultimate layer of the GNN model for 200 clean (blue dots) and 200 graph inputs embedded with a Trojan (red dots) for graph-based datasets.

5.5 Robust DNNs and MDTD Performance

Although MDTD estimates the magnitude of adversarial noise required to misclassify an input sample and uses this to determine whether the sample contains a Trojan trigger or not, this information can be potentially exploited by an adversary. Specifically, an adversary that has additional knowledge of the defense mechanism deployed by the user could use DNN models that are trained on both, clean samples and samples that are perturbed with adversarial noise [43]. Such DNN models are termed robust DNNs. The use of such models has been shown to provide improved levels of robustness to noise, but at the cost of a lower classification accuracy for clean samples [43, 74]. We carry out experiments to investigate the effect of using robust DNN models on the performance of MDTD. We use the robust learning technique proposed in [43]. We set the step size to 0.00784 and examine two noise levels ($\epsilon = 0.01, 0.1$).

Table 5.4 shows the classification accuracy, ASR, and F_1-scores for DNN models trained with two levels of adversarial noise perturbations ϵ for different types of triggers on the CIFAR10 dataset. We observe that smaller values of ϵ is not effective in deteriorating the performance of *MDTD* (compare F_1 scores for CIFAR10 in Table 5.2 for $\epsilon = 0$ and $\epsilon = 0.01$ in Table 5.4). Increasing the value of ϵ to 0.1 causes a significant drop in the values of F_1-scores of *MDTD*. However, this comes at the cost of significantly lower classification accuracy, which demonstrates a much lower utility for the attacker when using such a large magnitude of adversarial noise.

ϵ	Attack	Acc.	ASR	F_1: MDTD (IFGSM)
0	BadNets	81.18	97.4	0.77
	Blend	81.11%	99.95%	0.91
	Nature	81.52%	99.99%	0.92
	Trojan SQ	81.69%	100%	0.93
	Trojan WM	81.63%	100%	0.91
	L2 inv	81.46%	99.95%	0.91
0.01	BadNets	79.23	96.03	0.79
	Blend	80.19%	99.88%	0.74
	Nature	80%	100%	0.93
	Trojan SQ	80.06%	100%	0.91
	Trojan WM	79.81%	100%	0.90
	L2 inv	80.47%	99.98%	0.81
0.1	BadNets	36.35	53.74	0.53
	Blend	40.25%	86.88%	0.42
	Nature	40.32%	100%	0.85
	Trojan SQ	33.62%	100%	0.82
	Trojan WM	26.51%	100%	0.68
	L2 inv	35.28%	51.90%	0.58

Table 5.4: F_1-scores of MDTD (IFGSM) for DNN models trained with different levels of adversarial noise level robustness ($\epsilon = 0, 0.01, 0.1$) for the CIFAR10 dataset with six different types of Trojan triggers.

Chapter 6

CONCLUSIONS AND FUTURE DIRECTIONS

Sections 6.1 of this chapter discuss the possible improvements that can be done for the proposed MDTD mechanism (introduced in Chapter 4.1) and Section 6.2 presents the concluding remarks of this master's dissertation.

6.1 Future Work

We list the following research directions for my dissertation research:

- **Black-box Adversarial Learning for DNNs with Graph Inputs:**

  Multiple black-box adversarial learning techniques have been proposed for DNN models with image inputs [10, 14, 47]. For DNNs with graph inputs, to the best of our knowledge, the only black-box adversarial learning method was presented in [48]. This approach considered the removal or addition of nodes and edges as a 'noise' term, which does not apply to our framework, since we assume that triggers are embedded into feature values of nodes. Adapting SOTA black-box adversarial learning methods such as HopSkipJump [14] beyond image-based inputs to use MDTD for Trojan input detection is an interesting direction of future research.

- **Expanding MDTD input detection to other input domains:**

  Sequence-to-sequence (seq2seq) text models [58] map an input sequence $x = \{x_1, \ldots, x_k\}$ to an output sequence $y = \{y_1, \ldots, y_n\}$, possibly of different length. A backdoor attack on a text model consists of using a specific term or word in the input sequence as the trigger.

Similar to Trojaned models for images and graphs, the adversary's objective is to ensure that the Trojaned text model behaves normally for text inputs that do not contain the trigger word or phrase, and produces an adversary-desired output for inputs that contain the trigger word or phrase. The authors of [6] showed that Trojan triggers could be embedded into seq2seq text models that perform text summarization tasks. The presence of a trigger in the input produced a summarized output that had different properties than desired. For e.g., a trigger in the input text caused the output text to have a different 'sentiment' than in the case when the trigger was absent (clean input). A metric that is widely used to determine the quality of text summarization is the *ROUGE* score [39], which compares the output of a text model with a human-written summary. Although the trigger word in a text summary can be identified by brute-force, such a procedure will be computationally expensive. Further, SOTA in Trojan detection for text models focus on text classification tasks [56]. We believe that MDTD can be used in conjunction with *ROUGE* scores to efficiently detect the presence of a trigger word in inputs to text models when a user has access only to a small number of clean inputs. This can be accomplished by using adversarial learning methods to estimate a threshold on the number of words removed that results in the maximum change in *ROUGE* scores.

- **Choice of learning methods:**

The choice of adversarial learning method used to estimate the minimum required magnitude of noise for changing the output of the model has a significant impact on the performance of MDTD. For example, in the black-box setting, MDTD (HopSkipJump) obtains a lower F_1-score, since it cannot estimate this noise well-enough. On the other hand, in the white-box setting, we use BIM and FGSM which are computationally inexpensive methods that allow effective estimation of this noise parameter. Other white-box adversarial learning methods with higher computational complexity that can estimate the minimum required noise with greater precision can be studied and formal guarantees on their performance should be ex-

plored.

- **Natural Backdoor Attacks:**

  Our focus so far has been on embedded backdoor attacks in which an adversary embeds a predefined trigger into inputs given to DNN models. However, an attacker who has knowledge of the user's model can learn an adversarial noise which can then be used as a *natural backdoor* [60] that can be used to ensure that the DNN model output is the adversary-desired target class. Such an adversarial noise is also termed a *universal perturbation*, and it was shown in [46] that it was possible for an adversary to learn a universal perturbation and use it to simulate behavior similar to a backdoor attack. We further observed that the certified radii of samples perturbed with the universal noise is higher compared to clean samples. We propose to investigate the development of a variant of MDTD that will be able to distinguish between clean samples and samples with a natural Trojan trigger.

6.2 Conclusion

In this dissertation, we explored the challenges against training DNNs, the most significant one being the potential for adversarial attacks, which can be carried out against both image and graph input datasets. Adversarial attacks are intentional modifications to input data that are designed to trick the DNN into making incorrect predictions. Leveraging the insight that input samples containing a Trojan trigger are typically located farther away from a decision boundary than clean samples, we proposed MDTD - a multi domain Trojan detector for deep neural networks. Trojan triggers are hidden patterns or features that have been inserted into an input sample with the intention of causing the DNN to misclassify the sample when the trigger is activated. Such triggers can be inserted into the training data by an attacker, leading to a Trojaned model. MDTD did not require knowledge of attacker strategy, which may not always be available, in order to detect Trojans in image and graph-based input datasets. By using adversarial learning techniques to estimate the distance of a sample to a decision boundary, MDTD then computed the smallest magnitude

of adversarial noise required for the DNN model to misclassify the sample. This was then used to infer whether the sample was Trojaned or not. MDTD was shown to be more effective than SOTA methods in identifying Trojaned samples through extensive evaluations on five image and four graph datasets. Moreover, MDTD was able to determine if an input sample contained a Trojan trigger even when a user of the DNN model had access to only a small number of clean samples. This is a significant advantage, as it reduces the amount of data required for Trojan detection and makes it easier to detect Trojan triggers in real-world applications. This research contributes to the ongoing efforts to improve the security and reliability of DNNs, which are essential for their adoption in critical applications.

BIBLIOGRAPHY

[1] Amazon, Machine learning at AWS, 2018.

[2] BigML Inc. bigml, 2018.

[3] Caffe, Caffe Model Zoo, 2018.

[4] Milton Abramowitz and Irene A Stegun. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, volume 55. US Government Printing Office, 1964.

[5] William Aiken, Hyoungshick Kim, Simon Woo, and Jungwoo Ryoo. Neural network laundering: Removing black-box backdoor watermarks from deep neural networks. *Computers & Security*, 106:102277, 2021.

[6] Eugene Bagdasaryan and Vitaly Shmatikov. Spinning language models: Risks of propaganda-as-a-service and countermeasures. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1532–1532. IEEE Computer Society, 2022.

[7] Neha Bansal, Arun Sharma, and R. K. Singh. A review on the application of deep learning in legal domain. In John MacIntyre, Ilias Maglogiannis, Lazaros Iliadis, and Elias Pimenidis, editors, *Artificial Intelligence Applications and Innovations*, pages 374–381. Springer International Publishing, 2019.

[8] Jacob Buckman, Aurko Roy, Colin Raffel, and Ian Goodfellow. Thermometer encoding: One hot way to resist adversarial examples. In *International Conference on Learning Representations*, 2018.

[9] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proc. ACM Workshop on Artificial Intelligence and Security*, pages 3–14, 2017.

[10] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy (SP)*, pages 39–57. IEEE, 2017.

[11] Mainak Chakraborty, Harish C. Kumawat, Sunita Vikrant Dhavale, and Arockia Bazil Raj A. Application of dnn for radar micro-doppler signature-based human suspicious activity recognition. *Pattern Recognition Letters*, 2022.

[12] Tox21 Data Challenge. https://tripod.nih.gov/tox21/. 2014.

[13] Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian Molloy, and Biplav Srivastava. Detecting backdoor attacks on deep neural networks by activation clustering. *Artificial Intelligence Safety Workshop, (AAAI)*, 2019.

[14] Jianbo Chen, Michael I Jordan, and Martin J Wainwright. HopSkipJumpAttack: A query-efficient decision-based attack. In *IEEE Symposium on Security and Privacy*, pages 1277–1294, 2020.

[15] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

[16] Christopher A Choquette-Choo, Florian Tramer, Nicholas Carlini, and Nicolas Papernot. Label-only membership inference attacks. In *International Conference on Machine Learning*, pages 1964–1974. PMLR, 2021.

[17] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In *International Conference on Machine Learning*, pages 1310–1320. PMLR, 2019.

[18] Guneet S Dhillon, Kamyar Azizzadenesheli, Zachary C Lipton, Jeremy D Bernstein, Jean Kossaifi, Aran Khanna, and Animashree Anandkumar. Stochastic activation pruning for robust adversarial defense. In *International Conference on Learning Representations*, 2018.

[19] National Center for High-performance Computing (NCHC) and Taiwan Computer Security Incident Response Team (TWCSIRT). Malware knowledge base.

[20] Yansong Gao, Change Xu, Derui Wang, Shiping Chen, Damith C Ranasinghe, and Surya Nepal. STRIP: A defence against trojan attacks on deep neural networks. In *Proceedings of the 35th Annual Computer Security Applications Conference*, pages 113–125, 2019.

[21] Cheng Gongye, Hongjia Li, Xiang Zhang, Majid Sabbagh, Geng Yuan, Xue Lin, Thomas Wahl, and Yunsi Fei. New passive and active attacks on deep neural networks in medical applications. In *Proceedings of the 39th International Conference on Computer-Aided Design*. Association for Computing Machinery, 2020.

[22] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *International Conference on Learning Representations (ICLR).*, 2015.

[23] Sorin Grigorescu, Bogdan Trasnea, Tiberiu Cocias, and Gigel Macesanu. A survey of deep learning techniques for autonomous driving. *Journal of Field Robotics*, 37(3):362–386, 2020.

[24] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. BadNets: Evaluating backdooring attacks on deep neural networks. *IEEE Access*, 7:47230–47244, 2019.

[25] William L Hamilton. Graph representation learning. *Synthesis Lectures on Artifical Intelligence and Machine Learning*, 14(3):1–159, 2020.

[26] William L Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*, 2017.

[27] Sebastian Houben, Johannes Stallkamp, Jan Salmen, Marc Schlipsing, and Christian Igel. Detection of traffic signs in real-world images: The German Traffic Sign Detection Benchmark. In *International Joint Conference on Neural Networks*, number 1288, 2013.

[28] Ling Huang, Anthony D. Joseph, Blaine Nelson, Benjamin I.P. Rubinstein, and J. D. Tygar. Adversarial machine learning. page 43–58. Association for Computing Machinery, 2011.

[29] Bingchen Jiang and Zhao Li. Defending against backdoor attack on graph neural network by explainability. *arXiv preprint arXiv:2209.02902*, 2022.

[30] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *Technical Report, University of Toronto*, 2009.

[31] Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *Artificial Intelligence Safety and Security*, pages 99–112. Chapman and Hall/CRC, 2018.

[32] Keita Kurita, Paul Michel, and Graham Neubig. Weight poisoning attacks on pre-trained models. *arXiv preprint arXiv:2004.06660*, 2020.

[33] Kenneth L Lange, Roderick JA Little, and Jeremy MG Taylor. Robust statistical modeling using the t distribution. *Journal of the American Statistical Association*, 84(408):881–896, 1989.

[34] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graphs over time: Densification laws, shrinking diameters and possible explanations. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*. Association for Computing Machinery, 2005.

[35] Hang Li. Deep learning for natural language processing: advantages and challenges. *National Science Review*, pages 24–26, 2017.

[36] Shaofeng Li, Benjamin Zi Hao Zhao, Jiahao Yu, Minhui Xue, Dali Kaafar, and Haojin Zhu. Invisible backdoor attacks against deep neural networks. *Annual Network and Distributed System Security Symposium (NDSS)*, 2019.

[37] Yiming Li, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. Backdoor learning: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–18, 2022.

[38] Zheng Li and Yang Zhang. Membership leakage in label-only exposures. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, pages 880–895, 2021.

[39] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.

[40] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *International Symposium on Research in Attacks, Intrusions, and Defenses*, pages 273–294. Springer, 2018.

[41] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. Trojaning attack on neural networks. *Annual Network and Distributed System Security Symposium (NDSS)*, 2018.

[42] Andrew L. Maas, Peng Qi, Ziang Xie, Awni Y. Hannun, Christopher T. Lengerich, Daniel Jurafsky, and Andrew Y. Ng. Building dnn acoustic models for large vocabulary speech recognition. *Computer Speech Language*, 41:195–213, 2017.

[43] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *International Conference on Learning Representations (ICLR).*, 2018.

[44] Maurizio Lucia Guarracino Manzo, Giordano. Performance evaluation of adversarial attacks on whole-graph embedding models. In *Learning and Intelligent Optimization*, pages 219–236. Springer, 2021.

[45] Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomization and Probabilistic Techniques in Algorithms and Data Analysis*. Cambridge University Press, 2017.

[46] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1765–1773, 2017.

[47] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. DeepFool: A simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2574–2582, 2016.

[48] Jiaming Mu, Binghui Wang, Qi Li, Kun Sun, Mingwei Xu, and Zhuotao Liu. A hard label black-box adversarial attack against graph neural networks. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 108–125, 2021.

[49] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

[50] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008.

[51] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael Wellman. Towards the science of security and privacy in machine learning. *IEEE European Symposium on Security and Privacy*, 2018.

[52] Ricardo Rendall, Ivan Castillo, Bo Lu, Brenda Colegrove, Michael Broadway, Leo H. Chiang, and Marco S. Reis. Image-based manufacturing analytics: Improving the accuracy of an industrial pellet classification system using deep neural networks. *Chemometrics and Intelligent Laboratory Systems*, pages 26–35, 2018.

[53] Ricardo Rendall, Ivan Castillo, Bo Lu, Brenda Colegrove, Michael Broadway, Leo H. Chiang, and Marco S. Reis. Image-based manufacturing analytics: Improving the accuracy of an industrial pellet classification system using deep neural networks. *Chemometrics and Intelligent Laboratory Systems*, pages 26–35, 2018.

[54] Ryan A. Rossi and Nesreen Ahmed. The network data repository with interactive graph analytics and visualization. In *AAAI Conference on Artificial Intelligence*, 2015.

[55] Guangyu Shen, Yingqi Liu, Guanhong Tao, Shengwei An, Qiuling Xu, Siyuan Cheng, Shiqing Ma, and Xiangyu Zhang. Backdoor scanning for deep neural networks through k-arm optimization. In *International Conference on Machine Learning*, pages 9525–9536. PMLR, 2021.

[56] Xuan Sheng, Zhaoyang Han, Piji Li, and Xiangmao Chang. A survey on backdoor attack and defense in natural language processing. *arXiv preprint arXiv:2211.11958*, 2022.

[57] Walter Andrew Shewhart. *Economic control of quality of manufactured product*. Macmillan And Co Ltd, London, 1931.

[58] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in Neural Information Processing Systems*, 27, 2014.

[59] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations (ICLR)*, 2014.

[60] Guanhong Tao, Yingqi Liu, Guangyu Shen, Qiuling Xu, Shengwei An, Zhuo Zhang, and Xiangyu Zhang. Model orthogonalization: Class distance hardening in neural networks for better security. In *IEEE Symposium on Security and Privacy (SP)*, volume 3, 2022.

[61] Brandon Tran, Jerry Li, and Aleksander Madry. Spectral signatures in backdoor attacks. *Advances in Neural Information Processing Systems*, 31, 2018.

[62] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(11), 2008.

[63] Miguel Villarreal-Vasquez and Bharat Bhargava. ConFoc: Content-focus protection against trojan attacks on neural networks. *arXiv preprint arXiv:2007.00711*, 2020.

[64] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. Neural Cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 707–723. IEEE, 2019.

[65] Chongke Wu, Sicong Shao, Cihan Tunc, and Salim Hariri. Video anomaly detection using pre-trained deep convolutional neural nets and context mining. In *2020 IEEE/ACS 17th International Conference on Computer Systems and Applications (AICCSA)*, pages 1–8, 2020.

[66] Huijun Wu, Chen Wang, Yuriy Tyshetskiy, Andrew Docherty, Kai Lu, and Liming Zhu. Adversarial examples on graph data: Deep insights into attack and defense. *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI'19)*, 2019.

[67] Zhaohan Xi, Ren Pang, Shouling Ji, and Ting Wang. Graph backdoor. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 1523–1540, 2021.

[68] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille. Mitigating adversarial effects through randomization. In *International Conference on Learning Representations*, 2018.

[69] Ziang Yan, Yiwen Guo, and Changshui Zhang. Deep Defense: Training dnns with improved adversarial robustness. *Advances in Neural Information Processing Systems*, 31, 2018.

[70] Pinar Yanardag and S.V.N. Vishwanathan. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, page 1365–1374, 2015.

[71] Kun-Hsing Yu, Andrew L Beam, and Isaac S Kohane. Artificial intelligence in healthcare. *Nature Biomedical Engineering*, 2(10):719–731, 2018.

[72] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016.

[73] Yi Zeng, Won Park, Z Morley Mao, and Ruoxi Jia. Rethinking the backdoor attacks' triggers: A frequency perspective. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16473–16481, 2021.

[74] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. Theoretically principled trade-off between robustness and accuracy. In *International Conference on Machine Learning*, pages 7472–7482. PMLR, 2019.

[75] Zaixi Zhang, Jinyuan Jia, Binghui Wang, and Neil Zhenqiang Gong. Backdoor attacks to graph neural networks. In *Proceedings of the 26th ACM Symposium on Access Control Models and Technologies*, pages 15–26, 2021.

[76] Xiaojun Zhou, Jingyi He, and Chunhua Yang. An ensemble learning method based on deep neural network and group decision making. *Knowledge-Based Systems*, 2022.

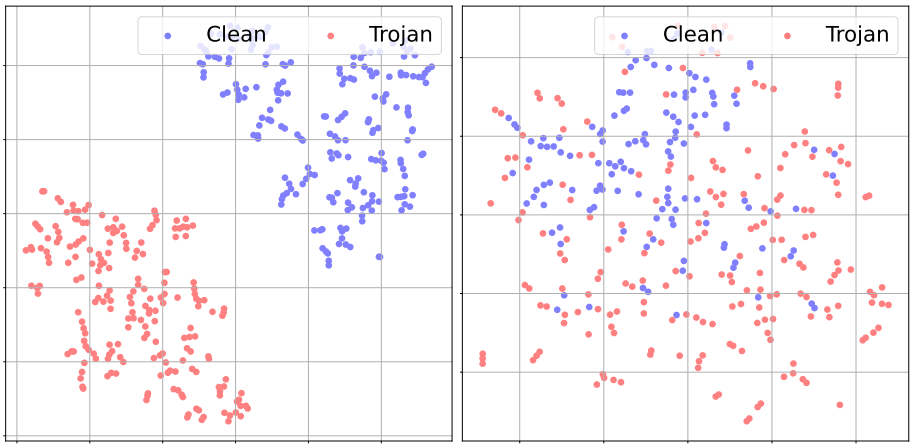
[77] Liuwan Zhu, Rui Ning, Cong Wang, Chunsheng Xin, and Hongyi Wu. Gangsweep: Sweep out neural backdoors by GAN. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 3173–3181, 2020.

[78] Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. Texygen: A benchmarking platform for text generation models. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 1097–1100, 2018.

Appendix A

T-SNE VISUALIZATIONS FOR DNN MODELS EMBEDDED WITH A NATURAL BACKDOOR

Fig. A.1 shows that DNN models containing natural backdoors also generate different values in the feature space at intermediate layers for clean and Trojaned samples. We further observe that the certified radii of samples perturbed with the universal noise is higher compared to clean samples. However, our experiments reveal that Trojan detection methods, including DCT-based detection, STRIP, and MDTD have high false positive rates for the natural backdoor attack. This is exemplified by the fact that clean and Trojan samples are not easily separable from the t-SNE visualization for the GTSRB dataset in Fig. A.1.



CIFAR100 GTSRB

Figure A.1: t-SNE visualizations for the output of the penultimate layer of the DNN model for 200 clean samples (blue dots) from the target class and 200 Trojaned samples embedded with a *natural backdoor* (red dots) misclassified to the target class for the CIFAR100 and GTSRB datasets.